MASTER EN SCIENCES INFORMATIQUES

La base de données dans la gestion de réseaux : étude d'un cas concret

Briard, François

Award date:
1993

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique
Rue Grandgagnage, 21b
B-5000 NAMUR

La base de données dans la gestion de réseaux: étude d'un cas concret

François Briard

Promoteur : Philippe van Bastelaer

Mémoire présenté en vue
de l'obtention du grade de
Licencié et Maître en Informatique

Année académique 1992 - 1993

Résumé

Ce mémoire traite de la place de la base de données au sein de la gestion de réseaux et plus particulièrement celle regroupant des informations générales sur le réseau et non directement liées à son fonctionnement technique.

Dans une première partie, on analyse ce que pourrait être, de manière générale, l'information liée à la gestion de réseau. Ensuite, on étudie comment est définie l'information de gestion par deux standards de gestion de réseaux: SNMP et CMIP-CMIS de OSI et la place qu'il font aux différents types d'informations définis précédemment.

La seconde partie est une analyse approfondie d'une base de données de gestion de réseaux appelée LANDB et développée au CERN (Organisation Européenne pour la Recherche Nucléaire) à Genève. Cette analyse se base sur le travail effectué pendant un stage de fin d'études au CERN.
Après une brève description des moyens du CERN en matière de réseaux, on analyse rapidement une première application ayant servi de base à l'application LANDB.
Ensuite, après une analyse des objectifs généraux de LANDB, on effectue une description en détail de la structure de la base de données puis des interfaces utilisateurs. De plus, lors de l'analyse des interfaces utilisateur, on tire les principes généraux de développement pouvant également servir à des applications du même genre.

Abstract

This memoir is about the role of a database in the field of network management and more particularly databases gathering general informations on the network more than technical ones.

In the first part, it is analysed what the information used in network management should be. Afterwards, we describe the way two standards of network management (i.e. SNMP and OSI's CMIP-CMIS) define managemement information and which kind of network management, as defined before, they support.

The second part is an in-depth study of a real network management database application called LANDB developped at CERN (European Organization for Nuclear Research) in Geneva. That analysis is based on the work performed during a training period at CERN.
After a brief description of network environment at CERN, a quick analysis is made of a first application which led to LANDB application.
After a description of the main goals of LANDB, we analyse the structure of the database and user interfaces. We also give the general user interface development principles which can be used when building such applications.

Remerciements

Avant de nous plonger plus avant dans ce travail, je tiens à exprimer mes plus vifs remerciements à toutes les personnes qui m'ont aidé de près ou de loin à la réalisation de ce mémoire.

Je tiens à remercier plus particulièrement:

Monsieur Philippe van Bastelaer qui m'a donné la chance de pouvoir effectuer un stage et un mémoire dans un domaine qui m'intéresse au plus haut point depuis que je l'ai découvert lors de mes études ainsi que pour le temps qu'il a consacré et les conseils qu'il m'a prodigués lors la rédaction de ce travail;

Monsieur Olivier Francis Martin du CERN qui m'a permis de travailler à ses côtés pendant six mois au projet LANDB, période pendant laquelle il s'est montré particulièrement attentif et riche en conseils de toutes sortes;

toutes les personnes que j'ai rencontrées au CERN et qui m'ont aidé à remplir la tâche qui était la mienne par leurs avis judicieux;

ainsi que toutes celles et ceux que j'aurais oubliés et qui ont pourtant supporté mes sautes d'humeur durant la rédaction de ce mémoire.

Merci.

François Briard
Août 1993

Table des matières

Introduction 1

1ère partie:
la base de données dans la gestion de réseaux 3

Chapitre 1: Gestion de réseaux 4

1.1. Evolution des réseaux et intérêt de leur gestion 4
1.2. Délimitation du domaine de la gestion de réseaux 6
1.3. Les acteurs de la gestion de réseaux 11

Chapitre 2: Analyse de deux standards: SNMP et CMIP-CMIS de OSI 13

2.1. SNMP 13
2.2. CMIP-CMIS de OSI 24

Conclusion de la première partie 27

2ème partie:
Etude d'un cas concret: la gestion de réseaux au CERN 28

Chapitre 1: Le cas particulier du CERN 29

1.1. Le CERN: histoire et place dans le monde de la recherche scientifique 29
1.2. Les moyens informatiques 30
1.3. Problèmes posés lors de la gestion des réseaux locaux du CERN 32
1.4. Vers une gestion générale des réseaux 34
1.5. Conclusion 34

Chapitre 2: l'application CSNET 35

2.1. Objectifs 35
2.2. Structure de la base de données de CSNET 38
2.3. Qualités de CSNET 47
2.4. Faiblesses de CSNET 47

Chapitre 3: LANDB: présentation générale 49

3.1. Présentation du projet LANDB 49
3.2. Objectifs de l'application LANDB 51

Chapitre 4: Outils utilisés et architecture 56

4.1. Outils utilisés 56
4.2. Architecture de l'application LANDB 60

Chapitre 5: Analyse de la base de données 65

5.1. Evolution par rapport à CSNET 65
5.2. Schéma de la base de données 65
5.3. Remarques concernant le schéma 65
5.4. Tables particulières 67
5.5. Description des tables 72
5.6. Conclusion 97

Chapitre 6: Analyse des interfaces 98

6.1. Présentation générale des interfaces 98
6.2. Présentation générale des interfaces de gestion de l 'adressage 104
6.3. Modularité 107
6.4. Sécurité et contrôle 114
6.5. Aide à l'utilisateur 123
6.6. Conclusion 126

Conclusion 128

Conclusion générale 128
Critique de la démarche adoptée 130
Propositions pour la suite 131

Bibliographie 132

Ouvrages 132
Articles 133

Annexes

Annexe 1: Schéma Oracle*Case de la base de données LANDB

Annexe 2: Manuel d'utilisation des interfaces de gestion des adresses

Annexe 3: Evaluation des interfaces de gestion des adresses auprès des utilisateurs

Introduction

L'utilisation de réseaux informatiques est maintenant de plus en plus courante de par le monde. Ce phénomène touche toutes les classes d'utilisateurs informatiques et cela dans tous les secteurs de l'économie.

Alors qu'ils servent principalement à véhiculer de l'information, on remarque que paradoxalement, on manque souvent d'informations sur les réseaux eux-mêmes. Or, comme pour n'importe quel outil, les gestionnaires tant économiques que techniques ont besoin d'un maximum d'informations sur ce nouveau moyen de transmission afin de le gérer au mieux, d'autant plus qu'il est techniquement complexe.

Pendant des années, des standards de communication associés à un fabricant et non compatibles entre eux ont vu le jour et, avec eux, des outils de gestion permettant d'analyser certaines caractéristiques de ces réseaux dits propriétaires. Ces outils étaient relativement simples étant donné l'unité technique et la taille relativement restreinte de ces réseaux. Mais, depuis quelques années, on assiste à une volonté d'intégrer les différents standards afin de leur permettre de coexister pour obtenir des réseaux multi-standards et multi-vendeurs. Les réseaux actuels se caractérisent donc par une taille et une complexité beaucoup plus importante.

Cette situation a mis à jour le manque crucial d'outils de gestion multi-standards capables de prendre en compte la taille et les facettes multiples des réseaux actuels.

De plus, la plupart des outils de gestion de réseaux existant sur le marché permettent principalement la gestion de l'état des réseaux en temps réel à savoir, la gestion du trafic, des pannes et autres événements pouvant survenir sur les réseaux. Or, il existe toute une série d'informations n'étant pas directement liées au fonctionnement en temps réel et qui sont stockées sur des bases de données: informations sur les machines du réseau, sur les utilisateurs, etc.
Ces informations sont également nécessaires aux gestionnaires de réseau et demandent autant d'attention que les informations de gestion de réseau au sens où on l'entend classiquement.

Une étude de Andersen Consulting citée dans [Pin91] signale d'ailleurs "qu'alors que les utilisateurs sont généralement satisfaits de la qualité du réseau, ils émettent des critiques en ce qui concerne les services" et que "le succès dans le domaine des télécommunications dans les années nonante se fera plus grâce aux services qu'aux fonctionnalités."

Il est donc capital pour le gestionnaire de réseaux de pouvoir accéder à un maximum d'information afin de pouvoir rendre des services aux utilisateurs.

Dans la première partie de ce mémoire, nous allons tout d'abord essayer de définir ce que peuvent être les informations de gestion d'un réseau. Nous tenterons également de définir

deux types de gestion de réseaux: la gestion temps réel ou on-line et la gestion off-line, à laquelle nous nous intéresserons par la suite.

Nous étudierons ensuite comment ces informations peuvent être définies dans deux standards de gestion de réseaux pour systèmes ouverts: SNMP (Internet) et CMIP-CMIS de OSI.

Dans la deuxième partie, nous tenterons d'étudier un cas concret. Il s'agit de l'application LANDB permettant la gestion de la base de données contenant des informations sur les réseaux du CERN (Organisation Européenne pour la Recherche Nucléaire). Cette analyse est basée sur un stage de fin d'études effectué de août 1992 à février 1993 au siège du CERN à Genève.

Après avoir brossé un portrait technique du CERN en ce qui concerne les réseaux, nous analyserons brièvement la première application de gestion de base de données d'informations sur les réseaux au CERN: CSNET.

Ensuite, une étude détaillée de l'application actuelle, LANDB, permettra de présenter la solution adoptée au CERN en insistant plus particulièrement sur la structure de la base de données et sur le fonctionnement des interfaces utilisateurs.

Lors de ces deux analyses, nous tenterons de mettre en lumière les principes qui ont régi le développement de cette application afin de prendre en compte le caractère hautement hétérogène des réseaux et de la population d'utilisateurs au CERN.

1ère partie

La base de données dans la gestion de réseaux

Chapitre 1
Gestion de réseaux

La question de la gestion des réseaux est actuellement indissociable de l'idée même de réseaux. Cela est dû à l'évolution technique qu'ont connue les réseaux ces dernières années et à la complexification qui en a découlé.
Ce chapitre va essayer de mettre en lumière les raisons qui appellent une gestion des réseaux en décrivant l'évolution qu'ils ont suivi ces dernières années.
On traitera ensuite du problème de la délimitation du domaine de la gestion de réseaux et des personnes concernées par cette gestion en mettant l'accent sur la définition des données manipulées par la gestion de réseau.
On terminera enfin par une analyse succincte des données gérées par deux standards de gestion de réseaux existants et de leurs limites: SNMP et CMIP-CMIS.

1.1. Evolution des réseaux et intérêt de leur gestion

Par gestion des réseaux, on entend la collection, la maintenance et l'utilisation d'informations les concernant afin de faciliter leur création et leur exploitation.
Marshall T. Rose [Rose91] définit trois périodes qui marquent des changements successifs dans la structure des réseaux informatiques qui correspondent à trois idées que l'on peut se faire de la gestion des réseaux.
Jusqu'au milieu des années 70, les télécommunications se limitaient surtout à échanger des données entre un "mainframe" et plusieurs terminaux. On ne pouvait alors pas encore parler de "gestion de réseaux" au sens où on l'entend maintenant.
Apparurent alors, fin des années 70, les réseaux dits "propriétaires" c'est-à-dire composés de plusieurs ordinateurs issus d'un seul et même fabricant, régis par un protocole unique et véhiculés par un médium unique. Il était relativement aisé de savoir comment et à qui envoyer des données sur le réseau étant donné leur unité technique et leur taille relativement petite. La gestion de réseaux était donc relativement restreinte, car elle ne portait que sur un nombre limité d'équipements fonctionnant tous de manière similaire. Une liste des machines composant le réseau et des utilisateurs y ayant accès permettait de répondre à la plupart des questions qui pouvaient se poser au gestionnaire du réseau, le plus souvent un spécialiste. C'est donc plus en taille qu'en complexité qu'évolua la gestion des réseaux.
Au début des années 80, par contre, différents réseaux propriétaires commencèrent à s'interconnecter pour d'évidentes raisons économiques et technologiques. On vit alors apparaître les réseaux dits "multi-vendeurs" et "multi-protocoles". Ces réseaux qui deviennent de plus en plus fréquents (même chez les constructeurs informatiques!) ont, par leur caractère plus hétérogène, largement complexifié leur gestion.

Diversité des matériels

Les réseaux rassemblent actuellement des machines (ordinateurs, terminaux, répéteurs, gateways...) fabriquées par des constructeurs différents. Cela a été rendu possible grâce à des standards de communication et d'interfaçage permettant leur interconnexion. La demande de la part des utilisateurs à avoir accès à différents types de matériel (ordinateurs de calcul, ordinateurs de gestion...) a également poussé les concepteurs de réseaux à les rendre "multi-vendeurs". Ces différentes machines ne fonctionnent pas toutes selon les mêmes "principes" (système d'exploitation, codage des données, type d'affichage, périphériques utilisés...). Il est donc dorénavant utile de connaître la marque et le type de chaque machine composant un réseau afin d'étudier les différentes possibilités techniques d'interconnexion (utilisation d'une imprimante d'un type donné par plusieurs ordinateurs...).

Diversité des protocoles

Le nombre de protocoles de communication qui existent à l'heure actuelle sur le marché des réseaux est bien représentatif de la difficulté de standardisation dans ce domaine. Les standards de communication de haut niveau: TCP/IP, DECnet, Domain, AppleTalk... et les standards de niveaux inférieurs (aspects plus physiques): Ethernet, Token Ring, FDDI... coexistent avec parfois beaucoup de difficultés.
Le standard ISO (Interconnexion des Systèmes Ouverts) tente bien de définir une "philosophie" générale et, même s'il commence à être de plus en plus suivi, il ne fait pas encore l'unanimité.
Il est désormais nécessaire de savoir avec précision quel protocole de communication est utilisé avec quelle machine ainsi que de gérer toutes les informations liées à ce protocole (adresses, domaines...).
Il est aussi utile de connaître les différents protocoles de bas niveau utilisés et leur topologie afin de prévoir les possibilités et les impossibilités techniques de communication point à point.

Diversité des données véhiculées

Alors qu'auparavant les réseaux informatiques se bornaient à transmettre du texte et des valeurs numériques, la demande actuelle est beaucoup plus complexe. Son, image, vidéo viennent s'ajouter en posant parfois de réels problèmes en demandant un temps de réponse quasi immédiat (temps réel).
Là encore, la topologie du réseau doit être connue parfaitement ainsi que les capacités de chaque segment qui le compose afin de déterminer quel utilisateur pourra bénéficier de quelle application et donc de quel type de données (applications multimedia, de dessins, de traitement de textes...).
Afin d'assurer des temps de réponse assez courts pour le temps réel, il est fondamental de connaître les performances du réseau et, donc, d'effectuer des tests assez fréquents.

Diversité des utilisateurs

L'évolution des réseaux durant ces dernières années a également ouverts ceux-ci à un plus grand nombre d'utilisateurs. Ils ne sont plus l'apanage des informaticiens (programmeurs, ingénieurs...) et des scientifiques, mais sont désormais utilisés par des utilisateurs n'ayant pas de "formation technique informatique" (administration, secrétariat, décideurs...).
Il est utile, dans le cadre de la gestion des réseaux, de connaître ces différents utilisateurs, leurs fonctions et leurs connaissances afin de les aider au mieux dans l'utilisation de cet outil complexe qu'est le réseau informatique.
Ces utilisateurs sont, en outre, de plus en plus impliqués dans la gestion (administrative, financière...) de ces mêmes réseaux. Cette remarque sera développée au point suivant.

1.2. Délimitation du domaine de la gestion de réseaux

Un des problèmes majeurs de la gestion des réseaux est de définir ce que cette gestion doit englober. On a, en effet, vu que le nombre d'informations touchant de près ou de loin les réseaux et qu'il est intéressant de gérer est vaste. Il y a, là, un choix à effectuer entre une gestion purement technique (machines composant le réseau, informations relatives aux protocoles...) ou intégrant en plus des aspects plus économiques ou humains (utilisateurs, informations administratives et financières...). Analysons deux manières de classifier la gestion de réseaux.

1.2.1. La classification OSI

L'Organisation de Standardisation Internationale (OSI) a défini de son côté 5 domaines de gestion de réseaux [ISO7498-4] dans son standard ISO (Interconnexion des Systèmes Ouverts). Cette classification semble plus s'axer sur les différentes fonctions qu'implique la gestion d'un réseau que sur la nature des données elle-même.

Gestion de configuration (configuration management)

La gestion de configuration reprend la gestion de l'aspect logique et physique du réseau et la gestion des noms. Elle regroupe les informations concernant la localisation des éléments du réseau, leur nom, la topologie du réseau, son état...

Gestion des performances (performance management)

La gestion des performances comprend la collection de données statistiques (taux d'erreurs, flux de données...) à propos du réseau et de son histoire ainsi que la production de rapports basés sur ces mêmes statistiques afin d'aider le gestionnaire de réseau dans des actes décisionnels à court et à long terme.

Gestion des erreurs (fault management)

La gestion des erreurs est l'activité de détection, d'isolement et de correction des problèmes sur le réseau. Cette activité requiert un temps de réponse rapide et est fréquemment effectuée par les opérateurs et les ingénieurs (cfr. section 1.3.).
La gestion des erreurs comprend également la tenue d'un historique des erreurs déjà survenues afin de pouvoir se servir de l'expérience passée.

Gestion financière (accounting management)

La gestion financière permet d'identifier les coûts et les charges de l'utilisation du réseau. Elle permet, outre de facturer aux utilisateurs l'emploi qu'ils font du réseau (à partir de statistiques d'utilisation), de vérifier l'utilisation qui est faite du réseau autant pour le gestionnaire que pour l'utilisateur.

Gestion de la sécurité (security management)

La gestion de sécurité permet d'éviter tout accès illégal ou dangereux à tout ou partie du réseau ainsi qu'aux informations de gestion elles-mêmes.

1.2.2. Une classification orientée sur la nature des données de gestion

Voici maintenant une classification faite sur la base de l'expérience qui sera décrite dans la deuxième partie concernant la gestion des réseaux du CERN. Cette classification se base plus sur la nature des informations manipulées que sur les fonctions qu'elles permettent d'effectuer.

Gestion matérielle

Un réseau est avant tout composé de machines et d'équipements techniques divers. Il est dès lors évident que la gestion va devoir maintenir toute l'information associée à ces équipements avec, entre autres:

- Nom donné à la machine
- Marque et type de machine
- Système d'exploitation
- Localisation
- Spécifications techniques diverses

Ce type d'informations est géré depuis qu'il existe des réseaux informatiques puisqu'elles touchent à l'essence même des réseaux. La question ne se pose donc pas de savoir s'il faut ou non les intégrer à la gestion des réseaux.

Gestion protocolaire

Chaque protocole de communication demande aussi la gestion de différentes informations dont:

- Adresse de chaque machine du réseau pour ce protocole
- Nom de chaque machine du réseau pour ce protocole

L'apparition de réseaux multi-vendeurs a également amené les réseaux multi-protocoles. En effet, un réseau propriétaire était fréquemment associé à un protocole de communication (parfois associés à des architectures développées par le fabricant du matériel lui-même: DECnet, SNA...). L'interconnexion de ces différents réseaux propriétaires rend encore plus indispensable le maintien d'informations concernant les protocoles de communication coexistant sur le réseau.

Gestion logicielle

Un réseau n'a d'intérêt qu'en fonction des logiciels qui utilisent ses possibilités. Il est également intéressant de connaître les logiciels installés sur les différentes machines composant le réseau, à savoir:

- Le nom des logiciels
- Leur version
- Système d'exploitation sous lesquels tournent les logiciels

Les systèmes d'exploitation constituent une classe toute particulière des logiciels fonctionnant sur les machines du réseau. Ils interviennent dans plusieurs aspects de la gestion des réseaux.

Gestion d'exploitation

Un réseau "existe" un certain laps de temps avec des modifications, des erreurs... dans sa conception et dans son utilisation. Il peut être utile de garder une trace de ces événements afin d'aider à expliquer des problèmes techniques, d'aider à une gestion à long terme... Il peut s'avérer également utile de mettre en relation les données de gestion du réseau avec la réalité à un moment donné (monitoring). La gestion d'exploitation a, pour cela, besoin des types d'information suivants:

- Opérations de modification du réseau (description, date, responsable...)
- Erreurs signalées sur le réseau (description, date, responsable...)
- Données concernant la structure réelle du réseau à un moment donné

Gestion administrative

Un réseau n'est qu'un outil utilisé par une organisation ayant une structure et régie selon certaines règles. Il est intéressant de spécifier où chaque élément du réseau se situe dans cette organisation.
Les informations administratives d'un réseau comprennent entre autres:

- La division administrative responsable d'une machine
- Le compte de dépenses et de budget concernant le matériel
- Les licences d'exploitation

Gestion humaine

La dernière composante essentielle, pour ne pas dire une des principales, d'un réseau sont les personnes l'utilisant, le gérant et le concevant. Il est des problèmes qui ne conçoivent pas de solution sans un contact humain et il est donc impératif de maintenir, entre autres, des informations sur:

- les utilisateurs
- les responsables techniques
- les responsables de l'exploitation
- les responsables administratifs
- les fournisseurs

1.2.3. Définition d'un gestion *off-line* du réseau

Lorsque l'on se réfère à la littérature concernant la gestion de réseau, on entend parler de gestion d'erreurs, de gestion de trafic, de gestion de pannes...(cfr. [Ree90]) L'impression qu'on en retire laisse penser que la gestion de réseau n'est qu'affaire de gestion en temps réel, de gestion *on-line*.
Or comme nous venons de le voir, toute une série d'informations concernant le réseau peuvent être recueillies autrement que par des outils fonctionnant en temps réel sur le réseau. D'ailleurs, ces mêmes outils utilisent beaucoup de ces informations souvent stockées dans des bases de données.
David P. Reed le rappelle très justement dans [Ree90]: "La gestion événementielle, la gestion de trafic et l'évaluation de réseaux sont parmi les applications grandes demanderesses de moyens au niveau des bases de données".
Comme les données se trouvant sur ces bases ne peuvent pas toutes être recueillies par des programmes automatiques fonctionnant en temps réel sur le réseau, il faut veiller à définir des applications permettant de les collecter de manière *off-line*, c'est-à-dire de manière différée, auprès des différents acteurs de la gestion de réseau.
Ce que nous appelons ici *gestion off-line de réseaux* est plus souvent appelé *gestion de parc machines,* car il s'agit de gérer des données sur les machines, données qui existent, qu'elles soient branchées sur un réseau ou pas. Mais à partir du moment où les données recueillies concernent des machines branchées sur le réseau et où ces informations sont utilisées par les outils classiques de gestion du réseau, il ne semble pas trop hasardeux de qualifier la gestion de telles données de gestion de réseau à part entière.
Néanmoins, pour différencier les niveaux de gestion, nous parlerons de gestion *on-line* du réseau lorsqu'il s'agit d'applications se référant au fonctionnement en temps réel du réseau (gestion du trafic, des pannes...) et de gestion *off-line* du réseau lorsqu'on veut faire allusion aux applications gérant des données concernant le réseau, mais sur son état à un moment donné plus que durant une période donnée.

Bien que ces deux niveaux de gestion manipulent des types de données différentes, il est nécessaire qu'ils s'en échangent périodiquement. Ainsi, une application de gestion de pannes pourra utiliser les données sur le personnel d'une application de gestion off-line afin de

connaître l'e-mail du responsable d'une machine. De même, une application de gestion off-line pourra périodiquement faire appel à une application de détection de machine sur le réseau pour vérifier l'existence de données rentrées par un utilisateur.
Nous verrons, dans la deuxième partie de ce mémoire, comment l'application de gestion off-line du CERN échange des données avec les applications de gestion on-line.

La deuxième partie de ce chapitre consistera en l'analyse d'une application de gestion off-line de réseau, raison pour laquelle nous tenions à la situer par rapport à ce qu'on entend communément dans la littérature par gestion de réseau.

1.2.4. Conclusion

Il est certain qu'il n'y a pas de solution unique quant à la délimitation du domaine de gestion des réseaux: il n'est pas toujours possible d'obtenir toutes les informations voulues, certaines d'entre elles sont obsolètes pour certains types de réseaux ou d'organisations...
Il faut choisir, parmi celles qui sont disponibles, les informations auxquelles on donne la priorité, celles qui paraissent pertinentes, celles qui semblent utilisables.
Une chose est certaine: on ne peut, dans la gestion de réseau, tenir compte que de l'aspect matériel et technique. Il faut prendre en compte le contexte global dans lequel le réseau est utilisé.
Ainsi, à côté de ce qu'on a appelé la gestion on-line, il ne faut pas délaisser l'aspect off-line de la gestion de réseau, à savoir la collecte et la gestion de toutes les données auxquelles les applications automatiques ne peuvent avoir accès.

1.3. Les acteurs de la gestion de réseaux

Comme nous venons de le voir, la gestion de réseaux demande de maintenir un nombre important d'informations de tous types et récoltées à de nombreuses sources. Au-delà d'une certaine complexité, il n'est pas pensable de confier la gestion d'un réseau à une seule personne: la taille des informations et leurs spécificités demanderaient des connaissances trop vastes.
Il est donc tout à fait normal de penser à diviser la fonction du gestionnaire de réseau en confiant aux personnes les plus concernées la gestion d'une partie des informations.
De même, les gestionnaires de réseaux ne sont pas les seules personnes à être intéressées par les informations de gestion. Ces informations sont, en effet, un outil pour nombre de personnes dont les décideurs, les administrations, les utilisateurs, etc.
Analysons les différents groupes d'acteurs concernés par la gestion des réseaux. Nous nous basons pour cela sur la classification faite par D. Heyvaert [Hey91].

Décideurs politiques

Les décideurs politiques déterminent quelle sera l'utilisation finale et l'évolution du réseau, les budgets alloués et toute autre décision concernant l'utilisation du réseau dans la politique globale de l'organisation.
Ils interviennent principalement lors de la décision de créer un réseau ou lors de tout changement fondamental le concernant.
Les informations de gestion dont ils ont besoin sont principalement des informations sur les performances, les services offerts, leur qualité, etc. Elles couvrent une longue période.

Administrateur de réseau

L'administrateur de réseau est responsable du bon fonctionnement général du réseau et de la planification des moyens matériels et humains.
Il a besoin d'informations générales (topologie, performances, services offerts) aussi bien que détaillées (zones protocolaires, utilisateurs, comptabilité).

Opérateurs

Les opérateurs sont responsables du contact avec les utilisateurs finaux en cas de problème, de modifications et d'évolution. Ils sont également souvent en rapport avec les entreprises fournissant le matériel.
Ils sont le point de rencontre entre les différents acteurs d'un réseau informatique.
Ils ont besoin d'informations concernant les utilisateurs, les responsables, les protocoles, la topologie du réseau, etc. Ce sont probablement les personnes qui ont besoin de l'information la plus complète et le plus rapidement. Ils doivent, en effet, pouvoir répondre rapidement à tout problème qui leur est posé.

Ingénieurs réseau

Les ingénieurs réseau prennent en charge la maintenance des équipements du réseau, la localisation physique des problèmes, leur réparation et tout aspect directement lié à l'aspect technique du réseau.
Ils ont donc besoin d'informations concernant l'aspect physique du réseau: ses performances, sa topologie, les différentes machines qui le composent, etc.

Responsables administratifs

Les responsables administratifs désirent savoir qui a la charge d'un matériel, qui peut les tenir au courant sur son usure, son remplacement éventuel, son coût et toute information liée à la gestion économique de ce capital.

Utilisateurs

Les utilisateurs d'un réseau peuvent, quant à eux, avoir besoin d'obtenir des informations sur les services qui leur sont fournis, les personnes à contacter en cas de problèmes, la façon d'accéder à une autre machine, etc.

En conclusion

Les personnes concernées par la gestion des réseaux informatiques sont nombreuses et issues de milieux divers.
La gestion assurée par une personne unique aura tendance à disparaître de plus en plus avec la complexification et l'interconnectibilité des réseaux. Il faut dès lors penser à décentraliser la fonction de gestion des réseaux tout en lui gardant un minimum de cohérence.
La figure 1.1. illustre les différents intervenants de la gestion de réseau et les données manipulées par ces derniers.

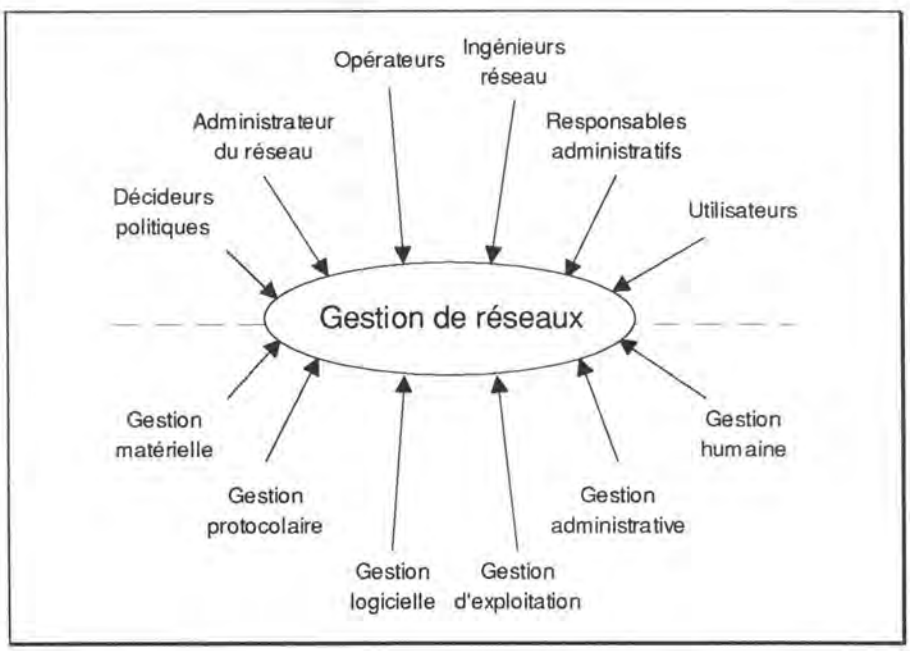

Fig. 1.1.: La coopération de divers intervenants dans la gestion des réseaux

Chapitre 2

Analyse de deux standards: SNMP et CMIP-CMIS de OSI

Il existe déjà plusieurs standards décrivant des structures de données et des procédures permettant la gestion des réseaux. Nous analyserons dans cette section les informations gérées par deux d'entre eux: SNMP puis CMIP-CMIS de OSI. Nous ne nous attaderons donc pas sur la manière dont ces deux standards gèrent effectivement des données pour nous limiter à la structuration qui est faite des données.

2.1. SNMP

Le protocole SNMP (signifiant Simple Network Management Protocol) est le protocole de gestion de réseaux pour la suite de protocoles de communication TCP/IP.

On considère que le réseau est composé de "noeuds gérés" sur lesquels tournent des "agents" pouvant envoyer et recevoir des informations de gestion à au moins une station de gestion où tournent les applications de gestion proprement dites.

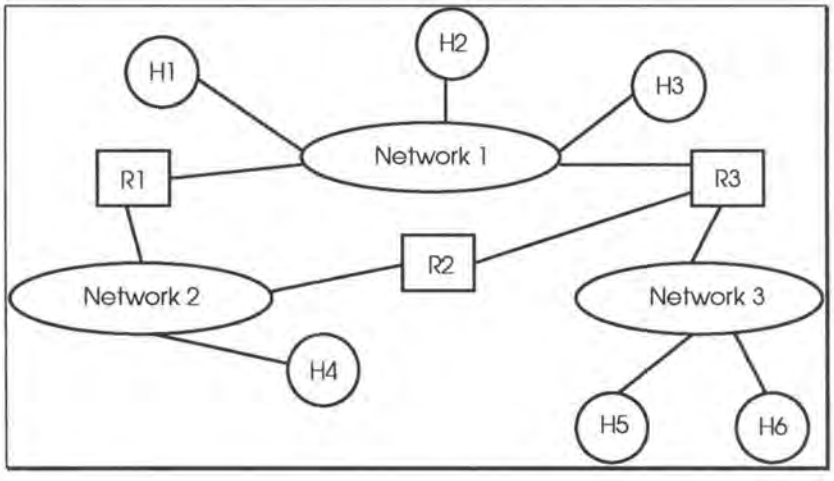

Fig 2.1. Un exemple de réseau Internet

Dans l'exemple repris par la figure 2.1., nous avons un réseau internet (basé sur la suite de protocoles TCP/IP) composé de trois sous-réseaux réliés entre eux par des routeurs (R dans le schéma) et auxquels sont attachés différentes machines ("hosts", H dans le schéma). Pour SNMP, on considère que les "hosts" et les routeurs de ce réseau sont les noeuds gérés. Un de ces noeuds, au moins, est station de gestion.

- Les informations de gestion

  La structure des informations de gestion est définie selon la *SMI* (Structure of Management Information) afin de bâtir une *MIB* (Management Information Base) contenant les informations proprement dites (et qui est donc une occurrence des règles définies dans la SMI).
  Cette classification des informations de gestion en SMI et MIB n'est pas propre à SNMP et, on le verra dans la section 2.2., elle est également présente dans le standard CMIP-CMIS de OSI.

- Le protocole de gestion

  La manière d'échanger ces informations de gestion est définie par le protocole *SNMP* (Simple Network Management Protocol).

Nous allons analyser, dans cette section, les informations contenues dans la MIB après une description des règles définies par la SMI.

2.1.1. Structure of Management Information (SMI)

La SMI décrit la structure des informations contenues par la MIB à savoir l'état dans lequel se trouvent les éléments du réseau. Comme ces éléments ne sont pas de nature semblable, les informations les décrivant seront diverses.
Pour pouvoir définir n'importe quel type d'information de la même manière, on utilise le langage de définition ASN.1 (Abstract Syntax Notation 1) défini par OSI [ISO8824][1].
La SMI, quant à elle, permettra de définir les informations de gestion selon des règles communes édictées en ASN.1.

La SMI définit la structure de la MIB selon 4 principes [Rose91]:
- la définition des objets gérés
- la structuration des objets gérés
- la syntaxe
- l'identification des occurrences

[1] Pour des raisons de place, il ne sera pas fait de description de la syntaxe ASN.1 dans ces lignes. Pour plus de détails, le lecteur peut se référer à [Rose91].

- la définition des objets gérés

Les objets présents dans la MIB sont définis selon la syntaxe ASN.1 par la macro suivante:

```
OBJECT-TYPE MACRO ::=
BEGIN
     TYPE NOTATION ::=   "SYNTAX"  type (TYPE
ObjectSyntax)
                         "ACCESS"  Access
                         "STATUS"  Status
     VALUE NOTATION ::= value (VALUE ObjectName)

     Access ::= "read-only"
                         | "read-write"
                         | "write-only"
                         | "not-accessible"
     Status ::= "mandatory"
                         | "optional"
                         | "obsolete"
END
```

où

- la clause SYNTAX définit le type de données qui modélise l'objet (type qui doit être défini dans la syntaxe ASN.1)
- la clause ACCESS détermine le niveau d'accès de l'objet considéré
- la clause STATUS définit le niveau d'implémentation requis pour l'objet
- la valeur (VALUE) contient le nom de l'objet considéré

Un exemple d'application de cette syntaxe pourrait être

```
sysDescr OBJECT-TYPE
     SYNTAX OCTET STRING
     ACCESS read-only
     STATUS mandatory
     ::= { system 1 }
```

qui décrit un objet appelé `sysDescr` de type `octet string` (suite d'octets dont la valeur va de 0 à 255), dont la valeur ne peut être que lue et que tous les noeuds gérés doivent implémenter. Le nom (valeur) de cet objet est: `system 1` car il est le premier noeud fils du sous-arbre `system`.
Nous verrons plus loin pourquoi l'objet porte deux noms.

- la structuration des objets gérés

  Dans le contexte de la gestion des réseaux, les informations manipulées sont organisées de façon hiérarchique. Elles sont dès lors reprises dans une structure d'arbre.
  L'arbre définissant la MIB est lui-même un sous-arbre d'une structure plus générale reprenant tous les objets définis par ISO et le CCITT (Comité Consultatif International pour le Télégraphe et le Téléphone). La structure d'arbre permet de déléguer des responsabilités aux différents organismes de ISO et du CCITT. Ainsi, l'organisme responsable d'un sous-arbre a la possibilité de le définir à sa guise.

  Un arbre est composé d'une racine à laquelle est rattachée une série de noeuds étiquetés. Une étiquette est un entier non-négatif associé si possible à une brève description textuelle. Ces noeuds peuvent être eux-mêmes les racines de sous-arbres de niveau inférieur.

  Pour toute la description qui suit, le lecteur peut se référer à la figure 2.2. Nous ne nous intéresserons qu'aux sous-arbres nous amenant à la description de la MIB. Nous nommerons l'arbre général "arbre de l'information de gestion" traduction de "Management Information Tree" dans [Emb90].

  Dans le sous-arbre iso(1), on retrouve:

  - standard(0) contient toutes les informations sur les standards internationaux (FTAM, par exemple, est le standard international 8571, il est donc désigné par le chemin {1.0.8571} ou {iso(1) standard(0) 8571}). Ainsi, tous les noms relatifs au standard FTAM devront commencer par le préfixe 1.0.8571.
  - registration-authority(1) est réservé pour les autorités d'enregistrements d'OSI
  - member-body(2) est la racine d'un sous-arbre composé d'un noeud par membre d'OSI
  - identified-organization(3) est la racine d'un sous-arbre composé d'un noeud par organisation qu'ISO désire favoriser

    Le noeud-fils n°6 de org(3) (nom court de identified-organization(3)) est assigné au département de la défense (dod(6)) qui a lui-même déterminé que son noeud-fils n°1 était assigné à Internet. Ce noeud est lui-même la racine d'un sous-arbre contenant les sous-arbres directory(1), mgmt(2), experimental(3) et private(4). Les sous-arbres mgmt(2) et private(4) ne sont composés, à l'heure actuelle, que d'un seul noeud chacun: respectivement mib et enterprises.

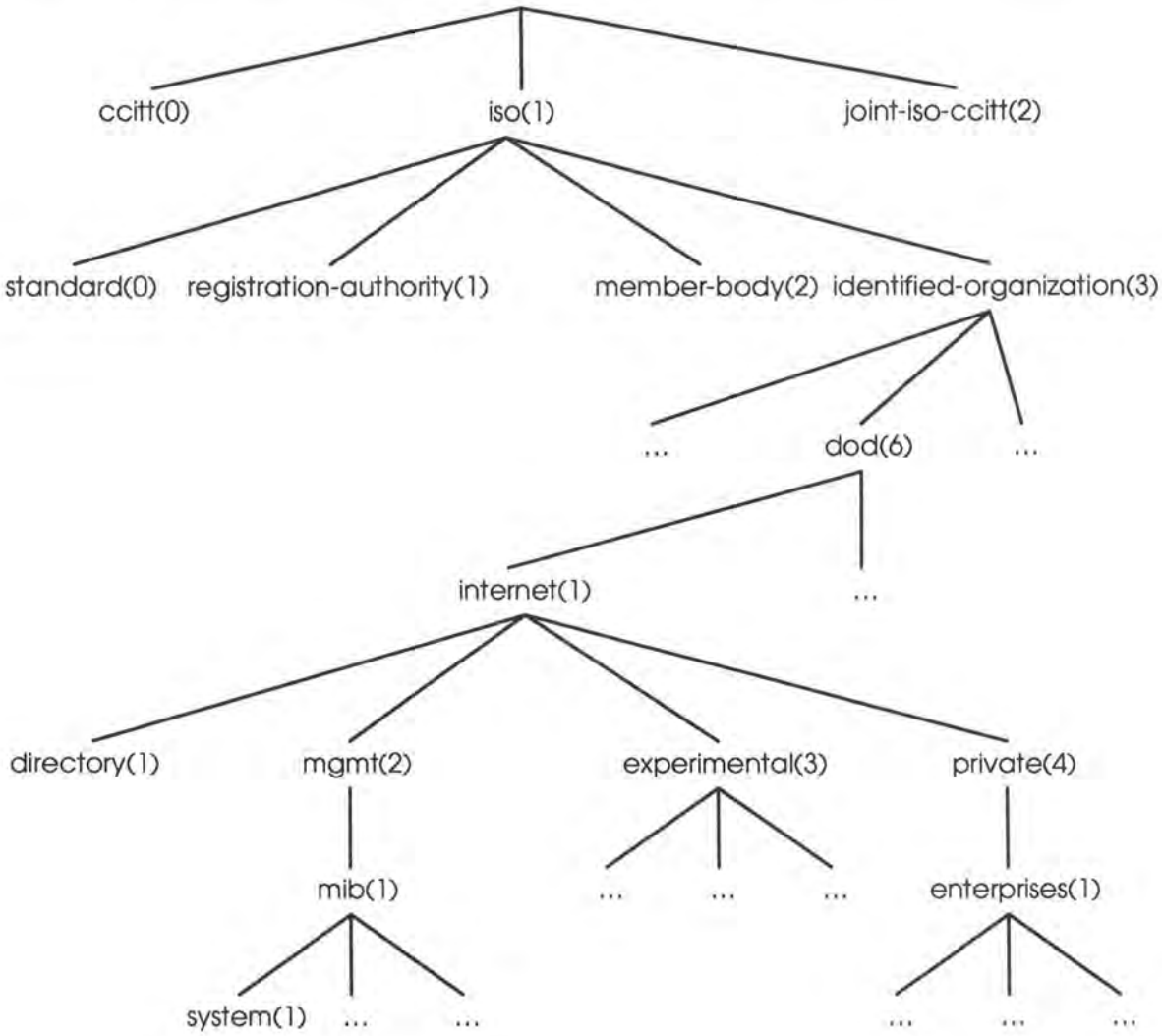

fig. 2.2.: la structuration des noms

- mib(2.1) contient les objets de la MIB du standard Internet
- experimental(3) contient les objets de MIB expérimentales
- enterprises(4.1) contient les objets des MIB d'entreprises privées[2]

Il est bon de remarquer que la structure des noms est extensible à l'infini comme il n'est pas donné de limite quand à la largeur et à la profondeur de l'arbre. Dès lors, il paraît intéressant, voire obligatoire, de ne pas réassigner un noeud déjà utilisé.
Avec cette structuration, l'objet `sysDescr` donné en exemple au point précédent peut être identifié sans erreur par:

```
{iso(1) org(3) dod(6) internet(1) mgmt(2) mib(1) 1}
```

[2] Le CERN dont il sera question dans la 2ème partie s'est vu attribuer le noeud n° 96.

ou, plus simplement,

```
1.3.6.1.2.1.1
```

Cette suite de nombre est un exemple de ce qu'on appelle un "object identifier".

On peut remarquer que tout élément de l'arbre de l'information de gestion porte deux noms comme nous l'avons vu avec l'exemple de l'objet `sysDescr`, plus haut.

En fait, à partir d'un noeud donné de l'arbre de l'information de gestion, tous les noeuds fils sont uniquement identifiés grâce à un nom relativement identifiant (RDN - Relative Distinguished Name). Par relativement, on entend que le nom identifie le noeud par rapport à son noeud père uniquement. Ainsi, des noeuds se situant à des niveaux totalement différents de l'arbre peuvent porter le même nom ce qui évite de devoir utiliser des noms extrêmement particuliers et fort peu significatifs.
Un nom relativement identifiant est composé d'un nom simple auquel est associé un nombre.
Afin de pouvoir identifier un élément de l'arbre sans équivoque et vu que le RDN ne permet pas de le faire, chaque élément de l'arbre reçoit également un nom identifiant (DN - Distinguished Name). Ce nom identifiant est composé de la suite de noms relativement identifiants (RDN) depuis la racine de l'arbre (ROOT) jusqu'à l'objet désigné.

(Voir [Emb90] pour plus d'informations à ce sujet)

- la syntaxe

La clause SYNTAX utilisée dans la définition des objets de la MIB définit le type de l'objet. Ce type doit être un type conforme à la syntaxe ASN.1. Néanmoins, ces types doivent appartenir à un des trois groupes suivants:

- les types simples (*simple types*) qui ne sont que des types primitifs de ASN.1:

```
INTEGER
OCTET STRING
OBJECT IDENTIFIER
NULL
```

- les types liés aux applications (*application-wide types*):

`IpAddress`	pour représenter une adresse IP
`NetworkAddress`	pour représenter une adresse d'une entité relative à un protocole de transfert donné (IpAddress n'est en fait qu'une occurrence du type NetworkAddress)
`Counter`	pour représenter un entier positif limité supérieurement qui ne peut être qu'incrémenté de 1
`Gauge`	pour représenter un entier positif limité supérieurement, mais pouvant être incrémenté aussi bien que décrémenté

`TimeTicks`	pour représenter un entier positif qui permet de compter le temps en centième de secondes
`Opaque`	pour n'importe quelle représentation arbitraire

- les types construits (*simple constructed types*):

```
list
table
```

- l'identification des occurrences

Les objets définis dans la SMI ne sont que des "patrons". Seules les *occurrences* de ces objets sont manipulées par les protocoles de gestion tels que SNMP. Il est donc nécessaire d'avoir, en plus de l'identificateur d'un objet (le nom de l'objet), un *identificateur d'occurrence*. C'est la responsabilité du protocole de gestion et non celle du SMI.

2.1.2. Management Information Base (MIB)

La MIB est une occurrence de la structure définie par la SMI. Nous allons nous intéresser d'un peu plus près à la MIB définie pour la suite de protocoles Internet.

- MIB-I

L'idée maîtresse des concepteurs de SNMP était qu'il fallait que le standard soit implémentable le plus rapidement possible. Le contenu de la MIB se devait donc d'être le plus simple et le plus petit possible.
Cela est motivé par ce que Marshall T. Rose appelle "l'axiome fondamental" [Rose91]:

> *Ajouter la gestion de réseau aux noeuds d'un réseau doit avoir un impact minimal et refléter le plus petit commun dénominateur*

MIB-I, la première version de la MIB [RFC1156], a donc tenté de définir un nombre minimum d'objets et ainsi uniquement ceux qui paraissaient absolument nécessaires à la gestion des erreurs et de configuration. Les objets de la MIB-I n'étaient qu'au nombre de 114 et étaient obligatoires, non dérivables les uns des autres, généraux...
Ils étaient classés en 8 groupes:

- system	3
- interfaces	22
- IP address translation (at)	3
- internet protocol (ip)	33
- internet control message protocol (icmp)	26
- transmission control protocol (tcp)	17
- user datagram protocol (udp)	4
- exterior gateway protocol (egp)	6
Nombre total d'objets de la MIB	114

Si un noeud du réseau demande l'implémentation d'un objet d'un groupe particulier, alors tous les objets du groupe en question doivent être implémentés pour ce noeud.

- MIB-II

MIB-II est la deuxième version de la MIB pour la suite de protocoles Internet. C'est une version améliorée de la MIB-I (tout en restant compatible avec celle-ci). Elle fut spécifiée à la fin de l'année 1989 dans [RFC1158]. Les modifications par rapport à la MIB-I sont l'ajout ou le retrait d'objets dans chacun des groupes et l'ajout de 2 groupes supplémentaires:

- transmission
- snmp

L'apparition de MIB-II a permis, entre autres, un support meilleur des noeuds multi-protocoles.

- Description sommaire des groupes d'objets de la MIB-II

Les groupes peuvent être représentés dans l'arbre Internet comme suit:

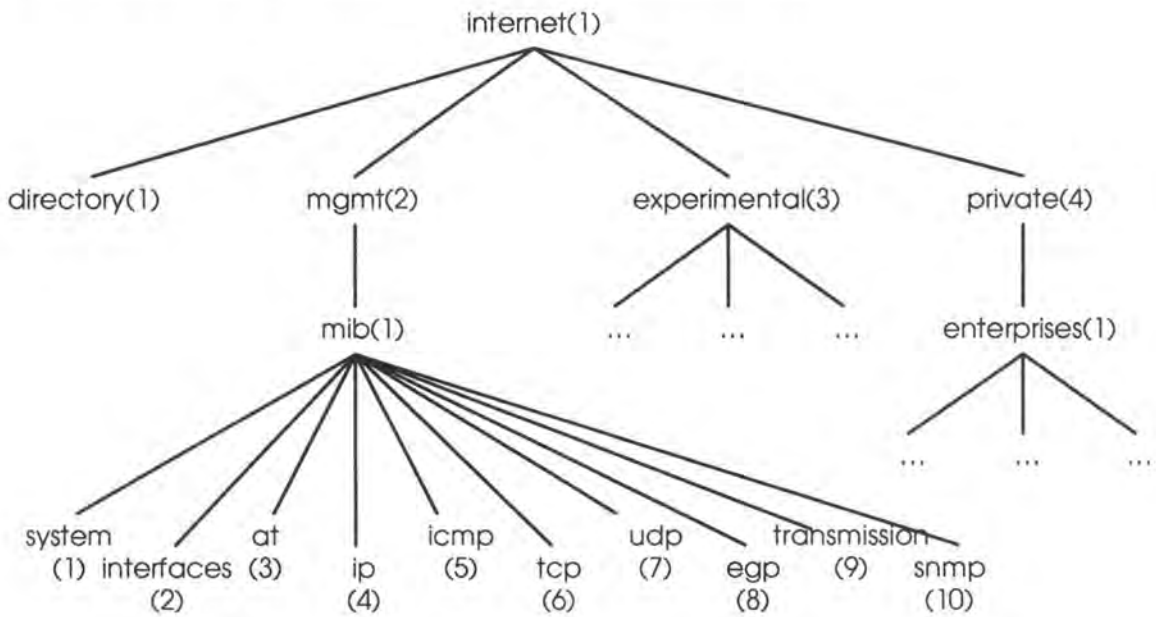

fig. 2.3.: les groupes d'objets de la MIB

- SYSTEM (1)

  Le groupe SYSTEM contient les informations sur les équipements gérés par la MIB. Il s'agit surtout d'informations de configuration: fabricant, OS... On y retrouve, entre autres:

- sysDescr	description textuelle de l'équipement
- sysContact	nom de la personne à contacter pour tout ce qui concerne cet équipement
- sysName	nom de l'équipement
- sysLocation	localisation physique de l'équipement
- sysServices	services offerts par l'équipement

- INTERFACES (2)

  Le groupe INTERFACES contient des informations concernant les entités de la couche interface.

- ifNumber	nombre d'interfaces réseau
- ifTable	table dont chacun des rangs décrit une des interfaces au moyen d'une vingtaine d'objets

- ADDRESS TRANSLATION - AT (3)

  Le groupe AT contient une table permettant d'associer une adresse protocolaire à une adresse physique. Chaque rang de la table contient trois colonnes:

- atIfIndex	numéro de l'interface
- atPhysAddress	adresse physique
- atNetAddress	adresse protocolaire

  Dans MIB-II, ce groupe n'est plus utilisé et est intégré au groupe IP (4). Il est néanmoins toujours présent dans l'arbre d'information de gestion, car on ne peut supprimer un élément de cet arbre, toute modifiaction donnant lieu à la création d'un nouvel élément.

- INTERNET PROTOCOL - IP (4)

  Le groupe IP contient des informations sur la couche IP dont:

  - nombre de datagrammes envoyés et reçus pendant une période déterminée
  - nombre de datagrammes rejetés pour tel ou tel type d'erreur
  - nombre de datagrammes envoyés/reçus de la couche supérieure
  - adresses IP associées au noeud
  - adresses physiques associées aux adresses IP (ajout de MIB-II)

  etc...

- INTERNET CONTROL MESSAGE PROTOCOL - ICMP (5)

  Le groupe ICMP rassemble 26 compteurs stockant, pour chaque message du protocole ICMP, le nombre de fois qu'il a été envoyé par l'entité locale et le nombre de fois qu'il a été généré par l'entité locale.
  Quatre de ces compteurs sont chargés de contenir le nombre total de messages ICMP reçus, envoyés, reçus en erreur ou non envoyés à cause d'une erreur.

- TRANSMISSION CONTROL PROTOCOL - TCP (6)

  Le groupe TCP rassemble les informations concernant le protocole TCP, à savoir, des informations concernant:

  - le Retransmission Timeout (algorithme utilisé, temps minimum, etc.)
  - le nombre de connexions (ouvertes, passives/actives, réétablies, etc.)
  - le nombre de segments reçus/émis/retransmis

  La MIB-II compte en plus des informations concernant:

  - le nombre de segments refusés
  - les connexions courantes (état, entité correspondante, etc.)

- USER DATAGRAM PROTOCOL - UDP (7)

  Le groupe UDP contient les informations sur le protocole UDP, c'est-à-dire:

  - le nombre de datagrammes reçus et délivrés à l'utilisateur
  etc...

  La MIB-II ajoute une table décrivant des informations portant sur les entités de la couche application utilisant le protocole UDP.

- EXTERIOR GATEWAY PROTOCOL - EGP (8)

  Le groupe EGP rassemble les informations portant sur le protocole EGP. Il s'agit, entre autres, d'informations concernant:

  - le nombre de messages EGP reçus/émis
  - le nombre de messages EGP reçus/émis avec erreur
  - une table des voisins et de leur état

  MIB-II a rajouté quelques informations à cette table.

- TRANSMISSION (9)

  Le groupe TRANSMISSION, uniquement présent dans MIB-II, permet l'introduction dans la MIB de données spécifiques à un média. Il s'agit toujours d'un groupe expérimental.

- SNMP (10)

  Le groupe SNMP, lui aussi uniquement présent dans MIB-II, permet l'introduction dans la MIB de données spécifiques à SNMP. Il s'agit également d'un groupe expérimental.

2.1.3. Conclusion sur SNMP

La SMI, de par l'utilisation de la syntaxe ASN.1 et grâce à une structure de noms qui est tout sauf restrictive, permet de décrire des données de gestion de réseau très différentes et pouvant s'adapter aux situations.

Néanmoins, si on analyse la MIB définie pour la suite de protocoles Internet et utilisant la SMI, on remarque qu'à l'exception du groupe SYSTEM(1), les informations de gestion sont uniquement axées sur les protocoles eux-mêmes. Le groupe SYSTEM(1) est le seul à contenir des informations sur les composants physiques et humains du réseau (type de machine, localisation, contact...). Le reste n'est qu'information sur l'aspect protocolaire et logique du réseau.

Cette situation est principalement due au fait que la MIB a été définie par des concepteurs de protocole qui n'avaient comme but que de rédiger un standard leur permettant d'implémenter rapidement et en manipulant un minimum de données, un système de gestion des erreurs et de la configuration du réseau. Il n'est nulle part fait mention d'une volonté de gérer autre chose.

Comme nous l'avons déjà signalé, la gestion de réseaux peut couvrir un champ beaucoup plus vaste et le standard SMI-MIB, même s'il est performant pour la gestion au niveau protocolaire (et pour la suite de protocoles Internet seulement), semble trop limité si on veut l'intégrer dans une politique plus vaste de gestion de réseaux.

2.2. CMIP-CMIS de OSI

Le protocole de gestion de réseaux CMIP et le service associé CMIS sont la réponse de ISO à la question de la gestion des réseaux. Nous avons déjà énuméré les 5 domaines de gestion définis par le standard ISO:

- gestion de configuration
- gestion des performances
- gestion des erreurs
- gestion financière
- gestion de sécurité

Le protocole CMIP et le service CMIS suivent cette classification.
Comme pour le protocole SNMP, CMIP est un protocole de transfert de données de gestion entre les éléments du réseau et, comme pour SNMP, les informations de gestion sont contenues dans une MIB (Management Information Base) définie par une SMI (Structure of Management Information). Nous allons, dans cette section, étudier brièvement le contenu de cette MIB.

2.2.1. Principes généraux

L'organisation de la MIB de CMIP n'est pas la même que celle de SNMP. La MIB de CMIP est organisée de manière hiérarchique en définissant des classes d'objets et leurs relations.

- Les classes d'objets gérés

  Une classe d'objet rassemble les objets qui vérifient les mêmes propriétés. Un objet géré est donc une occurrence d'une classe d'objets déterminée. On peut, par exemple, avoir la classe d'objets "transport connection" dont chaque occurrence serait une connexion bien déterminée au niveau de la couche transport.
  Ces classes d'objets sont définies selon la syntaxe ASN.1. Elle comprennent:

  - les attributs ou propriétés de l'objet
  - les opérations que le service CMIS peut opérer sur l'objet
  - les actions que l'on peut effectuer sur l'objet
  - les événements que l'objet peut générer
  - les relations dans lesquelles intervient l'objet

- La hiérarchie d'enregistrement (*registration hierarchy*)

  Tout comme pour la MIB de SNMP, les noms des objets définis dans la MIB de CMIP-CMIS sont issus de "l'arbre d'information de gestion".

- La hiérarchie d'endiguement (*containment hierarchy*)

  Des objets peuvent contenir d'autres objets. On définit alors une relation "est contenu par" entre deux classes d'objets. Cela complique bien sûr l'identification des occurrences d'une classe d'objet par l'ajout de sous-arbres supplémentaires dans l'arbre d'enregistrement ASN.1.

- La hiérarchie d'héritage (*inheritance hierarchy*)

  Une classe d'objet peut bénéficier des propriétés d'une autre. On définit ainsi la relation "hérite des propriétés de" entre deux classes d'objets. On peut raffiner en ajoutant à la classe "héritante" des propriétés spécifiques; elle porte alors le nom de *sous-classe*, alors que la classe de niveau supérieur est appelée la *super-classe*.

  Cette notion d'héritage est fortement inspirée des concepts de la programmation et de la conception orienté-objet comme le signale d'ailleurs [Emb90].

2.2.2. Classes d'objets gérés

Les classes d'objets gérés étant associées aux opérations que l'on peut effectuer sur elles suivent la classification ISO de la gestion de réseaux:

- gestion de configuration

Informations concernant le nom des éléments du réseau, leur localisation...

- gestion des performances

Informations concernant le taux de transfert, le nombre de paquets reçus/émis...

- gestion des erreurs

Informations concernant le nombre et le type d'erreurs survenues...

- gestion financière

Informations concernant la quantité de données reçues/émises par un système à des fins de facturation...

- gestion de sécurité

Informations concernant les accès autorisés (tables...)...

2.2.3. Conclusion sur CMIP-CMIS de OSI

On peut faire pour la MIB de CMIP-CMIS les mêmes remarques que pour la MIB de SNMP. En effet, les informations manipulées sont fort axées sur le niveau technique et logique des réseaux.

Toutefois, CMIP gère en plus des aspects financiers et de sécurité que ne gère pas (ou très peu) SNMP. Cela est dû au fait qu'ISO cherche à gérer des systèmes *ouverts* et qu'aucun axiome fondamental réducteur n'est appliqué.

De plus, CMIP permet une souplesse beaucoup plus grande de par la définition "par objets" des informations de gestion, mais cela se fait au détriment d'une simplicité d'utilisation.

Conclusion de la 1ère partie

La question de la délimitation du domaine de la gestion des réseaux apparaît centrale dans un monde où les réseaux s'étendent, se complexifient et s'interconnectent.

Alors que des standards existent pour gérer l'aspect le plus technique des réseaux, à savoir l'échange de données, on ne peut plus ignorer aujourd'hui que le réseau s'inscrit dans un cadre plus vaste comme outil d'une organisation humaine et économique.

Il est dès lors utile d'étendre la notion de gestion de réseaux à tous les aspects qu'ils recouvrent et de définir des outils de gestion manipulant un maximum de données liées à l'utilisation d'un réseau tout en veillant à rester ouvert à toute évolution technique ou organisationnelle. Il ne faut donc pas rester fixé uniquement aux informations fixées par SNMP et CMIP.

Dans la deuxième partie, nous allons analyser le cas de la gestion des réseaux locaux du CERN (Organisation Européenne pour la Recherche Nucléaire) et voir comment on essaye d'y appliquer ces principes et les difficultés soulevées.

2ème partie

Etude d'un cas concret: la gestion de réseaux au CERN

Chapitre 1

Le cas particulier du CERN

Le CERN (Organisation européenne pour la recherche nucléaire) offre un exemple tout à fait intéressant d'organisation utilisant des réseaux informatiques d'une hétérogénéité et d'une complexité élevée. Il nous paraît donc intéressant d'analyser ce cas précis; c'est ce qui sera fait de manière générale dans ce chapitre et en détail dans les chapitres suivants.
Après une brève introduction historique sur la création du CERN et sa place dans le monde scientifique, nous analyserons les besoins en moyens informatiques de cette organisation avant de décrire la complexité de l'infrastructure existante.
Ce chapitre se terminera sur une analyse des problèmes auxquels il faudra être attentif lors de la conception des outils de gestion du réseau.

1.1. Le CERN: histoire et place dans le monde de la recherche scientifique

Le CERN (Organisation européenne pour la recherche nucléaire) est une organisation internationale implantée sur près de 550 hectares à cheval sur la frontière franco-suisse non loin de Genève. Elle regroupe 19 pays européens: l'Allemagne, l'Autriche, la Belgique, le Danemark, l'Espagne, la Finlande, la France, la Grèce, la Hongrie, l'Italie, la Norvège, les Pays-Bas, la Pologne, le Portugal, le Royaume-Uni, la Suède, la Suisse, la République Tchèque et la Slovaquie.
Le CERN emploie 3000 personnes venant des 19 pays membres et accueille en permanence 4000 chercheurs venant non seulement des pays membres, mais aussi du monde entier (USA, Japon, Chine, Russie, Canada, Inde...).
Le CERN a pour fonction la recherche scientifique dans le domaine de la physique des particules. Il possède pour cela le plus grand instrument scientifique du monde: le LEP (Large Electron Positron Collider), un accélérateur de particules de 27 kilomètres de circonférence et à 100 mètres de profondeur moyenne dont le coût s'élève à 1500 millions de francs suisses.
Cet appareil et quatre autres accélérateurs de particules permettent aux chercheurs de traquer les particules les plus élémentaires de la matière et les forces qui les gouvernent. Il y a actuellement 50 expériences en cours au CERN, toutes non-militaires.
Le CERN est né après la seconde guerre mondiale d'une volonté de reconstruction et de collaboration au moyen de la science et dans le but de freiner la fuite des cerveaux vers les Etats-Unis.
Cette situation exceptionnelle permet au CERN d'être au plus haut rang des laboratoires de physique des particules. Ceci s'est confirmé dans l'attribution de plusieurs prix Nobel à certains de ses chercheurs.

1.2. Les moyens informatiques

Les moyens informatiques du CERN, à l'image de l'industrie, évoluent constamment. Il y a en tout cas de la part de la division CN (Computer and Networks), qui a la charge de maintenir les moyens informatiques, une volonté de ne pas s'attacher à une solution donnée, mais de rester le plus ouvert possible (avec tous les problèmes que cela implique et que nous développerons plus tard). Les quelques chiffres données ci-dessous ne reflètent qu'une petite partie de la complexité des moyens informatiques mis en oeuvre au CERN; ils ont été communiqués par S. Jarp de la division CN.

De par sa position de leader et de pionnier, le CERN a toujours eu des besoins en moyens informatiques fort importants. Ils sont principalement motivés par le besoin de:

- gestion des laboratoires (comme dans n'importe quelle entreprise)
- contrôle des accélérateurs (160 ordinateurs sont utilisés pour diriger le LEP)
- acquisition de données (filtrage et stockage des données des accélérateurs)
- traitement des données (simulation, reconstruction, analyse...)
- communications internationales (accès, échanges de données...)

Par exemple, les quatre expériences rassemblées dans le LEP produisent chacune 7000 GBytes de données annuellement.

Les ordinateurs au CERN

Le CERN comprend un parc d'ordinateurs très varié: variété des constructeurs, variétés des puissances. Cela est dû à la grande gamme de tâches demandées au matériel informatique comme nous le verrons plus tard. On y retrouve principalement 3 types:

- Gros Ordinateurs: IBM 9000-900/VF, CRAY XMP/48, DEC VAX...
- Stations de travail: SUN Sparc Station, DEC Station...
- Micro Ordinateurs: PC compatible IBM, Apple Macintosh, NExT Station...

L'environnement logiciel au CERN

La diversité des matériels implique également une très grande diversité au niveau de l'environnement logiciel existant au CERN. On peut néanmoins les regrouper en 5 classes:

- Logiciels développés pour et par les physiciens (en FORTRAN presque exclusivement)
- Logiciels de visualisation graphique (GKS, PHIGS...)
- Logiciels utilisés par les ingénieurs (Euclid, AutoCAD, ANSYS, TOSCA...)
- Logiciels de Gestion de Base de Données (produits ORACLE en majorité)
- Logiciels de gestion et d'administration (Excel, Word, Power Point...)
- Logiciels d'application divers

Les réseaux au CERN

Une très bonne communicabilité entre les différents systèmes était également nécessaire pour qu'ils se complètent. Cela a poussé le CERN à développer très tôt des réseaux locaux, mais également des connexions vers les réseaux extérieurs au point de devenir souvent une référence en matière de réseaux hétérogènes.
Les buts principaux des réseaux du CERN sont de:

- Donner la possibilité aux physiciens d'analyser leurs données depuis leur laboratoire d'origine
- Interconnecter des centaines d'ordinateurs, 2000 PC, 2000 Macintosh, 500 VAXStations, 350 Apollos, 200 SUNs, 200 DECstations...

Tout cela doit se faire en tenant compte du fait que les protocoles de communication de haut niveau sont nombreux:

- TCP/IP
- DECnet
- SNA
- X.25
- AppleTalk
- Domain

De plus, ces réseaux utilisent différents standards de bas niveau:

- Ultranet
- FDDI
- Ethernet
- Token Ring

Les moyens physiques de liaison entre les ordinateurs sont également très divers:

- Câbles de cuivre
- Fibres optiques
- Liaisons satellites

Ceci illustre combien les réseaux locaux du CERN sont hétérogènes et complexes. Il va sans dire que la question de la gestion de tels réseaux dans une organisation de cette envergure est stratégique.

1.3. Problèmes posés lors de la gestion des réseaux locaux du CERN

Lorsque se pose la question de gérer les réseaux locaux du CERN, on se trouve face à différents grands problèmes.

Hétérogénéité matérielle

Comme nous l'avons vu précédemment, la variété des équipements et des protocoles de télécommunication au CERN est élevée. Lors de l'élaboration des procédures et des outils de gestion des réseaux locaux, il faut tenir compte de cette spécificité. Se pose alors la question d'une spécialisation plus ou moins grande.
Au point de vue des informations à maintenir, tout d'abord, faudra-t-il un outil assez souple pour décrire n'importe quelle configuration matérielle possible ou, au contraire, plusieurs outils ne se préoccupant chacun que de stocker les informations relatives à un seul type de matériel?
Au point de vue des personnes chargées de gérer le réseau ensuite, devrait-on avoir affaire à plusieurs spécialistes, s'occupant de la gestion d'un seul type de réseau local chacun, ou à des personnes plus polyvalentes, capables de comprendre le fonctionnement général de tous les réseaux locaux?
La solution visant à favoriser les spécificités a l'avantage de fournir des spécialistes prêts à répondre à des problèmes très précis, mais ne permet pas de suivre l'évolution technique avec autant de rapidité que lorsqu'on a des outils et des gens mieux formés à répondre à une situation générale.

Taille des réseaux

La taille des réseaux du CERN en empêche leur gestion par un nombre restreint de personnes. Mais, d'autre part, il faut rester attentif au fait de ne pas disperser les responsabilités dans ce domaine afin que l'information de gestion garde un minimum de cohérence. Chaque classe d'utilisateurs (et il y en a un nombre considérable au CERN: secrétaires, physiciens, ingénieurs, techniciens, informaticiens...) a sa propre vision des réseaux locaux et il faut éviter la division et la redondance des informations de gestion.

Complexité physique

Le CERN a la particularité de s'étendre, à cheval sur une frontière, sur une surface de quelques 550 hectares et, parfois, sur une profondeur de plus de 100 mètres (puits d'expérience du LEP). Il est donc évident que l'aspect topologique et physique des réseaux est assez complexe à saisir et à gérer et devra faire l'objet d'une attention toute particulière.

Hétérogénéité humaine

Il y a un grand nombre de classes d'utilisateurs des réseaux locaux au CERN: secrétaires, comptables, techniciens, ingénieurs, physiciens, informaticiens... Ils ont chacun un niveau de "culture informatique" fort différent et il faut en tenir compte si l'on désire rendre accessible à un plus grand nombre certaines données de gestion des réseaux.
De plus, chaque classe d'utilisateur considère les réseaux sous une approche particulière.
Le secteur administratif ne considère les composants des réseaux que dans une optique de coûts, de garantie...
Les ingénieurs réseau ne sont intéressés que par l'aspect technologique de ce dernier.
Les utilisateurs finaux (qui regroupent en fait la majorité des gens travaillant au CERN), par contre, n'ont d'intérêt envers les réseaux qu'en fonction des services qui leurs sont offerts.

Evolution rapide des réseaux

Comme pour tout réseau informatique, ceux du CERN se doivent de suivre l'évolution technique. Les outils de gestion des réseaux doivent donc tenir compte d'une certaine évolutivité et ne pas trop s'attacher à une image figée de la technologie. Cela est d'autant plus vrai que les réseaux sont complexes.

1.4. Vers une gestion générale des réseaux

Il paraît déjà possible d'orienter les réponses aux questions énoncées dans la section précédente.

Il semble qu'on doive s'orienter vers une solution plus générale et pas trop spécialisée, à savoir:

- des outils qui ne soient pas trop figés et qui soient applicables à différentes situations
- l'implication d'un nombre maximum d'acteurs et de leurs conceptions propres

Cela donnera un accès plus large aux informations de gestion ainsi qu'une plus grande souplesse permettant de mieux suivre l'évolution de la technologie.
On pourra également intégrer les différents points de vue qu'ont les utilisateurs vis-à-vis des réseaux. Il faudra donc ajouter à la gestion matérielle, protocolaire et d'exploitation les gestions logicielle, administrative et humaine.
Ces différentes informations pourront être récoltées auprès d'un nombre important d'acteurs. Cela implique la création d'outils de gestion assez souples et nous verrons dans le chapitre 5 en quoi cela peut poser un problème pour le cas précis du développement des interfaces hommes- machine.
Cette diversification de la gestion devra, par contre, veiller à intégrer une attention toute particulière à la qualité et la cohérence des informations de gestion du réseau. Il faudra donc penser à un recoupement des informations entre elles ainsi qu'avec la réalité.

Même si les problèmes soulevés ci-dessus ne sont pas propres au CERN et concernent bon nombre d'autres entreprises qui ont à gérer un ou plusieurs réseaux informatiques, le cas du CERN semble tout à fait particulier de par son ampleur et sa diversité à tous les niveaux.
La différence entre le CERN et une organisation de taille plus réduite se marquera principalement lors de l'analyse de solutions de gestion des réseaux. En effet, les solutions appliquées au CERN et que nous analysons dans les deux prochains chapitres pourront sembler quelque peu excessives si on veut les appliquer à une organisation plus restreinte.
Néanmoins, la taille du réseau analysé ne change en rien les problèmes organisationnels; ils existent toujours, même s'ils sont moins visibles dans une organisation plus petite. Le cas que nous analysons reste donc un exemple intéressant.

1.5. Conclusion

Le CERN offre un cas tout à fait particulier d'analyse des problèmes liés à la gestion des réseaux hétérogènes. Néanmoins, il est un bon guide pour déceler des problèmes communs à différents types d'organisations.
Il met également en valeur tout l'intérêt d'une solution de gestion plus générale visant à intégrer les différents aspects et acteurs de la gestion de réseaux.

Les deux prochains chapitres vont décrire deux applications successives de gestion des réseaux du CERN et mettre l'accent sur les difficultés rencontrées et les solutions mises en oeuvre.

Chapitre 2

L'application CSNET

Dès 1990, le groupe CN/CS (Computers and Networks / Communication Systems) du CERN utilisa l'application CSNET permettant la gestion d'une base de données centralisée concernant les aspects physiques et protocolaires des machines branchées sur le réseau. Elle fonctionna jusqu'en mars 1993, date à laquelle elle fut remplacée par l'application LANDB. Nous allons décrire l'application CSNET dans ce chapitre et mettre en lumière les avantages et les inconvénients de sa conception. Cette analyse servira de base pour l'étude de LANDB dans les chapitres suivants.

2.1. Objectifs

L'application CSNET est née de la volonté des gestionnaires de réseau du groupe CN/CS (Computers and Networks / Communication Systems) du CERN de compléter leurs outils de gestion on-line (état du réseau, taux de transfert des lignes, coupure de lignes, alarmes diverses...) par une gestion off-line centralisée du parc de machines sous leur responsabilité.

A l'époque, trois bases de données existaient déjà dans ce domaine:
- base de données sur le cablâge du réseau (1986)
- base de données sur l'adressage TCP/IP (1987)
- bases de données sur l'adressage DECnet (1987)

L'objectif principal de CSNET était d'intégrer ces trois bases de données en une seule, permettant ainsi d'accéder facilement aux diverses données concernant les machines du réseau.

C'est ainsi en 1989 que Gudrun Dalgeir fut chargée de faire le design de cette base de données et de l'application permettant aux gestionnaires des trois anciennes bases de consulter et de modifier la nouvelle. De plus, l'application CSNET devait assurer les mêmes fonctionnalités que les anciennes bases: programmes batch, mises à jour de bases de données externes (Name servers...), lien avec d'autres applications de gestion du réseau (gestion on-line...).

L'application CSNET était donc composée de 3 éléments principaux:

- une base de données
- des interfaces permettant de la consulter/modifier
- des programmes batch faisant le lien entre d'autres bases et/ou applications et la nouvelle base

La base de données

Chacune des trois bases originales avait des données communes ce qui justifiait parfaitement leur réunion. En effet, chacune des bases gérait des informations générales sur les machines (nom, responsable, localisation...); de plus, les bases protocolaires (DECnet et TCP/IP) devaient gérer des notions Ethernet (adresse Hardware), ce que faisait également la base de cablâge... Cette redondance d'information était source d'erreurs et de conflits.

La nouvelle base ainsi créée devait supprimer cette redondance et donc gérer les 4 classes de données suivantes:

- données concernant une machine (la "boîte" proprement dite): type, localisation, responsable, utilisateur... (ces données étant issues des trois bases originelles)
- données concernant le cablâge (composants Ethernet du réseau: bridge, routeurs, gateways...)
- données concernant l'adressage DECnet
- données concernant l'adressage TCP/IP

La base de données CSNET fut implémentée sous le SGBD ORACLE, produit choisi par le CERN pour toutes ses bases de données importantes. Nous détaillerons le design de cette base au point 2.2.

Les interfaces

CSNET fut agrémentée d'interfaces rédigées en SQL*forms 2.3, produit multi-plateformes développé par ORACLE. Quatre interfaces furent développées, chacune correspondant à une fonction précise de l'application. Ces interfaces étaient employées par seulement quatre utilisateurs. Les fonctions assurées étaient:

- superviseur TCP/IP

  La fonction du superviseur TCP/IP est d'assurer la gestion du parc d'adresses à un niveau général c'est-à-dire définir des zones d'adresses, attribuer ces zones à un groupe de machines particulières...

- gestion des adresses TCP/IP

  Cette fonction consiste principalement en l'attribution des adresses TCP/IP aux machines du réseau. Le gestionnaire TCP/IP reçoit la responsabilité d'une ou de plusieurs zones TCP/IP définies par le superviseur TCP/IP.

- gestion des adresses DECnet

  Cette fonction est similaire à la gestion des adresses TCP/IP, c'est l'attribution des adresses DECnet aux machines du réseau. Il n'y a pas, par contre, de définition de zones d'adresses.

- gestion du cablâge Ethernet

  La gestion du cablâge concerne la gestion de la topologie du réseau, des éléments Ethernet (bridges, gateways,...)

Chacune de ces interfaces était composée d'un seul écran reprenant les informations nécessaires. Ces écrans étaient très spécifiques à la tâche à accomplir et toute modification, tant au niveau de la structure de données qu'au niveau des fonctionnalités des utilisateurs, impliquait un travail important de refaçonnage des interfaces.

Les programmes batch

Divers programmes batch furent écrits. Parmi eux, on retrouvait les programmes batch associées aux anciennes bases de données et d'autres permettant le lien avec les autres outils de gestion on-line du réseau.

Voici quelques exemples de programmes batch:

- Mise à jour de la base propriétaire Name Server TCP/IP à partir des données de CSNET
- Divers rapports et fichiers systèmes (par exemple, le host file Unix est généré automatiquement à partir de CSNET)
- Transfert d'informations d'autres bases de données (par exemple, la base de données NETMPROD contenant les adresses TCP/IP de la zone 128.142) vers CSNET
- etc...

2.2. Structure de la base de données de CSNET

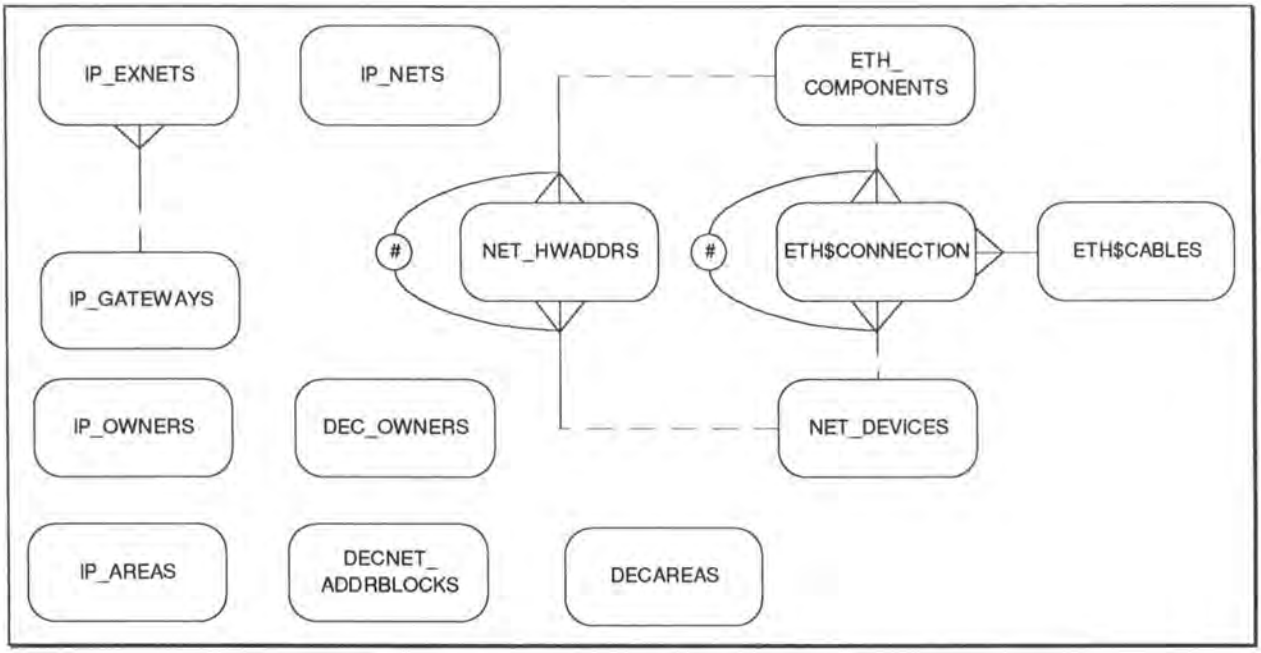

Fig. 2.1.: Structure de la base de données CSNET

La figure 2.1. reprend la structure de la base de données de CSNET. Nous avons volontairement retiré des tables spécifiques à certaines applications ou programmes batch pour nous intéresser à l'essentiel.

Le formalisme graphique utilisé est celui de Oracle*Case qui est expliqué en détail à la section 5.3. Nous y avons rajouté la convention graphique de représentation de contrainte d'exclusion issue du modèle Entités/Associations [Bod89] entre les tables ETH_COMPONENTS, NET_DEVICES et ETH$CONNECTION ainsi qu'entre ETH_COMPONENTS, NET_DEVICES et NET_HWADDRS.

La base de données est une base de données Oracle, SGBD généralement utilisé par le CERN.

Nous allons maintenant décrire en détail les différentes tables de la base de données.

DEC_OWNERS

La table DEC_OWNERS contient la liste de toutes les adresses DECnet (composées d'une area et d'un node number) ainsi que la personne en ayant la responsabilité. Ce principe de responsabilité sur un groupe d'adresses permet d'allouer à une personne donnée une série d'adresses dont elle peut gérer l'utilisation et la destination.
On remarquera que cette table est totalement redondante avec la table DECNET_ADDRBLOCKS (cfr. ci-dessous) et, alors qu'elle existait toujours dans la base de données, elle n'a jamais été utilisée!

Champs	*Description*
AreaNo	Area de l'adresse DECnet
NodeNo	Node Number de l'adresse DECnet
Owner	Responsable de l'adresse
Mod_Date	Date de la dernière modification de ce record

DECAREAS

La parc mondial des adresses DECnet est subdivisés en 63 areas de 1023 adresses (node numbers). Ainsi, chaque adresse DECnet est identifiée par le numéro de son area et de son node number.
La table DECAREAS contient pour chaque area DECnet la prochaine adresse (le prochain node number) libre. Elle sert principalement lors de l'attribution d'une adresse DECnet à une nouvelle machine sur le réseau.
On peut se poser la question de savoir si cette table est réellement utilse. En effet, une simple requête SQL (langage de requête utilisé avec Oracle) permettrait de calculer aisément la première adresse qui n'est pas associée à une machine dans la base de données pour une area connue. Il faut d'ailleurs effectuer cette requête pour remettre à jour la table DECAREAS chaque fois qu'une adresse libre est attribuée à une nouvelle machine.

Champs	*Description*
AreaNo	Numéro de l'area
NextFree	Numéro de la prochaine adresse libre
Remarks	Commentaires éventuels

DECNET_ADDRBLOCKS

La table DECNET_ADDRBLOCKS contient la liste des séries (blocks) d'adresses DECnet contiguës dont la responsabilité à été confiée à une personne ainsi que le nom de cette personne. Ce principe de responsabilité sur un groupe d'adresses était déjà utilisé pour le protocole TCP/IP (cfr. table IP_AREAS plus bas). Bien qu'intéressant, il n'a pas été implémenté réellement rendant cette table superflue.

Champs	*Description*
Block_Name	Nom du groupe d'adresses DECnet
AreaNo	Numéro de l'area des adresses
NodeNo_Low	Numéro de la première adresse du groupe
NodeNo_High	Numéro de la dernière adresse du groupe
NextFree	Prochaine adresse libre dans ce groupe
Owner	Nom du responsable du groupe d'adresses

ETH$CABLES

La table ETH$CABLES contient la liste de tous les câbles du réseau ainsi que divers renseignements sur leur type, état, longueur, etc. Plus de détails seront donnés sur ces différents renseignements dans l'analyse de LANDB (cfr. chapitre 3 à 5).
Signalons simplement que le segment logique contenu par le champ Log_Segment représente une section du câblage entre deux bridges.

Enfin, le CERN est divisé en 11 zones Ethernet, ce qui justifie la présence du champ Area permettant d'identifier à laquelle appartient le câble.

Champs	*Description*
Cab_Id	Identificateur du câble
Log_Segment	Segment logique auquel appartient le câble
Area	Zone ethernet à laquelle appartient le câble
Alias	Autre nom possible du câble
L	Longueur du câble
Status	Etat du câble
Ground	Point de mise à la terre du câble
Contact	Nom du responsable du câble
Contact_Fname	Prénom du responsable du câble
Remarks	Commentaires éventuels
Modif_Date	Date de la dernière modification de l'enregistrement
Modif_User	Utilisateur ayant effectué la dernière modification de l'enregistrement

ETH$CONNECTION

La table ETH$CONNECTION contient la liste de toutes les connections de machines (cfr. NET_DEVICES) ou d'appareils de bas niveau (cfr. ETH_COMPONENTS) avec un câble (cfr. ETH$CABLES).
Certains renseignements comme la position de la connexion sur le câble ou le type de câble sont très utiles à la gestion de la topologie du réseau.

Champs	*Description*
Cab_Id	Identificateur du câble où s'effectue la connexion
Dev_Id	Identificateur (nom) de la machine connectée (soit dans NET_DEVICES, soit dans ETH_COMPONENTS, d'où la contrainte d'exclusion dans le schéma)
Pos	Position de la connexion sur le câble
Port	
Fiber	Signale s'il s'agit d'une connexion optique
Virt_Con	
Modif_Date	Date de la dernière modification de l'enregistrement
Modif_User	Utilisateur ayant effectué la dernière modification de l'enregistrement

ETH_COMPONENTS

La table ETH_COMPONENTS contient tous les appareillages de bas niveau. Par appareillages de bas niveau, nous entendons tous les appareillages dont l'objectif est d'assurer le bon fonctionnement du réseau: bridges, repeaters, etc.

Une différenciation est faite entre ces appareillages et les machines plus orientées vers les services offerts à l'utilsateur tels que les ordinateurs, les terminaux, etc. Nous reviendrons plus tard sur cette différentiation.
Remarquons la présence du champ Alias1 pouvant contenir un autre nom de l'appareil. En effet, des utilisateurs peuvent connaître un appareil sous un autre nom (ancien nom de l'appareil par exemple). Signalons, à ce propos, la faiblesse de la conception de la base de données qui ne propose qu'un seul alias. Que faire lorsque plus de deux noms sont employés par les utilisateurs pour désigner la même machine?
Le champ Gen_Type contient le type générique de l'appareil comme par exemple "repeater" ou "multiplexer".

Champs	*Description*
Dev_Name	Nom de l'appareil
HwAddr	Adresse hardware de l'appareil
Alias1	Autre nom de l'appareil
Gen_Type	Type générique de l'appareil
Hw	Modèle d'appareil
Maker	Fabricant de l'appareil
SerialNo	Numéro de série de l'appareil
MaintC	Numéro du contrat de maintenance de l'appareil
Invent_Num	Numéro d'inventaire de l'appareil
SQE	
Log_Segment	Segment logique
Location	Localisation de l'appareil
Contact	Nom du responsable
Contact_FName	Prénom du responsable
Phone	Numéro de téléphone du responsable
Remarks	Commentaires éventuels
Mod_Date	Date de la dernière modification de l'enregistrement
Mod_By	Utilisateur ayant effectué la dernière modification de l'enregistrement

IP_AREAS

La table IP_AREAS contient les informations concernant le découpage des adresses TCP/IP gérées au CERN en zones de responsabilité. Ce principe de zones de responsabilité est totalement propriétaire et n'est défini nulle part dans le protocole TCP/IP.
Les adresses TCP/IP dont dispose le CERN sont subdivisées en zones (Areas) d'adresses contiguës dont une personne est responsable. Un responsable a le droit de disposer des adresses de sa zone afin de les attribuer à une machine ou un usage particulier.
Une adresse TCP/IP étant identifiée par 4 bytes délimitant chacun un sous-niveau dans la hiérarchie des adresses TCP/IP au niveau mondial, les zones TCP/IP sont définies au dernier sous-niveau.
Ce même principe de zones de responsabilité a été implémenté, mais sans utilisation réelle (cfr. DECNET_ADDRBLOCKS) pour le protocole DECnet.

Champs	*Description*
Area_Name	Nom de la zone
IP1	1er byte de toutes les adresses de la zone
IP2	2e byte de toutes les adresses de la zone
IP3	3e byte de toutes les adresses de la zone
IP4_Low	4e byte de la première adresse de la zone
IP4_High	4e byte de la dernière adresse de la zone
Owner	Responsable de la zone
NextFree	Prochaine adresse libre dans cette zone

IP_EXTNETS

La table IP_EXNETS contient la liste de tous les réseaux extérieurs TCP/IP auxquels le CERN est relié via un gateway. Ce réseau est défini par 3 bytes (cfr. [COM] pour plus d'informations).

Champs	*Description*
Net_Name	Nom du réseau
Gw_Name	Nom du gateway
Remarks	Commentaires éventuels
IP1	1er byte définissant le réseau
IP2	2e byte défninissant le réseau
IP3	3e byte définissant le réseau

IP_GATEWAYS

La table IP_GATEWAYS contient la liste de tous les gateways reliant le CERN à des réseaux TCP/IP externes. Un gateway reliant deux sous-réseaux, il est doté de deux adresses TCP/IP, une pour chaque sous-réseau.
La même remarque que pour la table ETH_COMPONENTS peut être faite pour les champs Alias qui sont, ici, au nombre de deux. Ce nombre peut paraître limitatif.
On remarquera que cette table est assez ressemblante à la table NET_DEVICES. La principale différence avec cette dernière est la possibilité de définir deux adresses TCP/IP.

Champs	*Description*
Gw_Name	Nom du gateway
Alias1	1er autre nom du gateway
Alias2	2e autre nom du gateway
Net1_IP1	1er byte de l'adresse pour le 1er sous-réseau
Net1_IP2	2e byte de l'adresse pour le 1er sous-réseau
Net1_IP3	3e byte de l'adresse pour le 1er sous-réseau
Net1_IP4	4e byte de l'adresse pour le 1er sous-réseau
Net2_IP1	1er byte de l'adresse pour le 2e sous-réseau
Net2_IP2	2e byte de l'adresse pour le 2e sous-réseau
Net2_IP3	3e byte de l'adresse pour le 2e sous-réseau
Net2_IP4	4e byte de l'adresse pour le 2e sous-réseau
Preference	
Hw	Type de machine
Op_Sys	Système d'exploitation du gateway
Function	
Building	Bâtiment où se trouve le gateway
Floor	Etage où se trouve le gateway
Room	Local où se trouve le gateway
Contact	Nom du responsable du gateway
Contact_FName	Prénom du responsable du gateway
Mod_Date	Date de la dernière modification de l'enregistrement
Mod_By	Utilisateur ayant effectué la dernière modification de l'enregistrement
Remarks	Commentaires éventuels

IP_NETS

La table IP_NETS contient la liste des réseaux TCP/IP du CERN (cfr [COM] pour plus de renseignements sur les réseaux).

Champs	*Description*
Net_Name	Nom du sous-réseau
IP1	1er byte définissant le réseau
IP1	2e byte défninissant le réseau
IP3	3e byte définissant le réseau
Alias1	1er autre nom du réseau
Alias2	2e autre nom du réseau
Class	Type de réseau (cfr [COM])
Mod_Date	Date de la dernière modification de l'enregistrement
Mod_By	Utilisateur ayant effectué la dernière modification de l'enregistrement

IP_OWNERS

La table IP_OWNERS contient la liste de toutes les adresses TCP/IP (composées de 4 bytes) ainsi que la personne en ayant la responsabilité.
Cette table est totalement redondante avec la table IP_AREAS (cfr. ci-dessus) tout comme l'était DEC_OWNERS.

Champs	*Description*
IP1	1er byte de l'adresse
IP2	2e byte de l'adresse
IP3	3e byte de l'adresse
IP4	4e byte de l'adresse
Owner	Responsable de l'adresse
Mod_Date	Date de la dernière modification de ce record

NET_DEVICES

La table NET_DEVICES est de loin la plus importante de la base. Elle regroupe toutes les machines de niveau supérieur (qui ne se retrouvent donc pas dans la table ETH_COMPONENTS).
Encore une fois, on peut regretter la présence d'un nombre fini de certains de ses champs: alias (4), adresse TCP/IP (1), adresse DECnet (1), Adresse hardware (2)... En effet, l'évolution technique permet actuellement des configurations très complexes (machines multi-adresses par exemple) que la rigidité de la structure de cette table ne permet pas de représenter. Le plus simple exemple est l'impossibilité de représenter des gateways TCP/IP (ayant 2 adresses TCP/IP) ce qui a entraîné la création d'une table propre (IP_GATEWAYS).
Nous verrons dans les chapitres suivants comment l'applicaion LANDB a réglé ce problème.

Champs	*Description*
Dev_Name	Nom de la machine
Alias1	1er autre nom de la machine
Alias2	2e autre nom de la machine
Alias3	3e autre nom de la machine
Alias4	4e autre nom de la machine
Domain	Numéro de réseau Domain (propriétaire)
HwAddr1	1ère adresse Hardware
HwAddr2	2ème adresse Hardware
AreaNo	Area de l'éventuelle adresse DECnet de la machine
NodeNo	Node number de l'éventuelle adresse DECnet de la machine
DecSoftAddr	Autre représentation de l'adresse DECnet (hex.)
MultiCast_Addr	
SerialNo	Numéro de série de la machine
DEC_OrderNo	Numéro de commande DEC
Hw	Modèle de la machine
Maker	Fabricant de la machine
Op_sys	Système d'exploitation de la machine
Protocols	Protocoles
TCP_Software	Software de gestion du protocole TCP/IP
External	
MaintC	Numéro du contrat de maintenance de la machine
Invent_Num	Numéro d'inventaire
SQE	
Log_Segment	Segment logique à laquelle appartient la machine
Location	Localisation de la machine
Phone	Numéro de téléphone du responsable
Contact	Nom du responsable
Contact_FName	Prénom du responsable
DEC_Contact	Nom du responsable DEC
DEC_ContactFName	Prénom du responsable DEC
DEC_Contact_EMail	Adresse e-mail du responsable DEC
DEC_Contact_Phone	Téléphone du responsable DEC
Adm_Owner	Groupe administratif propriétaire de la machine
Gen_Type	Type générique de la machine
LAT_Service	
Remarks	Commentaires éventuels
Mod_Date	Date de la dernière modification de l'enregistrement
Mod_By	Utilisateur ayant effectué la dernière modification de l'enregistrement

NET_HWADDRS

La table NET_HWADDRS contient toutes les adresses hardware existant dans les tables NET_DEVICES et ETH_COMPONENTS. Elle permet, de manière très peu esthétique, de s'assurer de l'unicité des adresses hardware. Cela est représenté à la figure ?.2. par la contrainte d'exclusion.
Ce type de structure devrait absolument disparaître. Ceci sera le cas dans LANDB comme nous le verrons.

Champs	*Description*
HwAddr	Adresse hardware
Dev_Name	Nom de la machine (de NET_DEVICES ou de ETH_COMPONENTS)
AddrType	Type d'adresse hardware (Ethernet, FDDI...)

Remarques générales sur la conception de la base de données

Comme on peut le voir, la structure de la base de données ne prend pas bien en compte l'aspect hétérogène du réseau. La création de tables réservées à un type de machines particulier (IP_GATEWAYS par exemple) ne devrait pas avoir lieu. Ainsi, certaines tables pourraient être avantageusement rassemblées en une seule moyennant quelques petites modifications dans leur structure.
Ensuite, certaines informations communes à plusieurs tables devraient peut-être faire l'objet de standardisation et de création de tables propres. Nous pensons plus particulièrement aux informations propres à la localisation de machines, aux informations sur les personnes, etc.
Enfin, la répétitivité fixe de certaines informations (adresses, alias...) ne semble pas être une bonne solution lorsque l'on connaît le caractère hautement instable des réseaux et l'apparition de configurations de plus en plus complexes.

Nous verrons plus bas, au point 2.4., quelques reproches particuliers que l'on peut faire à la structure de la base de données.

2.3. Qualités de CSNET

La simplicité du design de la base de données CSNET a, bien entendu, facilité le design des interfaces. Chaque interface était composée d'un seul écran étant donné le peu d'informations contenues par les tables de la base et la fonction unique remplie par chaque interface.

Cette simplicité a aussi entraîné une rapidité assez agréable lors de la consultation et de la modification de la base ainsi qu'une maintenabilité facilitée.

2.4. Faiblesses de CSNET

Il va sans dire que le design extrêmement simple de CSNET entraîne vite des problèmes, surtout dans un domaine aussi complexe et changeant que les réseaux et dans une organisation aussi importante que le CERN.

On peut faire les reproches suivants à la base CSNET:

Elle fait une distinction historique entre composants Ethernet (gateways, bridges...) et ordinateurs. Cette différence s'estompe de plus en plus de nos jours et ne se justifie plus. Par exemple, un composant Ethernet, tout comme un ordinateur, possède une ou plusieurs adresses protocolaires (TCP/IP...). Ce n'est que la finalité des appareils qui les différencie.

Par contre, elle ne fait pas clairement la différence entre la machine ("boîte") et ses aspects hardware (carte Ethernet) ou protocolaires (adresses TCP/IP et DECnet).
En effet, adresse TCP/IP ou DECnet se trouvaient dans la même table que la machine elle-même.
Cette différence devrait pourtant être marquée alors que l'on voit de plus en plus souvent des configurations complexes: machines multi-adresses, apparition des notions d'alias...

La structure de la base est trop rigide. Elle ne peut pas prendre en compte les configurations complexes: routers, mainframes, gateways, bridges. Par exemple, il n'est prévu d'associer qu'une seule adresse TCP/IP à chaque machine dans la base, or un gateway en a toujours au moins deux...

La sécurité n'était pas bien assurée dans la base CSNET. Beaucoup d'incohérences survenaient, entre autres à cause de l'absence de vérification des contraintes référentielles (des machines différentes pouvaient avoir la même adresse). De plus, il n'était pas prévu de back-up particulier ou d'historiques permettant de "revenir en arrière" en cas d'opération erronée.

Il y avait peu (voire aucune) vérification concernant l'orthographe de certains noms tels les noms de personnes et de fabricants. Cette situation a amené une redondance énorme: il n'était pas rare qu'une même personne ou un même type de machine se trouve plusieurs fois dans la base sous des orthographes différentes. La recherche d'une donnée existant pourtant bien dans la base ne donnait parfois aucun résultat à cause d'une différence orthographique.

Quatre interfaces (composées chacune d'un seul écran) permettaient de consulter et de modifier la base, chacune correspondant à une fonction bien précise. Chacune de ces interfaces contenait du code permettant d'accéder à la base. Ce code étant commun aux quatres interfaces, toute modification le concernant devait être effectuée quatre fois. Il va sans dire que ces interfaces étaient monolithiques et destinées à des tâches extrêmement précises et limitées. Toute modification dans les habitudes de travail impliquait une modification en profondeur des interfaces et l'ajout de nouvelles fonctions demandait l'écriture d'une toute nouvelle interface.

L'absence d'historique obligeait la mise à jour du name server NCP (TCP/IP) en temps réel. Dans la réalité, malgré une certaine rapidité, cela a posé beaucoup de problèmes et généré beaucoup d'erreurs.

Nous allons maintenant, dans les derniers chapitres, analyser l'application qui a succédé à CSNET, l'application LANDB.

Chapitre 3

LANDB: présentation générale

Au vu des faiblesses de l'application CSNET dont nous avons parlé au chapitre précédent, il a été décidé de créer une nouvelle application appelée LANDB (Local Area Network DataBase). Cette dernière, entrée en production en mars 1993, devrait permettre de pallier les défauts de CSNET et d'offrir de nouvelles fonctionnalités.
Nous allons, dans ce chapitre, présenter sommairement le projet LANDB ainsi que ses objectifs.

3.1. Présentation du projet LANDB

L'application LANDB a pour but de remplacer l'application existante CSNET dont nous avons parlé au chapitre 2. Elle doit gommer les faiblesses de CSNET tout en proposant de nouvelles fonctionnalités, dans le souci d'intégrer au maximum les diverses informations concernant le réseau.

3.1.1. Evolution par rapport à CSNET

LANDB est une nouvelle version de CSNET à plusieurs points de vue. Elle reprend tout d'abord toutes les données de l'ancienne application et remplit au moins les mêmes objectifs. De plus, elle utilise les mêmes outils (qui seront détaillés dans le chapitre 4) que CSNET, mais dans de nouvelles versions.

Là où LANDB diffère de CSNET, c'est tout d'abord dans l'apparition de nouveaux types de données:
- lignes TCP/IP externes
- informations sur les PC et les Macintoshs
- gestion des protocoles AppleTalk et Apollo/Domain
- gestions des produits installés sur les machines: description, licences d'exploitation...
- gestion des périphériques branchés aux machines

LANDB est également différente de CSNET dans les objectifs qu'elle s'est assignés. Nous les détaillerons plus bas dans la section 3.2.

D'application de groupe (CN/CS) qu'était CSNET, on est passé, avec LANDB à une application divisionnaire. En effet, un bon nombre de groupes de la division CN sont maintenant concernés par l'application LANDB:
- CN/CS
- CN/CE (applications de CAO)
- CN/SW (Software support)
- CN/CO (Computer Operators).

De plus, d'autres divisions se sont jointes aux utilisateurs de l'application:
- AS (Administrative support) pour la gestion des Macintosh et PC
- MT, SL, AS, PS et ECP pour la gestion TCP/IP et les PC
- les expériences du LEP OPAL et L3 pour la gestion TCP/IP

Des trois fonctions principales de CSNET, on est passé à un nombre de fonctionnalités beaucoup plus important dont les nouvelles sont:
- gestion d'un parc de machines SUN (CN/CE)
- gestion d'AppleTalk (non encore implémentée au moment de la rédaction de ce mémoire)
- gestion du cablâge du réseau (nouvelles informations...)
- suivi des licences (CN/SW)
- gestion du réseau externe

Enfin, LANDB s'intègre à une série d'outils de gestion on-line du réseau. Par intégration, on entend l'échange et la consultation réciproque de données. Parmi ces outils, on peut citer:
- programmes de monitoring (ARP, MOP, ELMS, AppleTalk Sniffers...): ces programmes analysent principalement le trafic sur le réseau et analysent les adresses des machines envoyant des données sur le réseau
- Trouble Ticket System: système de gestion et de suivi des pannes et avaries pouvant survenir sur le réseau
- CAW: programme graphique donnant entre autres l'état du réseau à un moment donné

Un exemple d'intégration de LANDB avec les outils de gestion du réseau est la vérification périodique du contenu de la base de données avec les résultats du monitoring du réseau (comparaison des adresses enregistrées dans la base et réellement utilisées sur les réseau...).

De nouvelles fonctionnalités sont déjà en projet dont la gestion des PC, des imprimantes, des produits et des licences à un niveau plus général. Les informations concernant ces fonctionnalités sont d'ailleurs d'ores et déjà inscrites dans la structure de la base de données.

Pour terminer, signalons que l'application LANDB compte aujourd'hui 40 utilisateurs dont la moitié l'utilise au moins une fois par semaine et une demi-douzaine tous les jours.

3.1.2. Calendrier du projet

Février-août 1990	Rapport préliminaire effectué par Olivier Martin lors de son stage
Juillet 1991	Début du développement par Olivier Martin engagé par le CERN
Juillet-octobre 1991	Tests préliminaires
Decembre 1991	Fin du design de la base de données
Mai 1992	Fin du design du code d'accès à la base de données
Juillet 1992	1ère application en production
Fin 1992	Prototype des interfaces Développements annexes: batch, rapports, émulations, etc.
22 mars 1992	Mise en production réduite
29 mars 1992	Mise en production complète

3.2. Objectifs de l'application LANDB

Globalement, les objectifs de l'application LANDB sont:

- rassembler l'information sur les machines connectées aux réseaux des divisions utilisatrices de LANDB
- améliorer la qualité des données déjà existantes dans CSNET et maintenir cette qualité
- s'adapter aux changements étant survenus et pouvant survenir dans le futur, tant au niveau des données manipulées qu'au niveau des utilisateurs

3.2.1. Rassembler l'information sur les machines connectées aux réseaux

Comme nous l'avons déjà dit, les informations auxquelles nous nous intéressons ici sont les informations de gestion off-line, c'est-à-dire des informations de type administratif, personnel, technique[1]...

Globalement, ces données peuvent être regroupées en 9 catégories:

A. les "machines" connectées au réseau[2]:
   type, nom, localisation,constructeur, modèle...
B. les utilisateurs:
   utilisateurs, responsables des machines, contacts divers...: nom, localisation, téléphone, division...
C. les données administratives
   numéro du bon de commande, gestion de la maintenance, comptabilité...
D. les applications installées:
   système d'exploitation, applications diverses, licences...
E. les périphériques
   imprimantes, mémoires de masse partagées...
F. le câblage
   câbles, connexions, type de câblage, positions des connexions...
G. l'adressage de bas niveau (hardware)
   type d'adressage hardware (Ethernet, Token Ring, FDDI), adresses hardware
H. l'adressage de haut niveau (TCP/IP, DECnet, Apollo)
   adresses, alias, noms, sous-réseaux...
I. données sur l'application elle-même
   utilisateurs, droits d'accès et de modification, programmes externes, état de l'application, pages d'aides en ligne, messages d'erreur...

Lors de la collecte de ces informations, il faut être attentif à trois types de problèmes:

---

[1] Les types de données manipulés par un tel type de gestion ont été décrits au chapitre 1. Le chapitre 5, quant à lui, décrit dans sa totalité les informations que gère l'application LANDB. Le lecteur peut s'y référer pour en avoir une idée précise.

[2] On appelle ici "machine connectée au réseau" tout appareil branché sur le réseau, qu'il soit en contact direct avec les utilisateurs (ordinateurs, terminaux...) ou que sa fonction soit plus attachée au contrôle et au fonctionnement du réseau: bridges, gateways, repeaters...

- ***centraliser*** des données provenant de plusieurs sources:

  On ne dispose pas toujours de la même quantité ni des mêmes informations pour tous les types de machines. Ainsi, on a très peu d'informations sur certains PC utilisés par l'administration. En effet, selon les gestionnaires, certains sont plus intéressés par les informations à caractère administratif (division responsable, date d'achat, numéro du contrat de maintenance...) que celles à caractère plus technique (adresse TCP/IP, software TCP/IP, Système d'exploitation installé...).
  Cela peut créer des problèmes tant au niveau du format qu'au niveau de la complétude des informations récoltées.
  De plus, il n'est pas rare de voir des informations concernant la même machine dispersées dans la base de données sous des noms différents et d'avoir ainsi une redondance non négligeable.
  Dans la réalité, les responsables de l'application sont parfois obligés de récolter des données "où ils peuvent" et il n'est pas toujours évident de contrôler la collecte des données.
  Il va donc falloir être attentif à développer une structure de base de données et des interfaces utilisateur à même de détecter ce type de problèmes.

- ***déléguer*** la maintenance des informations

  L'ensemble des données gérées par l'application LANDB est trop important (plus de 19.000 machines, 15.000 personnes...) pour pouvoir être géré par un nombre restreint d'utilisateurs comme cela se faisait sur CSNET. De plus, comme nous l'avons signalé au point précédent, il peut être nécessaire de s'adresser à plusieurs sources d'informations pour avoir une information complète sur une machine donnée.
  Il a donc été décidé de déléguer au maximum les responsabilités de collecte et de maintenance des informations de la base.
  Ceci justifie en grande partie le choix d'un gestionnaire de base de données multi-utilisateurs, à savoir, ORACLE ainsi qu'une philosophie de design des interfaces utilisateur multi-plateformes et modulaire (cfr. chapitre 6).

- assurer la ***sécurité*** de l'ensemble

  La complexité croissante de la base de données ainsi que la multiplicité des sources et la délégation de certaines responsabilités peuvent créer des situations d'erreur et de conflit. Il faut dès lors être attentif à bien gérer les droits d'accès, de modification... à la base ainsi que de détecter (voire corriger) manuellement ou, mieux encore, automatiquement, les éventuels conflits et contradictions pouvant survenir. Cela donnera lieu au développement d'un système d'historiques.

3.2.2. Améliorer la qualité des données

Un des grands reproches que l'on pouvait faire à la base de données CSNET était la pauvre qualité des données: redondance énorme, peu de vérifications de la forme des données voire aucune (orthographe, standards de représentation...), manque d'information...
Afin d'améliorer cet état de fait, l'application LANDB doit être attentive à:

- demander ***plus d'information*** lors de la connexion des machines

  Le meilleur moment pour récolter un maximum d'informations est au moment où on connecte cette machine sur le réseau.
  Néanmoins, selon la ou les fonctions de la personne encodant ces informations, il faudra peut-être se contenter d'informations touchant à un domaine particulier. On ne peut, en effet, pas demander d'informations concernant l'adressage TCP/IP à une secrétaire ignorant tout de ce protocole et seulement chargée d'encoder des données comptables ou administratives. Dans cette situation, qui est la plus fréquente, il faudra veiller à obtenir *au moins* toutes les informations dont dispose la personne. Cette contrainte doit être reflétée dans le design des interfaces utilisateur en évitant de faire des interfaces trop limitées et pas assez souples.

- ***contrôler les informations clés***

  Lors de la saisie ou de la modification d'informations, l'application CSNET laissait toute liberté aux utilisateurs quant au format ou à l'orthographe des données. De plus, elle ne vérifiait pas toujours les contraintes référentielles.
  En plus de générer de l'information redondante, cette situation créée des problèmes lorsqu'il s'agit de retrouver de l'information: la recherche sur une valeur donnée ne donne pas toujours de résultat même si l'information existe dans la base tout simplement parce que l'orthographe diffère ou que la valeur de recherche n'a pas été entrée dans le format correct.
  Cette situation n'est pas tolérable vis-à-vis de certaines données "clés": nom et prénom des personnes, nom des constructeurs, types de machines, format des adresses...
  Dès lors, les interfaces utilisateur de l'application LANDB doivent être à même de guider (voire forcer) l'utilisateur lors de l'introduction de données clé et de corriger leurs erreurs. On verra dans la section 6.4. quels mécanismes d'interface homme-machine ont été utilisés pour mener à bien cet objectif.

- disposer d'un ***feed-back*** sur les informations clés

  L'introduction des données concernant le réseau se faisant de manière off-line et des erreurs étant toujours possibles, il est probable que le contenu de la base de données ne reflète pas toujours *parfaitement* la réalité physique du réseau.
  Afin de détecter d'éventuelles erreurs, il est nécessaire que le couple base de données - interface utilisateur soit complété par une série de programmes batch analysant le réseau et corrigeant ou, au moins, signalant des erreurs ou anomalies entre la réalité et la base de données.
  Il est également nécessaire d'organiser un feed-back avec le personnel utilisant l'application.

3.2.3. S'adapter aux changements et être multifonctionnel

Les informations que gère l'application, les utilisateurs qui les manipulent, les machines concernées... sont extrêmement nombreux, différents et changeants. Dès lors, LANDB doit offrir plusieurs facettes afin de répondre aux besoins de tous les utilisateurs et être souple à tout changement pouvant survenir aussi bien au niveau des données manipulées par l'application qu'au niveau du travail des utilisateurs.
Cette multifonctionnalité et cette souplesse doivent se retrouver tant dans la structure de la base de données que dans le design des interfaces. Mais, cela, sans perdre de vue la maintenabilité de l'ensemble.

- définir une ***structure*** aussi ***flexible*** que possible

  La base de données doit être aussi souple que possible afin que tout changement n'affecte qu'une partie minime de sa structure. Ainsi, l'apparition de configurations complexes tels les multiport repeaters ayant plusieurs adresses TCP/IP ne devrait, dans le pire des cas, remettre en cause que la seule partie de la structure de la base de données concernant les adresses TCP/IP (ce qui n'était pas le cas avec CSNET). Il est donc nécessaire de définir une structure de base de données très "éclatée", ce qui a été fait, on le verra dans le chapitre 5 .
  Ainsi, si un type de donnée change fondamentalement (nouvelle version d'un protocole de communication par exemple), il suffira de modifier la seule partie de la base de données qui concerne ce type de données.

- assurer la ***modularité*** des interfaces

  Le même souci de décentralisation doit exister au niveau des interfaces utilisateurs. Il faut éviter de définir des interfaces trop spécifiques à une tâche regroupant toutes sortes d'informations différentes, comme c'était le cas dans CSNET, pour préférer une solution veillant à définir une interface modulaire. Par modulaire, on entend une philosophie de développement des interfaces qui tende à définir un maximum d'interfaces communes à plusieurs utilisateurs, attaché à un type de donnnée particulier plutôt qu'à tous les types de données associés à une tâche.
  Ainsi, si un type de données change fondamentalement, il suffira de ré-écrire l'interface le concernant sans devoir aller modifier toutes les interfaces spécifiques aux utilisateurs consultant ce type de données. De la même manière, si un utilisateur manipule un type de données en plus, il suffira de rajouter aux interfaces qu'il utilise celle permettant de gérer ce type de données supplémentaire.
  Cette modularité doit également permettre d'avoir un taux de "réutilisation" des interfaces assez élévé. En effet, dans certains cas, des utilisateurs ayant des fonctions différentes peuvent manipuler les mêmes données.
  Mais il faudra dès lors être attentif à ne pas trop disperser l'information et toujours permettre à l'utilisateur de visualiser un maximum d'informations au même moment.
  De plus, cette modularisation devra s'accompagner de mécanisme d'échange de données entre les interfaces assurant une certaine cohérence du point de vue de l'utilisateur.

- assurer la ***maintenabilité*** de l'ensemble

  Comme on l'a déjà signalé ci-avant, un effort de modularisation tant au niveau de la base de données qu'au niveau des interfaces utilisateur permettra une plus grande maintenabilité de l'application, à savoir suivre l'évolution des types de données et l'évolution du travail des utilisateurs.
  Néanmoins, une modularité excessive peut avoir l'effet inverse et il faut veiller à ne pas trop disperser l'information.
  Par exemple, l'existence d'une interface utilisateur ne servant qu'à une seule personne, car elle est la seule (et risque apparemment de le rester) à utiliser le type de données concernant cette interface, est une situation à éviter pour ne pas rendre la maintenabilité de l'application trop lourde.

Chapitre 4
Outils utilisés et architecture

Afin de respecter au mieux les objectifs fixés, les développeurs de LANDB se doivent d'adopter des outils et une achitecture compatibles avec ces derniers.
Dans ce chapitre, nous allons décrire les principes qui ont poussé les développeurs dans le choix de leurs outils et dans la conception générale de l'architecture de l'application.

4.1. Outils utilisés

Nous allons maintenant voir quels outils ont été utilisés pour développer les 3 composantes de l'application LANDB: base de données, interface utilisateur et programmes batch.

4.1.1. Base de données: Oracle version 6 et Oracle*Case

Comme nous l'avons déjà signalé, le CERN a choisi le gestionnaire de base de données relationnelles Oracle pour toutes ses bases de données de taille importante. LANDB n'échappe pas à la règle.
Outre une place prépondérante sur le marché des SGBD relationnels dans le monde des grandes organisations, Oracle offre un avantage majeur dans le cadre de l'application LANDB et un environnement tel que celui du CERN: il fonctionne selon une architecture distribuée permettant d'avoir une base de données centralisée installée sur une machine unique et plusieurs plateformes clientes pouvant être des machines de fabricants différents, fonctionnant avec des systèmes d'exploitation et des standards divers. L'architecture distribuée a l'autre avantage de rendre transparente la localisation des données à l'utilisateur [Gal89].

Pour permettre à un utilisateur d'accéder à la base de données depuis son environnement particulier, il suffit d'installer le SGBD Oracle sur ce dernier et de lui donner un accès (nom d'utilisateur, mot de passe et autorisations) à la base de données elle-même. Ainsi, il est nécessaire de ne maintenir qu'une seule version de la base sans empêcher pour autant un utilisateur d'y accéder depuis n'importe quel type d'environnement. Ceci répond parfaitement au problème d'environnement multi-plateformes si aigu au CERN.

Oracle gère également les accès concurrents à la base ce qui est absolument nécessaire dans le cadre d'une application multi-utilisateurs.

La version 6 de Oracle en utilisation au CERN propose en outre un outil CASE permettant de définir un méta-schéma de la base de données. Cet outil permet non seulement de définir de manière aisée la base et de suivre avec précision son évolution, mais également de faciliter la documentation la concernant (schéma de la base, rapports, statistiques...).

4.1.2. Interfaces utilisateur: SQL*forms 3.0

Critères de choix d'un outil de développement

Le problème de l'environnement multi-plateformes est encore plus aigu lorsqu'il s'agit des interfaces utilisateur. Deux philosophies s'offrent aux développeurs et aux responsables d'application en pareil cas:

- soit définir une interface utilisateur par type de plateforme sur laquelle tourne l'application.

L'avantage d'une telle solution est de pouvoir tirer parti de toutes les posssibilités de la plateforme visée comme, par exemple, lorsqu'on utilise une station de travail permettant l'usage de possibilités graphiques et de la souris. Dans un cas pareil, une interface de type fenêtré (WIMP - Window, Icon, Mouse, Pointer) sera développée.
De plus, l'utilisateur aura à sa disposition une interface probablement de même type que les interfaces qu'il utilise pour d'autres applications sur la même plateforme. Ce principe de ressemblance est particulièrement vrai avec des environnements graphiques (MS Windows, Finder Macintosh, X Windows...) mais reste utilisable avec des interfaces en mode texte: utilisation de touches particulières généralement employées pour le même type d'actions dans d'autres applications (la touche DO pour lancer un processus sur un clavier VAX par exemple), même "style" de présentation des informations que dans d'autres applications couramment utilisées...
Un désavantage de cette solution est le coût de développement et de maintenance d'autant d'interfaces qu'il y a de types de plateformes.
De plus, le risque est grand de voir se développer des interfaces ayant des potentialités différentes selon les plateformes (si on porte tout l'effort de développement sur un type d'interface précis au détriment des autres, par exemple).

- soit définir une interface fonctionnant sur tous les types de plateformes.

Cette solution a le grand avantage de limiter le coût de développement et de maintenance des interfaces. Il ne faut, en effet, maintenir qu'une seule interface pour toutes les plateformes.
Un autre avantage est de standardiser les interfaces et d'éviter toute différence de potentialités d'une plateforme à l'autre.
L'inconvénient majeur que l'on peut reprocher à cette solution est de devoir se satisfaire du plus petit dénominateur commun en ce qui concerne les possibilités (graphiques et interactives) des différentes plateformes. En effet, il est impensable de définir une interface WIMP sur un terminal texte.
Dès lors, l'utilisateur risque de devoir utiliser une interface ne respectant pas du tout le style des interfaces des autre applications qu'il utilise sur sa plateforme. Habitué à des interfaces graphiques, l'utilisateur étant obligé d'utiliser une interface en mode texte risque d'être déçu et de l'employer à contrecœur.

L'application CSNET avait choisi la seconde solution en utilisant un produit Oracle permettant de définir une interface commune à tous types de plateformes: SQL*forms 2.3. Ce produit avait, outre celui de réduire la charge de travail, l'avantage d'être spécifiquement destiné aux interfaces pour la consultation et la modification de bases de données.

Etant donné que l'application LANDB doit fonctionner sur environ 5 types de plateformes (Macintosh, SUN, DEC-VMS, DEC-Ultrix, IBM-VM/CMS...), que ce nombre risque d'augmenter dans les années à venir et que les interfaces doivent être extrêmement modulaires (cfr. section 6.3.) et seront donc importantes en taille, la deuxième solution a également été choisie pour LANDB.
Ce choix a été également poussé par l'utilisation et l'expérience acquise sur le produit SQL*forms avec CSNET.

*Caractéristiques générales de SQL*Forms 3.0*

SQL*forms permet de créer des interfaces pour la consultation et la modification de bases de données Oracle. Ces interfaces suivent la métaphore de la forme: elles ressemblent à un formulaire avec libellés, zones de saisie d'information, cadres... Les formes sont en mode texte de 80 caractères sur 24. L'utilisation de la souris n'est pas possible et le clavier est le seul périphérique d'entrée utilisable.
Une forme peut couvrir tout ou partie de l'écran. Lorsqu'une forme ne remplit qu'une partie de l'écran, elle peut être affichée à n'importe quel endroit de ce dernier *au-dessus* des formes déjà à l'écran et disparaît en rétablissant l'écran dans son état initial dès qu'on la quitte. Ce principe est bien connu dans les environnements graphiques de type fenêtrés (MS Windows, X-Windows, NeXT...). L'intérêt est que c'est SQL*forms qui se charge de gérer la mise à jour de l'écran et la superposition des différentes formes. On verra dans l'analyse des interfaces utilisateurs de LANDB en quoi ce principe a intéressé les développeurs.
Dans l'exemple de forme repris à la figure 4.1, les parties sombres représentent les zones de saisie de l'information.

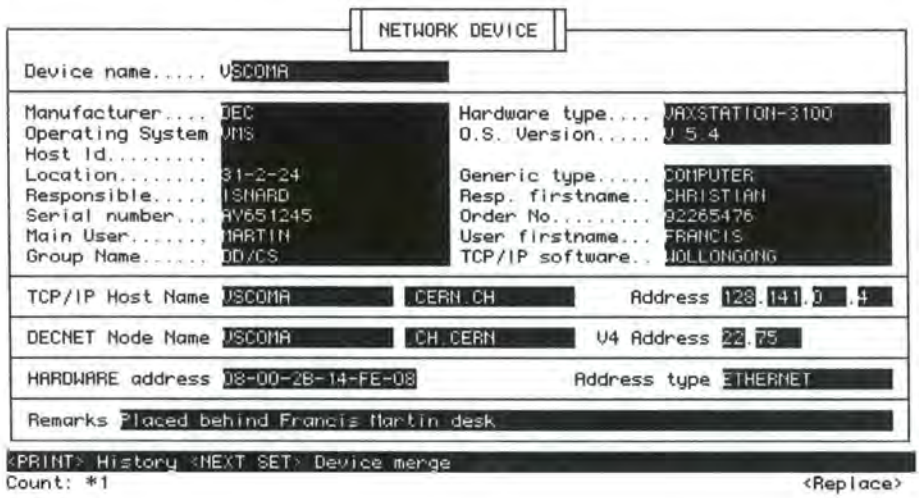

Fig. 4.1: Exemple de forme

La figure 4.2 donne un exemple de formes superposées, celle du dessus n'occupant qu'une partie de l'écran.

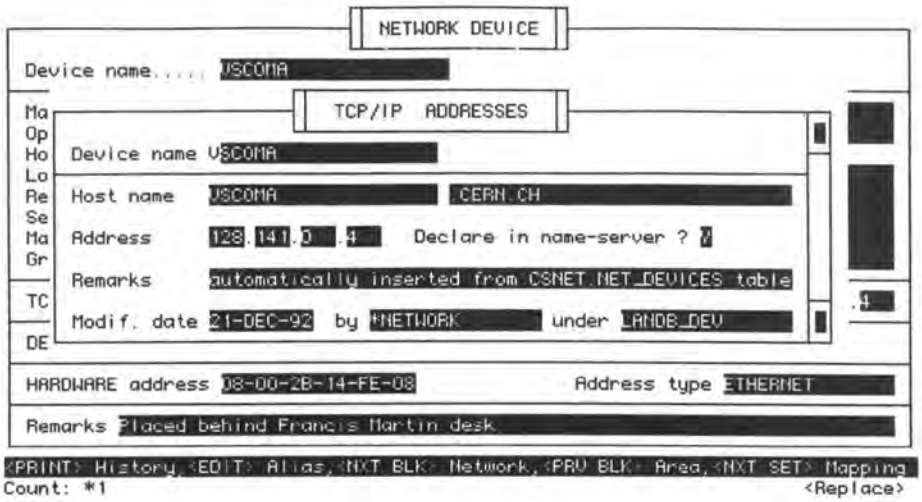

Fig. 4.2: Exemple de formes se superposant

4.1.3. Programmes batch: SQL, PL/SQL, Pro*C, Pro*Fortran...

Les différents programmes batch servent à intégrer l'application LANDB aux autres outils de gestion on-line du réseau du CERN. Cette fonction est essentielle, comme nous l'avons signalé dans la première partie.
Divers langages ont été utilisés pour développer ces batch dont le langage PL/SQL dont nous avons déjà parlé, mais également le SQL standard, Pro*C, Pro*Fortran...

Une certaine préférence est donnée au langage SQL. En effet, les programmes batch consistent souvent à utiliser les données de la base ou à les comparer avec d'autres. Il semble donc adéquat d'utiliser un langage d'accès à une base de données pour effectuer ce type de travail.
Un autre avantage de SQL sur les langages tels Pro*C et Pro*Fortran est d'être un standard et donc d'être relativement plus portable que d'autres langages.
Enfin, SQL étant un langage interprété, il n'est pas nécessaire de recompiler le code source à chaque fois qu'un système d'exploitation change de version, qu'on passe à une version supérieure d'Oracle...
Bien entendu, l'utilisation de SQL n'est possible que lorsque les données manipulées par le batch ne proviennent que de bases de données. Lorsque plusieurs sources sont utilisées, un langage conventionnel est souvent la seule solution. Néanmoins, les responsables de l'application tentent d'utiliser au maximum le langage SQL (ou PL/SQL) au dépend des langages conventionnels.

Nous ne nous étendrons pas plus sur le problème des programmes batch dans la suite de ce mémoire, ce qui mériterait un chapitre particulier. Ce choix est fait en fonction de la place qui nous est impartie dans ce mémoire et de l'état d'avancement des batch au moment de sa rédaction.

4.2. Architecture de l'application LANDB

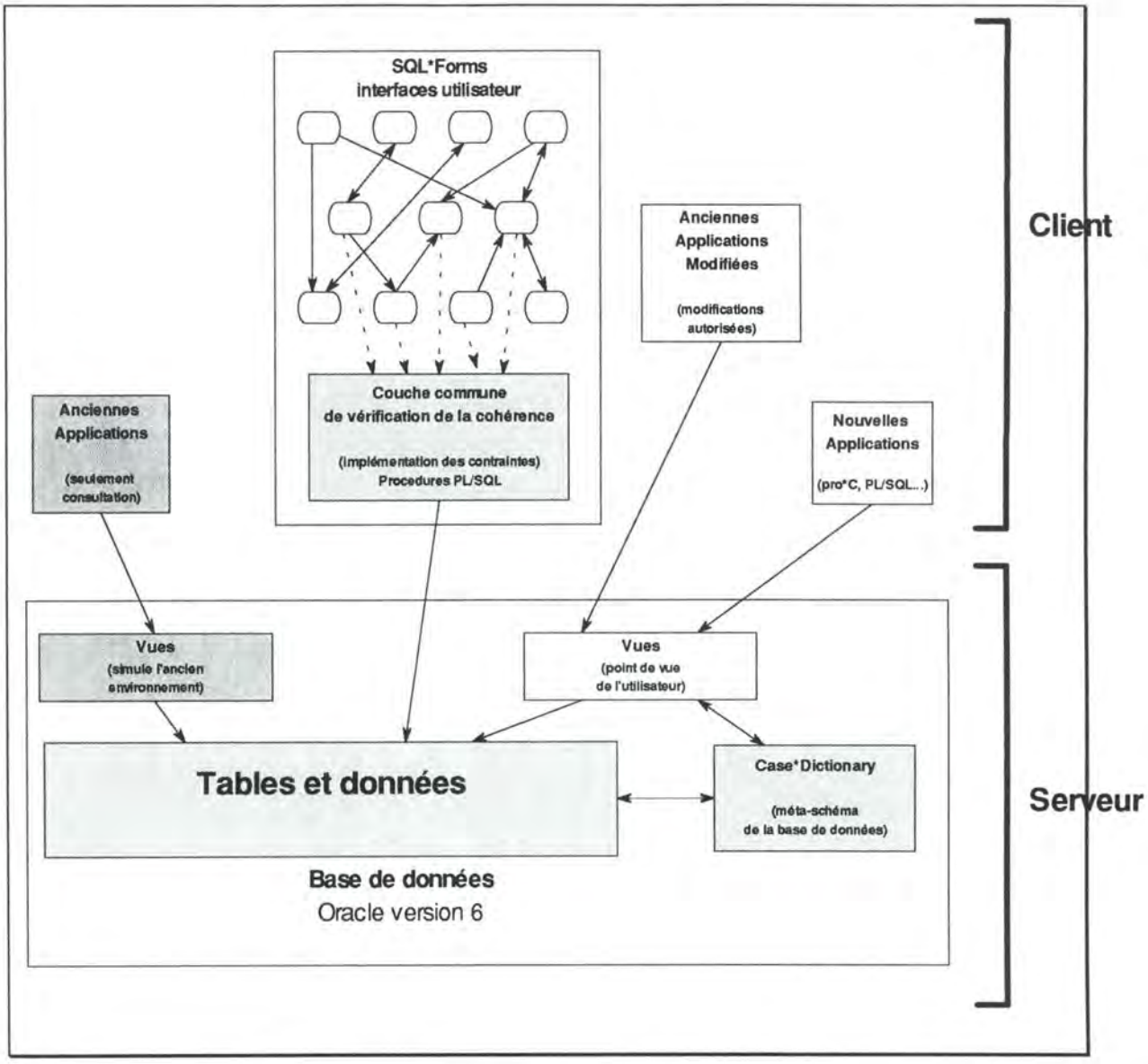

fig. 4.3.: l'architecture de l'application LANDB

L'architecture de l'application LANDB reprend à son avantage quelques caractéristiques des outils utilisés. Cette architecture est représentée à la figure 4.3.

4.2.1. La base de données

Architecture de la base de données

La base de données suit l'architecture client-serveur de Oracle et est donc accessible par différents clients (cfr.4.1.1.) depuis diverses plateformes.

LANDB est une base de données à architecture distribuée de type "Système centralisé dans un contexte réparti" [Gal89, p. 274], c'est-à-dire que les utilisateurs sont dispersés sur plusieurs sites, mais que les données sont centralisées en un site unique: il n'y a pas de répartition des données.

Selon [Gal89], le principal avantage d'un regroupement central des données sur une répartition de celles-ci est la facilité de gestion qui en découle:
- repérage aisé des données (car sur un lieu unique)
- dictionnaire de données unique et donc plus facile à gérer
- gestion centralisée des accès concurrents
- cohérence des données, de leur mise à jour et des contraintes d'intégrité

Les principaux désavantages de cette solution sont quant à eux:
- Les situations de pannes (problèmes de réseau entre clients et serveur, arrêt du serveur...) provoquent un blocage de l'accès aux données pour *tous* les utilisateurs
- Performance: tous les utilisateurs accédant aux mêmes données au même endroit, il n'est pas possible de moduler les performances d'accès
- Pas de responsabilité locale des données: il n'est pas prévu de déléguer une partie des données sur un site différent (par contre, une délégation de la *gestion* de certaines données est faite au niveau des utilisateurs, comme on le verra plus loin)

Bien que le nombre d'utilisateurs et leur dispersion géographique justifieraient la mise sur pied d'une architecture distribuée des données, étant donné la taille de la base de données LANDB et les liens étroits existant entre les différentes tables, les développeurs de LANDB ont décidé d'opter pour une architecture centralisée des données afin de ne pas encourir de gros problèmes de gestion de la base (cohérence, mise à jour...).

La figure 4.4. représente ce principe d'architecture distribuée. Comme on peut le voir, deux solutions s'offrent à l'utilisateur qui désire avoir accès à la base de données: soit installer Oracle sur sa machine et se brancher sur le serveur via le réseau (ce qui est le cas pour le Macintosh et la station de travail se trouvant à gauche sur le schéma), soit lancer une session sur une machine centrale (la machine DEC Ultrix du schéma, par exemple) sur laquelle Oracle est installé et qui accède à la base via le réseau.

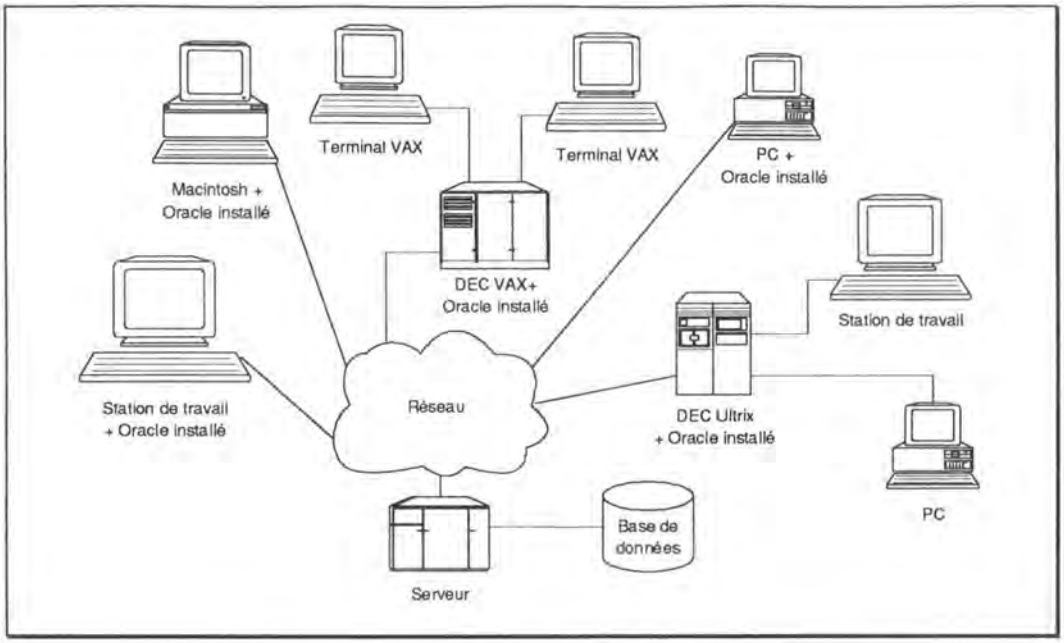

Fig. 4.4.: L'architecture d'accès à la base de données

Contenu de la base de données

La base de données comprend:

- les données elle-mêmes
- le méta-schéma

  Le méta-schéma décrit la structure de la base de données. Ceci est intéressant à des fins de maintenabilité et de documentation. Il est généré et géré par l'application Oracle*Case dont nous avons déjà parlé.
- les vues sur la base de données

  Les vues permettent de définir de nouvelles structures de données à partir de celles existantes et d'ainsi accéder aux données de manière différente. Cette fonctionnalité est intéressante en vue d'améliorer l'accès aux données, mais également en vue de permettre aux anciennes applications de fonctionner en leur donnant l'illusion d'accéder à l'ancienne base. De même, les vues permettent aux utilisateurs d'accéder aux données selon un point de vue qui leur est propre.

  Reprenons la définition faite par [Dat86, p. 12]:

  "Une vue est une table virtuelle c'est-à-dire une table qui n'existe pas "réellement", mais le fait croire à l'utilisateur. Les vues ne contiennent pas de données, mais leur définition en terme d'autres tables est stockée dans le dictionnaire de la base de données."

  La figure 4.5. reprend l'exemple classique du client et de ses commandes: une table représentant les client et une autre les commandes. Le champ Num_Client de Commande, doit, par contrainte référentielle, contenir une des valeurs possibles du champ Numero de la table Client. Cette contrainte référentielle exprime l'association de 0-N commandes à un client et d'1 et 1 seul client à une commande.
  La figure symbolise la définition d'une vue sur les tables "Client" et "Commande" afin de donner l'illusion de la table fictive "Derniere_Commande", table qui reprend

la dernière commande en date passée par tous les clients ainsi que les informations sur le client.

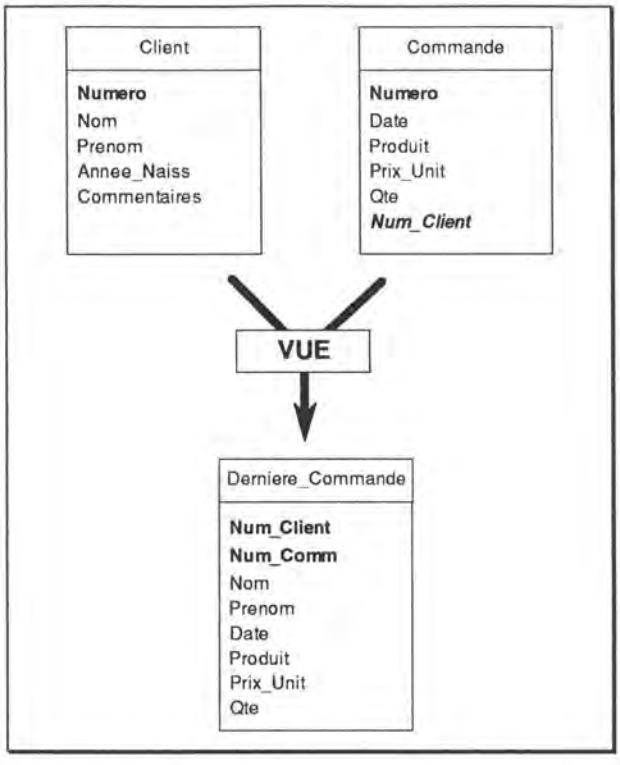

Fig. 4.5.: Un exemple de vue

4.2.2. Les interfaces utilisateur

Les interfaces utilisateur se situent au niveau du client. En effet, la version *exécutable* des interfaces est spécifique à un type de plateforme.

Les interfaces utilisateur sont composées de toute une série de formes correspondant chacune à un type d'information bien précis (machine, adresse TCP/IP, personne...) ou à une fonction précise d'un utilisateur. Ces formes s'appellent les unes les autres afin d'offrir à un utilisateur particulier l'ensemble des formes dont il pourrait avoir besoin, en fonction de sa tâche. Plus de détails seront donnés dans le chapitre 6.
La couche commune de procédures PL/SQL se trouvant entre les formes et la base de données sert à assurer la cohérence de la base. Nous en reparlerons plus en détail dans le chapitre 6.

4.2.3. Les programmes batch

Signalons seulement que d'anciens programmes batch fonctionnant à l'origine sur CSNET peuvent toujours être utilisées par LANDB grâce aux vues définies sur la nouvelle base, les anciens batch "ne se rendant pas compte" qu'ils travaillent sur une nouvelle base de données. Les anciennes applications ne pouvant fonctionner avec le système de vues ont dû être modifiées.
De même, des applications prévues pour de petites bases de données dorénavant absorbées par LANDB peuvent être éxecutées selon le même principe.

Nous allons dans le chapitre suivant décrire en détail la structure et le contenu de la base de données. Dans le dernier chapitre, nous nous attarderons sur les interfaces utilisateur. Nous ne parlerons plus des programmes batch dont l'état d'avancement étaient le moins avancés au début de la rédaction de ce mémoire et pour laisser une place plus importante à l'analyse et à l'évaluation des interfaces.

Chapitre 5

Analyse de la base de données

Nous allons, dans ce chapitre, analyser la base de données de l'application LANDB. Après avoir étudié différentes structures de données spécifiquement conçues pour atteindre certains des objectifs de l'application, nous décrirons en détail toutes les tables de la base.

5.1. Evolution par rapport à CSNET

Par rapport à l'application CSNET, la base de données de LANDB est beaucoup plus complexe: on passe d'une dizaine de tables à plus d'une cinquantaine. Ceci est la conséquence directe d'une volonté de souplesse et d'adaptabilité de la base de données face à la réalité changeante et instable des réseaux locaux.
Le contenu initial de la base de données LANDB a été récupéré de la base CSNET et hérite ainsi de la redondance et des erreurs de cette dernière. Néanmoins, des programmes batch et les interfaces utilisateur ont été développés en connaissance de cause et permettent, au fur et à mesure de l'utilisation de la base de données LANDB, de "nettoyer" cette dernière (détection de redondance, utilisation de listes de valeur corrigées périodiquement...).

5.2. Schéma de la base de données

Le schéma de la base de données peut être trouvé à l'annexe 1. Il peut être déplié de manière à être toujours visible lors de la lecture de ce chapitre.

5.3. Remarques concernant le schéma

Formalisme graphique utilisé

Le formalisme graphique utilisé pour le schéma de la base de données est celui de Oracle*Case. Chaque "boîte" représente une table et les associations définies entre elles sont symbolisées par des lignes. La figure 5.1. représente les quatres cas de figure auxquels on peut être confronté.

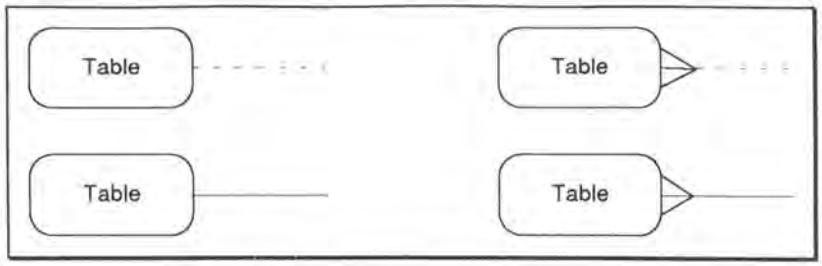

Fig. 5.1.: les 4 types de liens entre table et association

Une ligne pointillée partant d'une table représente l'aspect facultatif pour une occurrence de cette table à être associée à une occurrence de l'autre.
Une ligne pleine représente l'obligation pour toute occurrence de la table d'être associée à une occurrence de l'autre. Une "patte d'oie" symbolise, quant à elle, la possibilité pour *n* occurences de la table à laquelle est accrochée cette patte d'oie d'être associée à une occurrence de l'autre table. En particulier, la patte d'oie signale la table qui doit contenir la référence à la clé identifiante de l'autre table (clé étrangère) dans le cas d'un SGBD relationnel.
Donc, en conclusion, la ligne pointillée représente la connectivité minimale 0 pour le rôle joué par l'entité à laquelle cette ligne est attachée. La ligne continue quant à elle représente la connectivité minimale 1. La patte d'oie symbolise la connectivité maximale *n* pour le rôle jouée par l'entité se trouvant ***à l'autre extrémité*** de la ligne.
L'exemple classique Client-Commande-Produit est illustré selon ce formalisme graphique à la figure 5.2. en parallèle avec les modèles MAG [Hai86] et Entités/Associations [Bod89].

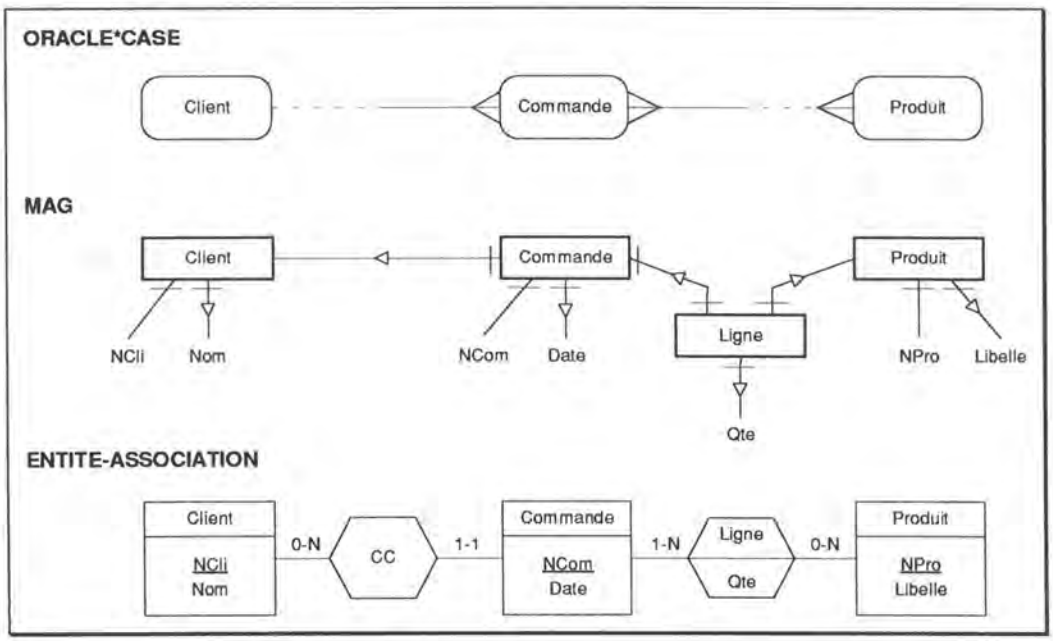

*Fig. 5.2.: exemple simple selon les modèles Oracle*Case, MAG et Entité-Association*

Limites de ce formalisme

On remarquera que le formalisme Oracle*Case se limite à représenter les différentes tables sans spécifier leurs attributs. De plus,on le voit bien dans la figure 5.2. entre les tables "Commande" et "Produit", la relation "Many-To-Many" est représentée par le formalisme

Oracle*Case alors qu'elle n'est pas implémentable dans une base de donnée relationnelle (donc avec Oracle) et devra laisser la place, selon la première règle de transformation [Hai86], à une table supplémentaire de liaison comme le représente le schéma MAG (table "Ligne").
Le formalisme d'Oracle*Case permet donc de créer un schéma conceptuel global d'aspect très général ne prenant pas en compte les spécificités relationnelles d'Oracle et n'allant pas plus bas que le niveau des tables. On peut donc plus le rapprocher du modèle Entité/Association. Malgré cette généralisation, ce formalisme semble intéressant pour réprésenter une base de données de la complexité et de la taille de LANDB (plus de 50 tables). Aller plus en détail risquerait de compliquer à outrance le schéma et sa compréhension. De plus, cette indépendance vis-à-vis d'un SGBD particulier permet d'appliquer éventuellement ce schéma à plusieurs types de SGBD.

5.4. Tables particulières

La base de données contient quelques tables dont le seul but n'est pas de représenter une réalité physique ou organisationnelle du réseau. Ces tables sont plutôt définies dans le but d'atteindre les objectifs que l'application s'est définie. Ce sont les listes de valeurs et les historiques. Nous allons les décrire en détail ainsi que les processus qui y sont associés.

5.4.1. Listes de valeur

L'un des buts de l'application LANDB par rapport à CSNET est d'améliorer la qualité des données. Pour cela, il faudra veiller à vérifier le format des données et/ou l'appartenance d'une valeur à un ensemble de valeurs autorisées. Par exemple, un système d'exploitation devra être entré en majuscules et son orthographe devra être unique: "UNIX" et non "Unix", "VM/CMS" et non "VM-CMS"
Alors que la vérification du format relève uniquement des interfaces utilisateur, il est important de s'attarder à la définition de listes de valeurs autorisées dans la base de données. Ainsi, toute une série de tables serviront à stocker les valeurs autorisées que les interfaces pourront consulter afin d'aider l'utilisateur à faire un choix correct et de vérifier ce qu'il a introduit.
Les listes de valeurs ne contenant qu'un champ (Logical_Group n'a que le champ group) verront souvent leurs valeurs recopiées dans d'autres tables telles quelles sans aucun mécanisme de contrainte référentielle. Elles servent donc de référence lors de l'encodage ou de la modification de certaines valeurs. Il existe donc un risque d'avoir une valeur incorrecte dans ces champs. Ce sera la tâche des interfaces utilisateurs de s'assurer qu'on choisit la valeur de ces champs dans la liste de valeurs mais il faudra également s'assurer de la mise à jour automatique de tout changement d'une valeur de la liste.
Par contre, les listes de valeurs plus complexes et contenant plusieurs champs (comme les personnes: nom, prénom, bureau, sémaphone, téléphone...) donneront lieu à une contrainte référentielle et verront leurs enregistrements référencés par une clé étrangère. Ainsi, une personne est référencée par son numéro identificateur dans la table "personal information" plutôt que par copie des champs la concernant. Donc, la validité de ces champs est mieux vérifiée.
Cette différence est illustrée à la figure 5.3.

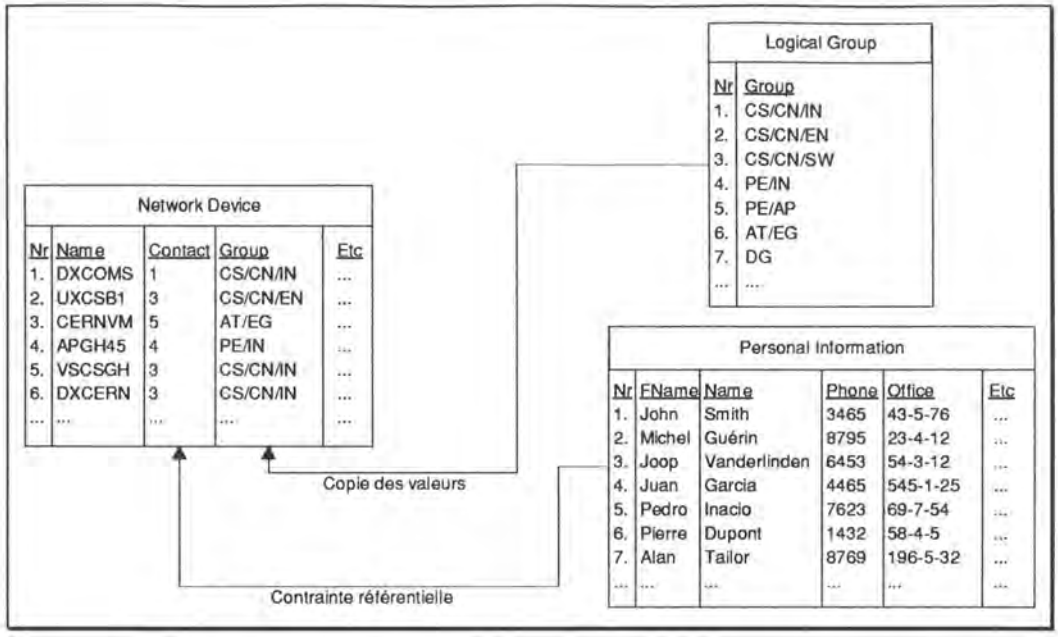

Fig. 5.3.: Exemple des deux types de références aux listes de valeurs

Parmi les listes de valeurs, on retrouve les tables:

- Operating_System
- Device_Type
- Personal_Information
- Logical_Group
- Generic_Type
- Hardware_Address_Type

Les listes de valeurs sont initialement créées à partir de toutes les valeurs contenues dans la base de données CSNET. Il y a donc une redondance assez élevée qui ne peut être éliminée qu'à la main. Ainsi, parmi plusieurs orthographes d'une même valeur, le gestionnaire de la base de données devra choisir la bonne et remplacer toutes les autres ainsi que mettre à jour les références et/ou les valeurs copiées dans les tables.
Les développeurs pensent donc que la qualité de ces listes de valeurs ira croissant au fur et à mesure de leur utilisation et de leur "nettoyage".
Un aperçu du fonctionnement des listes de valeurs au niveau des interfaces utilisateur sera fait plus loin dans la section 5.5.

5.4.2. Historiques

Certaines tables contenant des informations considérées comme importantes se voient doublées d'une table dite d'historique. L'historique permet de garder une trace de toutes les opérations modificatrices (donc pas la sélection) effectuées sur les enregistrements d'une table et d'ainsi conserver une image de tout enregistrement avant (pour la modification et la suppression) ou après (pour l'insertion) l'opération dont il est l'objet. L'historique permet en cas d'erreur de reconstruire une situation passée de la base et donc de "défaire" les erreurs.
Chaque opération faite sur un enregistrement de la table "mère" donne lieu à la création d'un enregistrement dans la table historique.

La table d'historique a donc toujours la même structure que la table "mère" à laquelle s'ajoute le champ Operation contenant le type d'opération SQL ayant donné lieu à la création d'un enregistrement d'historique: création (insert), modification (update) ou suppression (delete).
Les champs Modification_Date, Modified_By et Under_Account permettent de connaître respectivement la date, l'utilisateur et le compte Oracle sous lequel a été effectuée l'opération. Ces mêmes champs dans la table "mère" conservent ces mêmes informations en ce qui concerne l'opération dont résulte l'état actuel de l'enregistrement. Ces champs existent même dans les tables ne bénéficiant pas d'une table d'historique. On peut ainsi connaître au moins la dernière opération effectuée sur toute table ainsi que la personne qui l'a effectuée.
Le champ Modification_Date des tables d'historique fait partie de la clé primaire. Or, il n'existe pas de champ permettant de signaler l'heure à laquelle a été effectuée l'opération. Donc, l'historique ne peut conserver qu'une seule opération par jour pour une valeur donnée! Cette spécificité des historiques ne semble pas limitative au vu du nombre d'opérations généralement constaté sur la base de données CSNET et qui ne devrait pas trop changer avec LANDB.
Il n'est pas possible de supprimer des enregistrements de tables d'historique, car ces dernières agissent comme des "boîtes noires". Il faut néanmoins penser à supprimer périodiquement tous les enregistrements relatifs aux opérations antérieures à une date donnée afin d'éviter des tailles de tables historiques trop importantes.
Il faut également remarquer que les éventuelles contraintes référentielles existant dans la table "mère" n'ont plus de raison d'être dans la table d'historique. En effet, si un attribut d'un enregistrement doit référencer un enregistrement précis d'une autre table, une fois placé dans l'historique, aucune garantie n'est donnée que l'enregistrement référencé continuera d'exister ultérieurement. Ce problème est d'ailleurs la faiblesse principale du système d'historique tel qu'il a été défini ici.

Prenons un exemple simple composé de 3 tables: "Network Device" qui contient les informations sur les machines du réseau, "Address" qui contient les informations sur les adresses associées aux machines et enfin "Address History" qui est l'historique de la table "Address". On peut remarquer à la figure 5.4. que la table "Address History" contient au moins les opérations de création des adresses existant dans "Address". De plus, il existe une contrainte référentielle entre le champ "Device" de la table "Address" et le champ "Name" de la table "Network Device". En effet, toute adresse doit être associée à une machine existante.

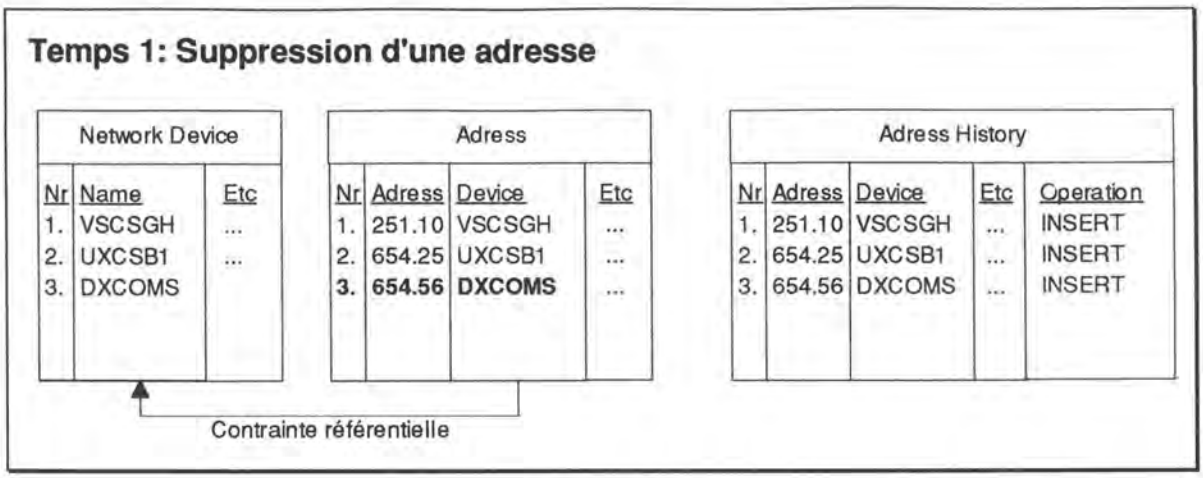

Fig. 5.4.: Etat de l'historique au temps 1

Supposons, maintenant, qu'on supprime l'adresse 654.56 associée à la machine DXCOMS. Cette opération aura pour effet de supprimer l'enregistrement de la table "Address" et d'en créer un dans la table "Address History" qui n'est qu'une copie de l'enregistrement supprimé avec le nom de l'opération effectuée: DELETE. Les tables se retrouvent donc en l'état représenté à la figure 5.5.

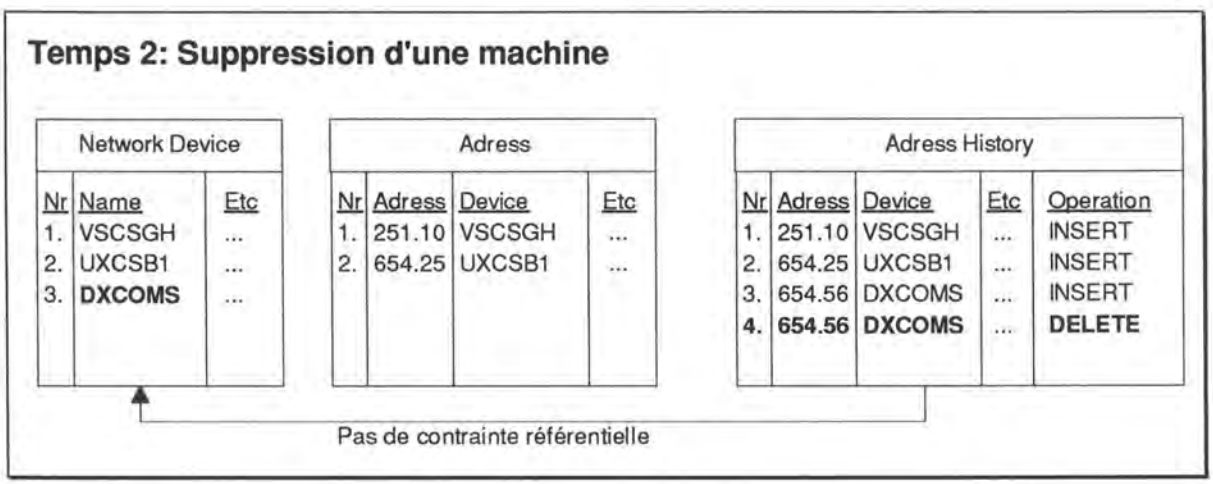

Fig. 5.5.: Etat de l'historique au temps 2

L'enregistrement d'historique est une copie de celui de "Address" que l'on vient de supprimer à la différence près qu'il n'existe plus de contrainte référentielle. En effet, imaginons que l'on supprime maintenant l'enregistrement ayant trait à la machine DXCOMS de la table "Network Device".

Temps 3: Historique irrécupérable

Network Device

Nr	Name	Etc
1.	VSCSGH	...
2.	UXCSB1	...

Adress

Nr	Adress	Device	Etc
1.	251.10	VSCSGH	...
2.	654.25	UXCSB1	...

Adress History

Nr	Adress	Device	Etc	Operation
1.	251.10	VSCSGH	...	INSERT
2.	654.25	UXCSB1	...	INSERT
3.	654.56	DXCOMS	...	INSERT
4.	654.56	DXCOMS	...	DELETE

Fig. 5.6.: Etat de l'historique au temps 3

On se retrouve ainsi en l'état décrit à la figure 5.6.
Les deux derniers enregistrements de la table "Address History" référencent alors des opérations sur une adresse d'une machine n'existant plus. Si on désire récupérer cette adresse en re-insérant l'adresse dans la table "Address", il faudra préalablement ré-insérer la machine DXCOMS dans la table "Network Device"... Les deux derniers enregistrements de "Adress History" sont donc irrécupérables directement. Ainsi, le mécanisme de récupération de données à partir de l'historique est dépendant des contraintes référentielles existant dans la table dont il est historique et de l'évolution des tables référencées.
Malgré cette faiblesse incontournable, les tables d'historique restent un très bon outil d'information lorsque l'on désire récupérer une erreur ou analyser comment elle a pu survenir.

5.5. Description des tables

Nous allons maintenant décrire chaque table de la base de données en détail. Chaque table sera d'abord décrite de manière générale puis la liste de ses champs sera donnée avec d'éventuels commentaires explicatifs. Les champs participant à la clé primaire de la table sont numérotés dans la colonne "Clé" et les références à des clés primaires d'autres tables sont spécifiées dans la colonne "Clé primaire de".

La différence fondamentale avec CSNET est l'utilisation massive du principe relationnel de la réference. Ainsi, tout type de données pouvant justifier de son importance fait l'objet d'une table propre.

APOLLO_CONNECTION

La table Apollo_Connection contient les informations concernant la connexion d'une machine du réseau à un réseau Apollo. Apollo est un protocole de niveau supérieur propriétaire et apparaît ponctuellement au CERN.

Champs	*Description*	*Clé*	*Clé primaire de*
Usual Name	Nom de la machine connectée	1	Network_Device
Network Number	Numéro du réseau auquel la machine est connectée	2	Apollo_Network
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

APOLLO_CONNECTION_HISTORY

La table Apollo_Connection_History est l'historique de la table Apollo_Connection.

APOLLO_NETWORK

La table Apollo_Network contient les informations concernant les réseaux du protocole Apollo gérées au CERN. Remarquons qu'un réseau Apollo est une série d'adresses Apollo selon un principe similaire aux sous-réseaux TCP/IP (cfr. table TCPIP_Networks)

Champs	*Description*	*Clé*	*Clé primaire de*
Network Number	Numéro du réseau Apollo	1	
Network_Type	Type du réseau Apollo		
Landb_Pers_Id	Responsable du réseau Apollo		Personal_ Information
Remarks	Commentaires éventuels		
Modification_Date			

Champs	*Description*	*Clé*	*Clé primaire de*
Modified_By			
Under_Account			

CABLE

La table Cable contient les informations sur tous les câbles du réseau sur lesquels peuvent être branchées les diverses machines. Cette table est essentielle dans la gestion de la topologie du réseau.

Champs	*Description*	*Clé*	*Clé primaire de*
Cable_Id	Identificateur du câble	1	
Alias	Alias du nom du câble (n'est plus utilisé)		
Area	Nom de la zone Ethernet (le CERN est subdivisé en 11 zones Ethernet avec chacune un responsable)		
Logical_Segment	Segment logique où se trouve le câble (un segment logique est une zone entre deux bridges)		
Status	Information sur l'état du câble		
Length	Longueur du câble		
Ground	Point de mise à la masse du câble		
Landb_Pers_Id	Responsable du câble		Personal_ Information
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

CABLE_HISTORY

La table Cable_History est l'historique de la table Cable.

DECNET_ADDRESS

La table Decnet_Address contient les informations sur les adresses du protocole DECnet **version 4**. Cette version est celle utilisée au CERN au moment du développement de LANDB. On verra plus loin les problèmes posés par l'apparition d'une version nouvelle et sans compatibilité descendante.
La première forme d'une adresse DECnet version 4 est un nom composé lui-même d'un nom (souvent celui de la machine sinon un autre s'il est déjà utilisé) et d'un domaine (*pays.organisation*). Par défaut, le domaine d'une adresse DECnet au CERN est CH.CERN.

La deuxième forme de l'adresse est numérique et formée d'une paire de nombres: l'*area* (sous-ensemble d'adresses DECnet au niveau mondial) allant de 1 à 63 et le *node number* (numéro identificateur de l'adresse au sein de son area) allant de 1 à 1023.

Champs	*Description*	*Clé*	*Clé primaire de*
Usual_Name	Nom de la machine adressée		Network_Device
Domain	Domaine de l'adresse	1	
Name	Nom de l'adresse	2	
Area	1ère partie de l'adresse numérique		
Node_Number	2e partie de l'adresse numérique		
Software_Address	3e format de l'adresse en hexadécimal		
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

DECNET_ADDRESS_HISTORY

La table Decnet_Address_History est l'historique de la table Decnet_Address.

DECNET_ADDRESS_V5

La table Decnet_Address_V5 contient les informations sur les adresses du protocole DECnet **version 5**. Cette nouvelle version du protocole gère non seulement des adresses d'un format tout à fait différent de celui de la version 4, mais une nouvelle notion: celle d'alias. De plus, la transition de la version 4 à la version 5 est planifiée par Digital sur plusieurs années! Enfin, la notion de nom d'adresse n'est plus identifiante d'une adresse et est partageable par plusieurs adresses. Pour ces raisons, il a été décidé de définir de nouvelles tables pour gérer les adresses DECnet version 5 plutôt que de modifier les tables concernant les adresses DECnet version 4.
Les problèmes posés par cette solution seront analysés plus en détail dans la section d'analyse des interfaces utilisateur ci-après. Signalons seulement que les adresses version 5 ne sont qu'un miroir des adresses version 4 tant que ces dernières sont toujours utilisées. Les spécificités de la version 5 ne sont donc pas utilisables.
L'adresse DECnet version 5 a, elle aussi, deux formats.
Le format alphanumérique ne change pas par rapport à la version 4: il est composé du nom (souvent celui de la machine sinon un autre s'il est déjà utilisé) et d'un domaine (*pays.organisation*). Par défaut, le domaine d'une adresse DECnet au CERN est 'CH.CERN'.
Signalons que la version 5 de DECnet permet l'association de plusieurs adresses au même nom. Donc, les champs Domain et Node_Name ne sont que des références à la table Decnet_V5_Name décrite plus loin.
La deuxième forme de l'adresse, la forme numérique, est, quant à elle, tout à fait différente de la version 4. En effet, la version 4 ne permettait d'avoir au maximum que 63*1023 adresses donc 65449 adresses. Ce format a été changé étant donné le nombre croissant de

machines utilisant le protocole DECnet. Le nouveau format est composé de 4 parties: prefix, area, node_address et suffix.

Champs	*Description*	*Clé*	*Clé primaire de*
Usual_Name	Nom de la machine adressée		Network_Device
Prefix	1ère partie de l'adresse	1	
Area	2ème partie de l'adresse	2	
Node_Address	3ème partie de l'adresse	3	
Suffix	4ème partie de l'adresse		
Domain	Domaine de l'adresse		Decnet_V5_ Name
Node_Name	Nom de l'adresse		Decnet_V5_ Name
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

DECNET_ADDRESS_V5_HISTORY

La table Decnet_Address_V5_History est l'historique de la table Decnet_Address_V5.

DECNET_V5_ALIAS

La table Decnet_V5_Alias contient les alias des adresses du protocole DECnet **version 5**. La notion d'alias existe entre autres dans le protocole TCP/IP. Elle permet de donner un autre nom à une adresse. Lors de la recherche d'une adresse, on sélectionnera non seulement les noms "standards", mais également les alias. Cette propriété est intéressante lorsqu'on veut rendre invisible à l'utilisateur un changement de nom d'adresse. Il suffit alors de donner l'ancien nom de l'adresse à un alias, l'utlisateur pourra toujours utiliser l'ancien nom sans se rendre compte qu'il a changé. Plusieurs alias peuvent être définis pour une même adresse.
La version 5 de DECnet n'étant pas encore supportée, la table DECNET_V5_ALIAS est actuellement inutilisée.

Champs	*Description*	*Clé*	*Clé primaire de*
Alias	Alias	1	
Domain	Domaine de l'adresse correspondant à l'alias		
Node_Name	Nom de l'adresse correspondant à l'alias		
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

DECNET_V5_ALIAS_HISTORY

La table Decnet_V5_Alias_History est l'historique de la table Decnet_V5_Alias.

DECNET_V5_NAME

Comme nous l'avons déjà signalé, la version 5 du protocole DECnet dissocie nom et adresse de telle manière qu'il est possible d'avoir un nom unique pour plusieurs adresses. Cette propriété est particulièrement intéressante pour les machines multi-adresses. Un seul nom permet d'identifier toutes les adresses d'une même machine alors qu'avec la version 4 le nom était identifiant de l'adresse. La table Decnet_V5_Name contient les noms des adresses DECnet version 5.
La version 5 de DECnet n'étant pas supportée à ce jour et ces dernières étant un miroir parfait de la version 4, dans la réalité, à chaque nom n'est associée qu'une seule adresse.

Champs	*Description*	*Clé*	*Clé primaire de*
Domain	Domaine de l'(des) adresse(s)	1	
Node_Name	Nom de l'(des) adresse(s)	2	
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

DESCRIPTION

La table Description reprend toutes les lignes de commentaires qu'on peut avoir à propos de l' installation d'un produit (logiciel) sur une machine.

Champs	*Description*	*Clé*	*Clé primaire de*
Usual_Name	Machine sur laquelle le produit est installé	1	Installation
Product_Name	Nom du produit installé	2	Installation
Version_Product	Version du produit installé	3	Installation
Operating_System	Système d'exploitation sous lequel le produit est installé	4	Installation
Operating_System_Version	Version du système d'exploitation	5	Installation
Line_Number	Numéro de la ligne de commentaire	6	
Load_Date		7	
Comment_Line	Ligne de commentaire		

DEVICE_TYPE

La table Device_Type est une liste de valeurs contenant toute les machines officiellement gérées au CERN. Elle permet d'éviter la redondance d'information principalement due à des différences orthographiques. Il n'était, en effet, pas rare de voir une machine répertoriée

plusieurs fois dans la base CSNET comme par exemple: DECSTATION 5000, DecStation-5000, DEC Station/5000 DEC 5000... La structure de la table permet également la recherche de machines selon le constructeur (DEC, HP, IBM, SUN...), le type générique (Computer, Bridge, Gateway...) et/ou le modèle (VAX 6000, Sparc Station...) ce qui est essentiel lorsqu'on voit la liste impressionante de modèles aux noms peu évocateurs.
Le champ Generic_Type est lui-même issu d'une autre liste de valeurs.

Champs	*Description*	*Clé*	*Clé primaire de*
Maker	Fabricant de la machine	1	
Hardware_Type	Modèle de la machine	2	
Generic_Type	Type générique de la machine	3	Generic_Type
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

DEVTYPE_USER

La table Devtype_User est la table de liaison qui assure la relation Many-To-Many existant entre les tables User_Account et Device_Type. Elle reprend donc la liste des utlisateurs de la base de données LANDB et des types de machines dont ils ont plus spécifiquement la responsabilité. Ce type d'information est nouveau par rapport à CSNET.

Champs	*Description*	*Clé*	*Clé primaire de*
Maker	Fabricant de la machine	1	Device_Type
Hardware_Type	Modèle de la machine	2	Device_Type
Generic_Type	Type générique de la machine	3	Device_Type
Oracle_Account	Compte Oracle de l'utilisateur	4	User_Account

ERRORS

Cette table n'est utilisée que dans le cadre des interfaces utilisateur. Elle contient l'ensemble des messages d'erreurs propres à l'application LANDB ainsi que les causes et les solutions possibles. Elle sert donc autant lorsque l'application fonctionne pour donner un maximum d'informations à l'utilisateur en cas d'erreur que dans un but de documentation (liste des erreurs).

Champs	*Description*	*Clé*	*Clé primaire de*
Error_Code	Code identificateur de l'erreur	1	
Error_Text	Texte du message d'erreur		
Action	Solution(s) pour remédier à l'erreur		
Cause	Cause(s) possible(s) de l'erreur		

EXTERNAL_IP_LINE_HISTORY

La table External_Ip_Line_History est la table d'historique de la table External_TCPIP (cfr ci-dessous).

EXTERNAL_TCPIP

La table External_TCPIP regroupe d'ensemble des lignes externes TCP/IP gérées au CERN. Une ligne externe est un concept propre à Internet et représente une liaison point à point à l'extérieur d'un LAN (celui du CERN dans ce cas-ci). Cette ligne peut être une ligne privée, une ligne louée sur un réseau public commuté, une fenêtre satellite...
Les lignes externes TCP/IP doivent susciter une attention toute particulière quant on connaît les activités du CERN (1/3 du trafic universitaire européen de longue distance transite par le CERN...).
Remarquons que les coordonnées du contact ne sont pas une référence à la table Personal_Information. Ceci s'explique par le fait que le contact peut être une personne extérieure au CERN (de l'autre côté de la ligne).

Champs	*Description*	*Clé*	*Clé primaire de*
Usual_Name	Machine du CERN de laquelle part la ligne TCP/IP externe	1	Network_Device
Line_Name	Nom de la ligne externe	2	
End_Location	Nom de la machine où aboutit la ligne TCP/IP externe		
Organization	Organisation responsable de la ligne TCP/IP externe aux yeux d'Internet (CERN, MIT...)		
Flow_Rate	Débit de la ligne		
Line_Type	Type de ligne: louée, privée, satellite...		
Contact_Name	Nom du responsable		
Contact_Firstname	Prénom du responsable		
Contact_Phone	Numéro de téléphone du responsable		
Contact_E_Mail	Adresse E-Mail du responsable		
Local_Domain	Domaine de l'adresse TCP/IP locale		
Local_Name	Nom de l'adresse TCP/IP locale		
Local_IP1	1er byte de l'adresse TCP/IP locale		
Local_IP2	2e byte de l'adresse TCP/IP locale		
Local_IP3	3e byte de l'adresse TCP/IP locale		
Local_IP4	4e byte de l'adresse TCP/IP locale		
Distant_Domain	Domaine de l'adresse TCP/IP distante		
Distant_Name	Nom de l'adresse TCP/IP distante		
Distant_IP1	1er byte de l'adresse TCP/IP distante		
Distant_IP2	2e byte de l'adresse TCP/IP distante		
Distant_IP3	3e byte de l'adresse TCP/IP distante		
Distant_IP4	4e byte de l'adresse TCP/IP distante		
Remarks	Commentaires éventuels		
Modification_Date			

Champs	*Description*	*Clé*	*Clé primaire de*
Modified_By			
Under_Account			

FINANCIAL_ADMIN

La table Financial_Admin regroupe tous les comptes financiers concernant les achats de matériels, la maintenance...

Champs	*Description*	*Clé*	*Clé primaire de*
Financial_Account	Identificateur du compte	1	
Landb_Pers_Id	Numéro identificateur du responsable du compte		Personal_ Information
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

FINANCIAL_ADMIN_HISTORY

La table Financial_Admin_History est l'historique de Financial_Admin.

GENERIC_TYPE

La table Generic_Type est une liste de valeurs contenant les types génériques de machines existant sur le réseau: ordinateur, terminal, bridge, repeater, gateway...

Champs	*Description*	*Clé*	*Clé primaire de*
Generic_Type	Nom du type générique	1	

HARDWARE_ADDRESS

La table Hardware_Address contient toutes les adresses hardware des machines branchées sur le réseau ainsi que, pour chacune d'elles, le type d'adresse, c'est-à-dire le type de protocole de niveau inférieur: Ethernet, Token Ring, FDDI.

Champs	*Description*	*Clé*	*Clé primaire de*
Usual_Name	Nom de la machine adressée		Network_Device
Hardware_Address	Adresse hardware	1	
Address_Type	Type de protocole		Hardware_ Address_Type
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

HARDWARE_ADDRESS_HISTORY

Hardware_Address_History est la table d'historique de Hardware_Address.

HARDWARE_ADDRESS_TYPE

La table Hardware_Address_Type est la liste de valeur permettant de choisir un type d'adresse hardware (type de protocole) pour la table Hardware_Address.

Champs	*Description*	*Clé*	*Clé primaire de*
Address_Type	Type de protocole	1	

HARDWARE_CONNECTION

La table Hardware_Connection contient toutes les connections entre les machines et les câbles. Elle permet également de spécifier la position de la machine sur le câble ainsi que l'endroit où est faite la connexion sur la machine (slot, port). Cette table est essentielle dans la gestion de la topologie du réseau.
Les Multiport Repeaters comportent une entrée et plusieurs sorties (ports). Il est donc nécessaire de pouvoir spécifier à quel endroit de l'appareil est faite la connexion. De plus, les modèles les plus récents proposent une série de slots de connexion étant eux-mêmes divisés en ports. Ceci explique la présence des deux champs Port et Slot pour la spécification de l'endroit où la connexion est faite sur la machine.

Champs	*Description*	*Clé*	*Clé primaire de*
Usual_Name	Nom de la machine câblée	1	Network_Device
Cable_Id	Identificateur du câble	2	Cable
Position	Position de la machine sur le câble		
Slot	Position du câble sur la machine		
Port	Position du câble sur la machine		
Fiber	Signale s'il s'agit d'une connexion fibre optique ou non		
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

HARDWARE_CONNECTION_HISTORY

La table Hardware_Connection_History est la table d'historique de la table Hardware_Connection.

HELP

La table Help n'est utilisée que dans le cadre des interface utilisateur. Elle contient l'ensemble des pages d'aide en ligne utilisables depuis l'application LANDB. Une page d'aide en ligne est définie pour chaque block de chaque forme.
Remarquons que la page d'aide a été découpée en 15 champs correspondant chacun à une ligne plutôt que d'être placée dans un seul champ de type LONG (qui peuvent contenir n'importe quelle combinaison alphanumérique jusqu'à 65534 caractères). Ce choix a été fait de façon à faciliter la présentation de l'aide à l'écran. En effet, les fonctions de manipulations de champ de type LONG peuvent provoquer une mise en forme confuse de l'aide.

Champs	*Description*	*Clé*	*Clé primaire de*
Form	Nom de la forme à laquelle appartient le block sur lequel porte l'aide	1	
Block	Nom du block sur lequel porte l'aide	2	
Help_Line1	1ère ligne d'aide		
Help_Line2	2e ligne d'aide		
Help_Line3	3e ligne d'aide		
Help_Line4	4e ligne d'aide		
Help_Line5	5e ligne d'aide		
Help_Line6	6e ligne d'aide		
Help_Line7	7e ligne d'aide		
Help_Line8	8e ligne d'aide		
Help_Line9	9e ligne d'aide		
Help_Line10	10e ligne d'aide		
Help_Line11	11e ligne d'aide		
Help_Line12	12e ligne d'aide		
Help_Line13	13e ligne d'aide		
Help_Line14	14e ligne d'aide		
Help_Line15	15e ligne d'aide		

INSTALLATION

La table Installation représente toutes les occurrences de l'installation d'un produit (software) sur une machine et sous un système d'exploitation donnés. Toutes les informations permettant à la personne chargée de gérer cet aspect du parc machine sont également rassemblées dans cette table: nombre d'utilisateurs, responsable, licence...

Champs	*Description*	*Clé*	*Clé primaire de*
Usual_Name	Machine sur laquelle le produit est installé	1	Network_Device
Product_Name	Nom du produit installé	2	Product
Version_Product	Version du produit installé	3	Product
Operating_System	Système d'exploitation sous lequel le produit est installé	4	Product
Operating_System_Version	Version du système d'exploitation	5	Product

Champs	*Description*	*Clé*	*Clé primaire de*
Landb_Pers_Id	Identificateur du responsable de l'installation du produit		Personal_ Information
Installation_Date	Date de l'installation du produit		
Max_Users	Nombre maximum d'utilisateurs		
License_Number	Numéro de la licence		
Swpid	Référence fournisseur de la copie du progiciel		
Enable_Code	Code software fournie par le fournisseur pour la protection contre le piratage		
Machine_Code	Code annexe à la clé de protection		
Enable_Code_Date	Date d'expiration de la clé de protection		
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

INSTALLATION_DATABASE

La table Installation_Database a une origine historique. Elle provient d'une ancienne application de la section Database. Elle permet de signaler l'installation d'une version d'un SGBD pour une base de données et une machine particulière.

Champs	*Description*	*Clé*	*Clé primaire de*
Database_Name	Nom de la base de données pour laquelle a été installé le SGBD	1	
Usual_Name	Nom de la machine où est installé le SGBD	2	Installation
Product_Name	Nom du SGBD	3	Installation
Version_Product	Version du SGBD	4	Installation
Operating_System	Système d'exploitation sous lequel fonctionne le SGBD	5	
Operating_System_Version	Version du système d'exploitation	6	

INSTALLATION_HISTORY

La table Installation_History est la table d'historique de la table Installation.

LICENSE

La table License contient toutes les licences d'exploitation des logiciels installés sur les machines gérées par LANDB.

Champs	*Description*	*Clé*	*Clé primaire de*
License_Number	Numéro de la licence	1	
Landb_Pers_Id	Responsable de la licence		Personal_ Information
Cost	Coût de la licence		
Beginning_Date	Date de début de la licence		
Expiration_Date	Date d'expiration de la licence		
Number_Copies	Nombre de copies de la licence		
Order_Number	Numéro du bon de commande		
Seller	Vendeur de la licence		
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

LICENSE_HISTORY

La table License_History est la table d'historique de la table License.

LICENSE_PRODUCT

La table License_Product est la table de liaison qui assure la relation Many-To-Many existant entre les tables License et Product. Elle représente le lien qui peut exister entre applications et licences d'exploitation.
Une relation Many-To-Many est nécessaire pour exprimer les nombreuses possibilités de licences existantes: licences sites, licences sites pour un système d'exploitation, licences pour un sous-produit, licences pour plusieurs produits...

Champs	*Description*	*Clé*	*Clé primaire de*
License_Number	Numéro de la licence	1	License
Product_Name	Nom du produit	2	Product
Version_Product	Version du produit	3	Product
Operating_System	Système d'exploitation sous lequel le produit est installé	4	Product
Operating_System_Version	Version du système d'exploitation	5	Product

LOGICAL_GROUP

La table Logical_Group a deux fonctions.
La table Logical_Group est tout d'abord une liste de valeur contenant des groupes de l'organigramme administratif du CERN. Il peut s'agir de divisions (CN, AT, PE...), de sections (CN/CS, AT/DI, AT/CO...) ou de groupes (CN/CS/IN, CN/CS/EN, CN/CS/CH...). Cette liste de valeur sert actuellement à connaître le groupe ayant administrativement la gestion d'une machine.
La table représentant des groupements administratifs, on a étendu ses propriétés jusqu'à lui permettre de représenter également des groupements techniques. Ainsi, la table Logical_Group contient également les noms des clusters de machines.

Champs	*Description*	*Clé*	*Clé primaire de*
Group_Name	Nom du groupe	1	
Alias	Autre nom du groupe		
Landb_Pers_Id	Identificateur du responsable du groupe		Personal_ Information
Server	Pour les clusters uniquement: nom du serveur ou du boot node du cluster		Network_Device
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

NETWORK_DEVICE

La table Network_Device contient les informations sur toutes les machines connectées au réseau: ordinateurs, stations de travail, terminaux, gateways, bridges, repeaters... Elle contient les informations directement liées à la "boîte", mais également des informations appartenant plutôt à d'autres catégories (software, administration...) mais d'une importance si mineure qu'elles ne justifient pas l'existence d'une table propre. C'est le cas de toute une série de champs ne concernant que le parc PC (et signalés par le mot PC dans le tableau qui suit).
Elle est la table centrale de la base autour de laquelle tout tourne.

Champs	*Description*	*Clé*	*Clé primaire de*
Usual_Name	Nom de la machine, de la "boîte"	1	
Operating_System	Type de système d'exploitation (UNIX, DOS, VM/CMS...)		Operating_ System
Operating_System_Version	Version du système d'exploitation		Operating_ System
Location	Local où se trouve la machine (Bâtiment-Etage-Local)		
Contact_Landb_Pers_Id	Responsable de la machine		Personal_ Information
Maker	Fabricant de la machine (DEC, IBM...)		Device_Type

Champs	*Description*	*Clé*	*Clé primaire de*
Hardware_Type	Nom du modèle de machine (VAX 600, Alpha...)		Device_Type
Generic_Type	Type de machine (Ordinateur, Gateway, Bridge, Terminal...)		Device_Type
Serial_Number	Numéro de série de la machine		
User_Landb_Pers_Id	Utilisateur principal de la machine		Personal_ Information
Installation_Date	Date d'installation de la machine		
Inventory_Number	Numéro d'inventaire		
Group_Name	Groupe qui a la responsabilité de la machine (CN/CS/IN...)		Logical_Group
Memory	Mémoire dont dispose la machine		
Order_Number	Numéro du bon de commande de la machine		
Financial_Account	Compte associé à l'achat de la machine		Financial_ Account
Mail_Router	Routeur par défaut pour le courrier électronique		Network_Device
TCPIP_Software	Software de gestion du protocole TCP/IP		
Maintenance_Contract_Numb	Numéro du contrat de maintenance de la machine		
Monthly_Maintenance_Cost	Coût mensuel de la maintenance		
Cost_Machine	Coût unitaire de la machine		
Maintenance_Company	Compagnie chargée de la maintenance de la machine		
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			
CPU_Serial_Number	(PC) Numérode série du CPU		
Bios_Id	(PC) Numéro de série du BIOS		
ROM_Serial_Number	(PC) Numéro de série de la ROM		
ROM_Version	(PC) Version de la ROM		
DOS_Free_Memory	(PC) Mémoire conventionnelle disponible		
EMS_Total_Memory	(PC) Mémoire paginée totale		
EMS_Total_Free_Memory	(PC) Mémoire paginée disponible		
EMS_Page_Address	(PC)		
EMS_Version	(PC) Version du pilote de mémoire paginée		
Memory_Ext	(PC) Extension de mémoire		
Enhanced_Key_Support	(PC) ?		
Enhanced_Key_Available	(PC) ?		
Bios_Date	(PC) Date du Bios		

Champs	*Description*	*Clé*	*Clé primaire de*
Serial_Ports	(PC) Nombre de ports série		
Processor	(PC) Type de processeur		
Bios_Signature	(PC) ?		
Coprocessor	(PC) Type de co-processeur		
Parallel_Ports	(PC) Nombre de ports parallèle		
Accepted_Date	(PC) Date d'acceptation de l'achat		
Novell_Network_Address	(PC) Adresse réseau Novell		
Game_Ports	(PC) Nombre de ports manette de jeu		
Hostid	Numéro d'identification unique au monde pour les machines Unix		
Maintenance_Number	?		
Apollo_Node_Id	Identificateur unique au monde d'une machine Apollo. N'est plus utilisé depuis le rachat de Apollo par HP.		
Logical_Segment	Segment logique auquel appartient la machine		
Lat_Service	?		
SQE	?		

NETWORK_DEVICE_HISTORY

Network_Device_History est la table d'historique de Network_Device.

OPERATING_SYSTEM

La table Operating_System est la liste de valeurs contenant tous les systèmes d'exploitation rencontrés sur les machines du CERN. Le nom du système (MS-DOS, AIX, UNIX, ULTRIX, VMS...) et sa version (3.4, 2.01, 5.6v2...) sont placés dans des champs différents afin de pouvoir effectuer des recherches sur le nom seul indépendamment de toute version. Le champ Top_Of_Range_Computer signale l'ordinateur le plus puissant existant au CERN fonctionnant avec le système d'exploitation de l'enregistrement. Cette information est utile dans le calcul de coût de licences, entre autres pour Oracle qui fixe le prix d'une licence d'un de ses produits pour un système d'exploitation donné en fonction de la machine la plus puissante susceptible de l'utiliser.

Champs	*Description*	*Clé*	*Clé primaire de*
Operating_System	Nom du système d'exploitation	1	
Operating_System_Version	Version du système d'exploitation	2	
Top_Of_Range_Computer	Ordinateur le plus puissant sous ce S.E.		Network_Device
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

OPERATING_USER

La table Operating_User est la table de liaison qui assure la relation Many-To-Many existant entre les tables Operating_System et User. Elle permet de représenter l'intérêt d'un utilisateur particulier pour un système d'exploitation. Cette table permet de créer une liste d'utilisateurs susceptibles d'être intéressés par l'achat d'un nouveau produit fonctionnant sur un système d'exploitation particulier.

Champs	*Description*	*Clé*	*Clé primaire de*
Operating_System	Nom du système d'exploitation	1	Operating_ System
Operating_System_Version	Version du système d'exploitation	2	Operating_ System
Oracle_Account	Compte Oracle de l'utilisateur	3	User_Account

PARTITION_DISK

La table Partition_Disk décrit les partitions d'un disque. Elle est donc associée à la table Peripherics dont les disques font partie.

Champs	*Description*	*Clé*	*Clé primaire de*
Peripheric_Name	Nom du disque	1	Peripheric
Usual_Name	Nom de la machine	2	Peripheric
Partition_Line	Définition de la partition	3	
Comment_Partition	Commentaires sur la partition		
Modification_Date			

PERIPHERIC

La table Peripheric (qui devrait d'ailleurs plutôt s'appeler Peripheral) regroupe tous les périphériques auxquels peuvent être connectées les machines du réseau.
Cette table est assez semblable à la table Network_Device, mais les développeurs ont estimé devoir créer une table à part étant donné la différence fondamentale de destination entre les périphériques et les autres machines du réseau (pas d'installation d'applications, pas toujours d'adresses, etc).

Champs	*Description*	*Clé*	*Clé primaire de*
Peripheric_Name	Nom du périphérique	1	Peripheric_Type
Usual_Name	Nom de la machine sur lequel est branché le périphérique	2	Network_Device
Peripheric_Type	Tpye de périphérique	3	Peripheric_Type
Address			
Interface			
Internal			
Maker	Fabricant du périphérique		

Champs	*Description*	*Clé*	*Clé primaire de*
Order_Number	Numéro du bon de commande		
Model	Modèle de périphérique		
Serial_Number	Numéro de série		
Maintenance_Company	Compagnie chargée de la maintenance		
Maintenance_Contract_Number	Numéro du contrat de maintenance		
Maintenance_Number	Numéro de maintenance		
Cost_Peripheric	Coût unitaire du périphérique		
Landb_Pers_Id	Identifiant du responsable		Personal_ Information
Monthly_Maintenance_Cost	Coût mensuel de la maintenance		
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

PERIPHERIC_HISTORY

La table Peripheric_History est l'historique de la table Peripheric.

PERIPHERIC_TYPE

La table Peripheric_Type contient une liste des types de périphériques. Elle est composée du type lui-même (imprimante, disque...) et d'un nom qui est en fait un préfixe obligatoire dans les noms de certains périphériques.

Champs	*Description*	*Clé*	*Clé primaire de*
Peripheric_Type	Type de périphérique	1	
Peripheric_Name	Préfixe de nom obligatoire	2	

PERSONAL_INFORMATION

La table Personal_Information contient toutes les informations sur les personnes. Elle sert de liste de valeur pour le choix des contacts, responsables, utilisateurs...
Sa place au sein de la base de données LANDB est capitale, étant donné la taille et la diversité du réseau à gérer. Le nombre d'utilisateurs, responsables, contacts... étant énorme, il est nécessaire de bénéficier d'une liste de valeurs complète permettant d'éviter un maximum de redondance (dû, ici, principalement aux fautes d'orthographes dans les noms), mais également d'obtenir tous les renseignements nécessaires pour contacter la personne: e-mail, téléphone, sémaphone...
Remarquons que 3 champs ont été définis pour des numéros de téléphone: Phone_Num1, Phone_Num2 et Phone_Num3. Cette solution a été préférée à la création d'une table supplémentaire Phone_Num contenant les numéros de téléphone, car la complexification causée n'aurait pu être justifiée par l'avantage mineur d'une répétitivité infinie du champ

Phone_Num. Dans ce cas précis, il semble que le nombre de 3 numéros de téléphone devrait être largement suffisant et stable dans l'avenir.
Cette table est créée puis mise à jour périodiquement par programme batch à partir de bases de données de personnes existant au CERN: la base de données du service du personnel, la base de données CCBD regroupant tous les comptes utilisateurs sur CERNVM (le mainframe central du CERN) et la base de données regroupant toutes les adresses électroniques: EMDIR.
La référence à ces bases est d'ailleurs conservée dans les champs: CERN_Id, CCID, Pers_Num et Source.

Champs	*Description*	*Clé*	*Clé primaire de*
Landb_Pers_Id	Identificateur (propre à la base LANDB, d'où le nom du champ)		
CERN_Id	Identifiant de la base du personnel		
CCID	Identifiant de la base CCBD		
Pers_Num	Identifiant de la base EMDIR		
Surname	Nom de famille de la personne		
First_Name	Prénom(s) de la personne		
Pref_Name	Nom habituellement donné à la personne (surnom, diminutifs...)		
Home_Inst	Institut d'origine de la personne		
Division	Division à laquelle appartient la personne		
CERN_Group	Groupe auquel appartient la personne		
Build	Bâtiment du bureau de la personne		
Flr_Room	Etage et pièce du bureau de la personne		
Phone_Num1	1er numéro de téléphone		
Phone_Num2	2e numéro de téléphone		
Phone_Num3	3e numéro de téléphone		
Beep_Num	Numéro de sémaphone		
Mint_Addr	Adresse de courier électronique		
Source	Base de données source de l'enregistrement		
Last_Modification	Date de la dernière mise à jour		
Flag_Known	Signale si la personne existe dans d'autres bases de données au CERN		
Remarks	Commentaires éventuels		

PRODUCT

La table Product contient la liste des produits installés sur les machines du réseau (donc les applications en tout genre, pas les systèmes d'exploitations repris dans la table Operating_System).
La structure de la table permet que des produits soient définis à la manière d'un arbre: un produit père peut être composé de plusieurs produits fils. Par exemple, Oracle version 6 est composé de Oracle version 6 pour Unix, Oracle version 6 pour VMS, Oracle version 6 pour

VM/CMS. Ou encore, PL/SQL 2.0.14 pour Solaris 2.2 est un sous-produit d'Oracle 7.0.12 pour Solaris 2.2.
L'élimination du type de chemin récursif ainsi créé ne se justifie pas étant donné le caractère optionnel de ce dernier.

Champs	*Description*	*Clé*	*Clé primaire de*
Product_Name	Nom du produit	1	
Version_Product	Version du produit	2	
Operating_System	Système d'exploitation avec lequel fonctionne ce produit	3	Operating_ System
Operating_System_Version	Version du système d'exploitation avec lequel fonctionne ce produit	4	Operating_ System
Landb_Pers_Id	Responsable de ce produit au CERN		Personal_ Information
Up_Product_Name	Nom du produit père		Product
Up_Version_Product	Version du produit père		Product
Up_Operating_System	Système d'exploitation avec lequel fonctionne le produit père		Product
Up_Operating_System_Version	Version du système d'exploitation avec lequel fonctionne le produit père		Product
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

PRODUCT_USER

La table Product_User est la table de liaison qui assure la relation Many-To-Many existant entre les tables User et Product. Elle représente le fait qu'un utilisateur employe un produit donné. Cette table pourrait, entre autres, permettre de créer une liste d'utilisateurs susceptibles d'être intéressés par l'achat d'une nouvelle version d'un produit qu'ils utilisent déjà.

Champs	*Description*	*Clé*	*Clé primaire de*
Product_Name	Nom du produit	1	
Version_Product	Version du produit	2	
Operating_System	Système d'exploitation avec lequel fonctionne ce produit	3	Operating_ System
Operating_System_Version	Version du système d'exploitation avec lequel fonctionne ce produit	4	Operating_ System
Oracle_Account	Compte Oracle de l'utilisateur	5	User_Account

ROLE_ACCOUNT

La table Product_User est la table de liaison qui assure la relation Many-To-Many existant entre les tables User_Account et Role_Tab_Privs. Elle représente l'octroi d'un rôle type à un utilisateur pour une table donnée (cfr table Role_Tab_Privs ci-dessous).

Champs	*Description*	*Clé*	*Clé primaire de*
Oracle_Account	Compte Oracle de l'utilisateur	1	User_Account
Role	Type de rôle accordé à l'utilisateur	2	Role_Tab_Privs

ROLE_TAB_PRIVS

La table Role_Tab_Privs n'est utile qu'au gestionnaire de l'application LANDB. Elle contient une liste de rôles (utilisateurs) types associés à une table et les privilèges d'accès à cette table qui en découlent.
Ainsi, lorsqu'un nouvel utilisateur sera enregistré dans la base de données, il suffira de lui assigner un rôle donné (TCP/IP Manager, DECnet Manager, Application Manager, Guest...) et de consulter (via un programme) tous les enregistrements de Role_Tab_Privs correspondants pour connaître et définir automatiquement les privilèges d'accès à la base de données dont il bénéficie.
Ce système permet de ne pas devoir définir des dizaines de privilèges lors de l'enregistrement de chaque nouvel utilisateur de l'application en lui assignant simplement un rôle type.

Champs	*Description*	*Clé*	*Clé primaire de*
Role	Rôle type	1	
Table_Name	Table associée au rôle	2	
Object_Type	Objet du privilège (Table, Vue, Fonction, Procédure...)		
Select_Priv	Privilège pour la sélection		
Insert_Priv	Privilège pour l'insertion		
Update_Priv	Privilège pour la modification		
Delete_Priv	Privilège pour la suppression		
Alter_Priv	Signale si l'utilisateur peut exécuter des instructions d'altération (modification de la structure d'une table)		
Reference_Priv	Signale si l'utilisateur peut créer une contrainte référentielle sur la table		
Execute_Priv	Signale si l'utilisateur peut exécuter l'objet du privilège (quand il s'agit de Fonctions, de Procédures...)		

SYSTEM

La table System reprend différentes informations propres à l'application LANDB.
Elle regroupe 3 types de données: les paramètres d'application concernant LANDB, les batch timestamps qui permettent de retenir la dernière date d'exécution d'un batch et les paramètres de programme qui sont des paramètres à passer à un programme SQL externe à LANDB.
Cette table devrait être beaucoup moins importante avec la version 7 de Oracle qui permet le passage de paramètres à des programmes SQL.
Cette table ne contient qu'un seul enregistrement ce qui justifie l'absence de clé primaire.

Champs	*Description*	*Clé*	*Clé primaire de*
Application_Parameter1			
Application_Parameter2			
Application_Parameter3			
Application_Parameter4			
Application_Parameter5			
Batch_Timestamp1			
Batch_Timestamp2			
Batch_Timestamp3			
Batch_Timestamp4			
Batch_Timestamp5			
Program_Parameter1			
Program_Parameter2			
Program_Parameter3			
Program_Parameter4			
Program_Parameter5			
Program_Parameter6			
Program_Parameter7			
Program_Parameter8			
Program_Parameter9			
Program_Parameter10			

TCPIP_ADDRESS

La table TCPIP_Address contient toutes les adresses du protocole de haut niveau TCP/IP associées à des machines du réseau.
La première forme de l'adresse est un nom découpé lui-même en nom et domaine (ce nom est assez semblable à celui d'une adresse DECnet à la différence près que le domaine s'exprime dans l'autre sens à savoir: *organisation.pays*). Ce nom identifie l'adresse.
La deuxième forme de l'adresse est une série de 4 bytes qui identifie également l'adresse.

Champs	*Description*	*Clé*	*Clé primaire de*
Usual_Name	Nom de la machine adressée		Network_Device
Domain	Domaine de l'adresse	1	
Name	Nom de l'adresse	2	

Champs	*Description*	*Clé*	*Clé primaire de*
IP1	1er byte de l'adresse numérique		
IP2	2e byte de l'adresse numérique		
IP3	3e byte de l'adresse numérique		
IP4	4e byte de l'adresse numérique		
Well_Known_Address	Signale si l'adresse doit se retrouver dans le Name Server		
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

TCPIP_ADDRESS_HISTORY

La table TCPIP_Address_History est l'historique de la table TCPIP_Address.

TCPIP_ALIAS

La table TCPIP_Alias contient tous les alias associés à des adresses TCPIP. Tout comme pour le protocole DECnet, il est possible de définir plusieurs noms pour une adresse TCP/IP, un seul d'entre eux est le nom régulier de l'adresse, les autres étant des alias. Pour rappel, cette propriété est intéressante lorsqu'on veut rendre invisible à l'utilisateur un changement de nom d'adresse. Il suffit alors de donner l'ancien nom de l'adresse à un alias, l'utlisateur pourra toujours utiliser l'ancien nom sans se rendre compte qu'il a changé.

Champs	*Description*	*Clé*	*Clé primaire de*
Domain	Domaine de l'adresse associée à l'alias	1	TCPIP_Address
Name	Nom de l'adresse associée à l'alias	2	TCPIP_Address
Alias	Alias de l'adresse		
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

TCPIP_ALIAS_HISTORY

La table TCPIP_Alias_History est l'historique de la table TCPIP_Alias.

TCPIP_AREA

La table TCPIP_Area contient la liste des zones TCP/IP définies au CERN. Ce concept est absolument interne au CERN et n'est pas un concept TCP/IP.
Le nombre d'adresses TCP/IP gérées par le CERN est impressionnant: plus de 65000. Il est donc nécessaire de diviser ce "parc d'adresses" en zones (area). Une zone est donc un sous-

ensemble continu d'adresses TCP/IP du CERN dont la gestion est confiée à une personne. Une personne a le droit d'allouer les adresses de ses zones selon ses besoins.
Les zones TCP/IP sont, en général, définies en fonction des besoins actuels, mais également futurs des responsables de l'adressage TCP/IP. Dès lors, il n'est pas rare de voir des zones TCP/IP de très grande taille, voire de trop grande taille et qui seront sous-utilisées.
De plus, les gestionnaires du parc d'adresses TCP/IP s'occupent soit de grandes parties soit de parties très restreintes de ce parc. Il n'est donc pas rare de voir des zones d'adresses TCP/IP "se chevaucher" créant ainsi le risque de voir naître des conflits entre plusieurs personnes désirant allouer la même adresse à des machines différentes. On verra plus loin comment les interfaces utilisateur permettent de limiter ce problème.

Champs	***Description***	***Clé***	***Clé primaire de***
Area_Name	Nom de la zone d'adresses TCP/IP	1	
IP1	1er byte de l'adresse de début ***et*** de fin de zone		
IP2_Low	2e byte de l'adresse de début de zone		
IP3_Low	3e byte de l'adresse de début de zone		
IP4_Low	4e byte de l'adresse de début de zone		
IP2_High	2e byte de l'adresse de fin de zone		
IP3_High	3e byte de l'adresse de fin de zone		
IP4_High	4e byte de l'adresse de fin de zone		
Oracle_Account	Bénéficiaire de la zone		User_Account
Landb_Pers_Id	Personne réelle en charge de la gestion de la zone		Personal_ Information
CommentLine1	1ère ligne de commentaire (but...)		
CommentLine2	2e ligne de commentaire (but...)		
CommentLine3	3e ligne de commentaire (but...)		
CommentLine4	4e ligne de commentaire (but...)		
CommentLine5	5e ligne de commentaire (but...)		
CommentLine6	6e ligne de commentaire (but...)		
CommentLine7	7e ligne de commentaire (but...)		
CommentLine8	8e ligne de commentaire (but...)		
CommentLine9	9e ligne de commentaire (but...)		
CommentLine10	10e ligne de commentaire (but...)		
CommentLine11	11e ligne de commentaire (but...)		
CommentLine12	12e ligne de commentaire (but...)		
Modification_Date			
Modified_By			
Under_Account			

TCPIP_NETWORK

La table TCPIP_Network regroupe les informations concernant les sous-réseaux TCP/IP du CERN.
Le protocole TCP/IP définit 4 classes de sous-réseaux correspondant au nombre de bytes fixés dans l'adresse: réseau, sous-réseau, area et enfin adresse [COM]. Les 3 premiers bytes

suffisent à définir les sous-réseaux. Pour cette raison, le 4ème byte n'est pas repris dans la table.

Champs	*Description*	*Clé*	*Clé primaire de*
Network_Name	Nom du sous-réseau	1	
Class	Classe de sous-réseau		
IP1	1er byte des adresses du sous-réseau		
IP2	2e byte des adresses du sous-réseau		
IP3	3e byte des adresses du sous-réseau		
Landb_Pers_Id	Personne responsable du sous-réseau		Personal_ Information
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

USER_ACCOUNT

La table User_Account regroupe tous les comptes Oracle pouvant avoir accès à l'application LANDB. Le but de cette table est simplement de pouvoir associer un commentaire de maximum 9 lignes à tout compte Oracle. Ce commentaire est souvent le nom de la personne utilisant normalement ce compte, son adresse e-mail, son téléphone et sa fonction en ce qui concerne LANDB. Il aurait peut-être été plus intéressant en ce qui concerne les données à caractère personnel (nom, téléphone, e-mail...) d'associer cette table à la table Personal_Information. Cela n'a pas été fait, probablement dans un souci de simplicité.

Champs	*Description*	*Clé*	*Clé primaire de*
Oracle_Account	Nom de l'utilisateur Oracle	1	
Remarks1	1ère ligne de commentaires		
Remarks2	2e ligne de commentaires		
Remarks3	3e ligne de commentaires		
Remarks4	4e ligne de commentaires		
Remarks5	5e ligne de commentaires		
Remarks6	6e ligne de commentaires		
Remarks7	7e ligne de commentaires		
Remarks8	8e ligne de commentaires		
Remarks9	9e ligne de commentaires		
Modification_Date			
Modified_By			
Under_Account			

USER_PROGRAM

La table User_Program regroupe tous les programmes que les utilisateurs peuvent exécuter depuis l'application LANDB. Ces programmes sont principalement des batch permettant des opérations de grande envergure sur la base de données (mise à jour des bases de données propriétaires DECnet et TCP/IP, backup particuliers...), mais peuvent également être des applications étrangères à LANDB mais nécessaires à l'utilisateur.
Au moment de la rédaction de ce document, la liste des programmes exécutables depuis LANDB étaient tous les outils de développement de l'application elle-même (SQL*Forms 3.0, SQL*Plus, Oracle*Case...) ainsi que des batch de mise à jour de la base de données.
Il est à noter qu'un programme est toujours associé à un système d'exploitation particulier et la commande permettant de l'exécuter depuis LANDB n'est utilisable que sous ce système particulier.

Champs	*Description*	*Clé*	*Clé primaire de*
Program_Name	Nom du programme	1	
System	Système d'exploitation du programme	2	
Call_Command	Ligne de commande d'exécution du programme		
Description	Description sommaire du programme		
Remarks	Commentaires éventuels		
Modification_Date			
Modified_By			
Under_Account			

USER_USER_PROGR

La table User_User_Progr est la table de liaison qui assure la relation Many-To-Many existant entre les tables User_Program et User_Account. Elle représente la possibilité pour un utilisateur de lancer un programme depuis l'application LANDB. Cette table permet de définir la liste des programmes externes qu'un utilisateur a le droit d'exécuter.

Champs	*Description*	*Clé*	*Clé primaire de*
Program_Name	Nom du programme	1	User_Program
System	Système d'exploitation du programme	2	User_Program
Oracle_Account	Compte Oracle de l'utilisateur	3	User_Account

5.6. Conclusion

La base de données de l'application LANDB est, comme nous venons de voir, extrêmement souple et modulaire. Les développeurs ont néanmoins évité de verser dans l'excès en créant une table secondaire à chaque fois que l'occasion s'en présentait, et cela, en justifiant leur décision.
Cette souplesse répond parfaitement aux objectifs d'adaptabilité pour une meilleure maintenance et de flexibilité pour faire face à l'hétérogénéité des réseaux à gérer que s'étaient fixés les développeurs.
La base de données met en plus en place une série de structures de données qui, en parallèle avec les interfaces utilisateur, permettront de bénéficier de mécanismes assurant la sécurité et le contrôle de certaines données sensibles; nous parlons ici des listes de valeurs et des historiques.

Chapitre 6
Analyse des interfaces

Une bonne structure de données ne suffit pas si des outils performants d'accès aux données ne l'accompagnent pas. Nous allons donc analyser ici le développement des interfaces utilisateur de l'application LANDB en nous basant sur un travail effectué dans le cadre d'un stage effectué dans la division CN du CERN. Cette analyse devrait permettre de mettre en lumière quelques principes nous paraissant fondamentaux dans l'élaboration des interfaces utilisateur d'applications semblables à LANDB: base de données complexe et de structure instable, utilisateurs nombreux et hétérogènes.

Pour rappel, les interfaces utilisateur doivent être définies de façon à assurer la sécurité de la base et le contrôle de données dites sensibles. De plus, elles doivent être assez modulaires pour pouvoir répondre aux attentes d'utilisateurs très divers et pour être facilement adaptées aux changements pouvant intervenir dans l'avenir (ajout de nouvelles informations, modification des structures de données existantes, changement dans les tâches des utilisateurs...).

Nous verrons comment ces principes ont été mis en œuvre dans le cadre particulier de LANDB pour les interfaces permettant la gestion de l'adressage des machines du réseau (DECnet, TCP/IP et Hardware), seules interfaces terminées au début de la rédaction de ce mémoire.

6.1. Présentation générale des interfaces

6.1.1. Fonctionnement de SQL*Forms 3.0

Le processus de création d'une forme avec SQL*forms 3.0 comporte 2 étapes principales (décrites à la figure 6.1):

- Générer grâce au "Designer" un code source définissant la forme sous tous ses aspects (ce code est indépendant de toute plateforme!)
- A partir du code source, générer un code exécutable pour une plateforme spécifique. Il faut donc générer autant de codes exécutables qu'il y a de plateformes sur lesquelles on désire faire fonctionner la forme.

On peut rapprocher ce principe de celui de code source et de compilation pour les langages de programmation.

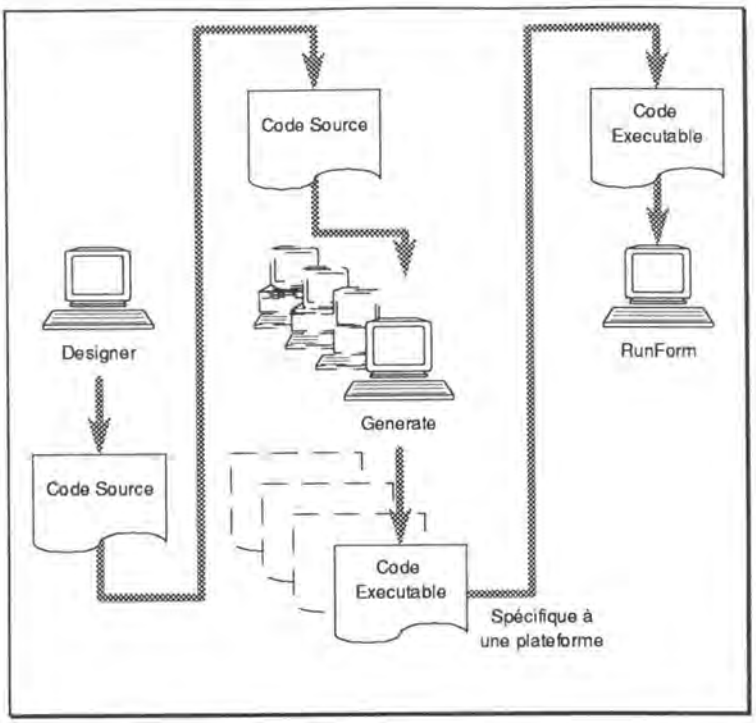

*Fig. 6.1.: Processus de création d'une forme sur SQL*forms 3.0*

Chacune de ces étapes est effectuée par un outil spécifique de SQL*forms:

- Le code source est généré automatiquement dans un fichier texte à partir du "Designer" de SQL*forms. Le designer utilise d'ailleurs des menus déroulants et de formes semblables à celles générées par SQL*forms.
- La "compilation" du code source en un code éxécutable est faite via l'option "Generate" de SQL*forms.
- Enfin, l'exécution de la forme est rendue possible grâce au programme "RunForm" de SQL*forms.

Le design d'une forme (génération du code source) consiste à en définir les 3 composantes principales:

- Structuration et définition des données manipulées par la forme: block, champs...
- Programmation événementielle
- Procédures de niveau général

Nous allons maintenant les décrire en détail.

6.1.2. Structuration des éléments de la forme et liens avec la base de données

Une forme est décomposée de manière hiérarchique selon le schéma de la figure 6.2.

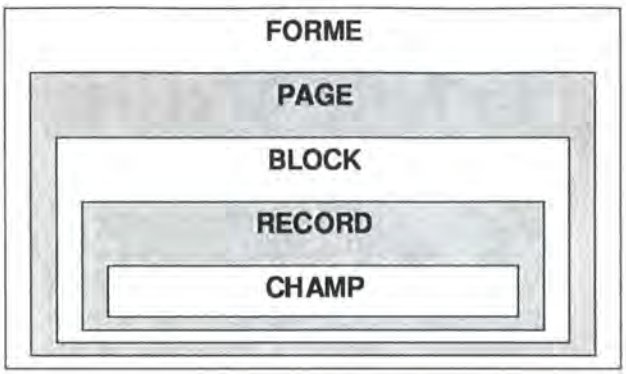

Fig. 6.2.: Hiérarchie des éléments de la forme

Un block peut être composé de plusieurs pages, elles-mêmes composées de plusieurs blocks. A un block, on peut associer un ou plusieurs records affichés simultanément. Ces records sont composés d'un ou plusieurs champs. On peut ainsi retrouver une configuration comme celle représentée à la figure 6.3.

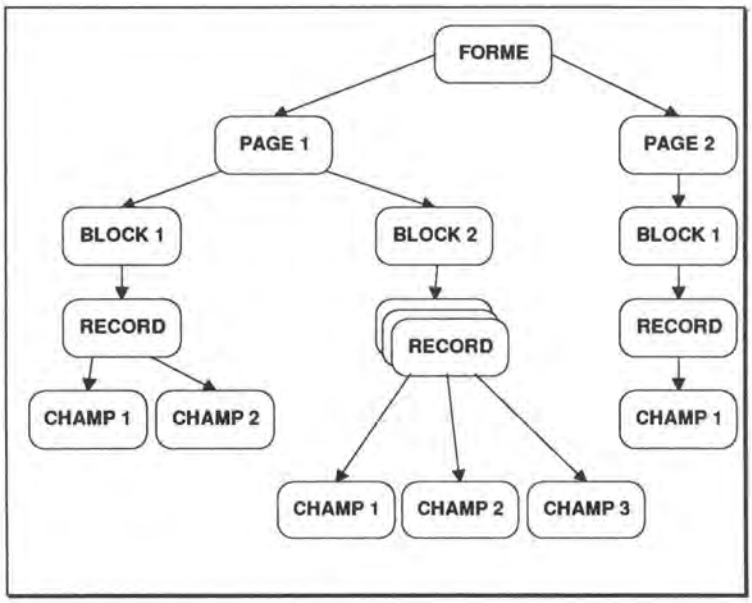

Fig. 6.3.: Hiérarchie des éléments de la forme

La *page* représente ce que l'on appelle communément un "écran". Ainsi, une forme devant permettre la manipulation d'un nombre important de données peut-être subdivisée en plusieurs pages.
Le *block* représente une entité associée à une table de la base de données. Il peut être composé d'un ou de plusieurs records de cette table. On parlera du block "client" associé à la table "client" et qui peut contenir des records de cette même table.
Un *record* correspond tout simplement à un record de la table associée au block dont il fait partie. Un block peut afficher un ou plusieurs records simultanément. Ceci est intéressant pour mieux représenter les liens One-To-Many entre les tables. On pourrait ainsi imaginer une forme composée d'un block client affichant un seul record (correspondant à un client) et d'un block commande affichant simultanément à l'écran plusieurs records (correspondant à un ensemble de commandes passées par le client affiché dans le block client). Les liens entre

les blocks peuvent être définis en fonction des clés primaires et étrangères définies dans la base de données. Cet exemple est repris plus loin à la figure 6.4.
Enfin, un *champ* est, soit un champ du record de la table (dont le type, le format... sont alors directement liés à la définition du champ dans la base), soit un champ calculé (non database field) à partir de champs de la base de données ou à partir de variables externes à la base de données.

Il s'agit donc de définir tous ces éléments et leur hiérarchie, mais également leurs attributs (taille, actions possibles, aide en ligne pour chaque champ...)

Un exemple illustre à la figure 6.4. les différents éléments d'une forme sur l'exemple classique du client et de ses commandes. La figure représente également les liens qui existent entre forme et base de données.

Les champs calculés peuvent l'être à partir de champs de la base (ce qui est le cas du champ Age dans l'exemple de la figure 6.4., qui est calculé à partir du champ Annee_Naiss de la base de données et de la date système), mais aussi à partir d'autres valeurs n'ayant aucune relation avec des informations de la base de données (le numéro de la page courante, par exemple).

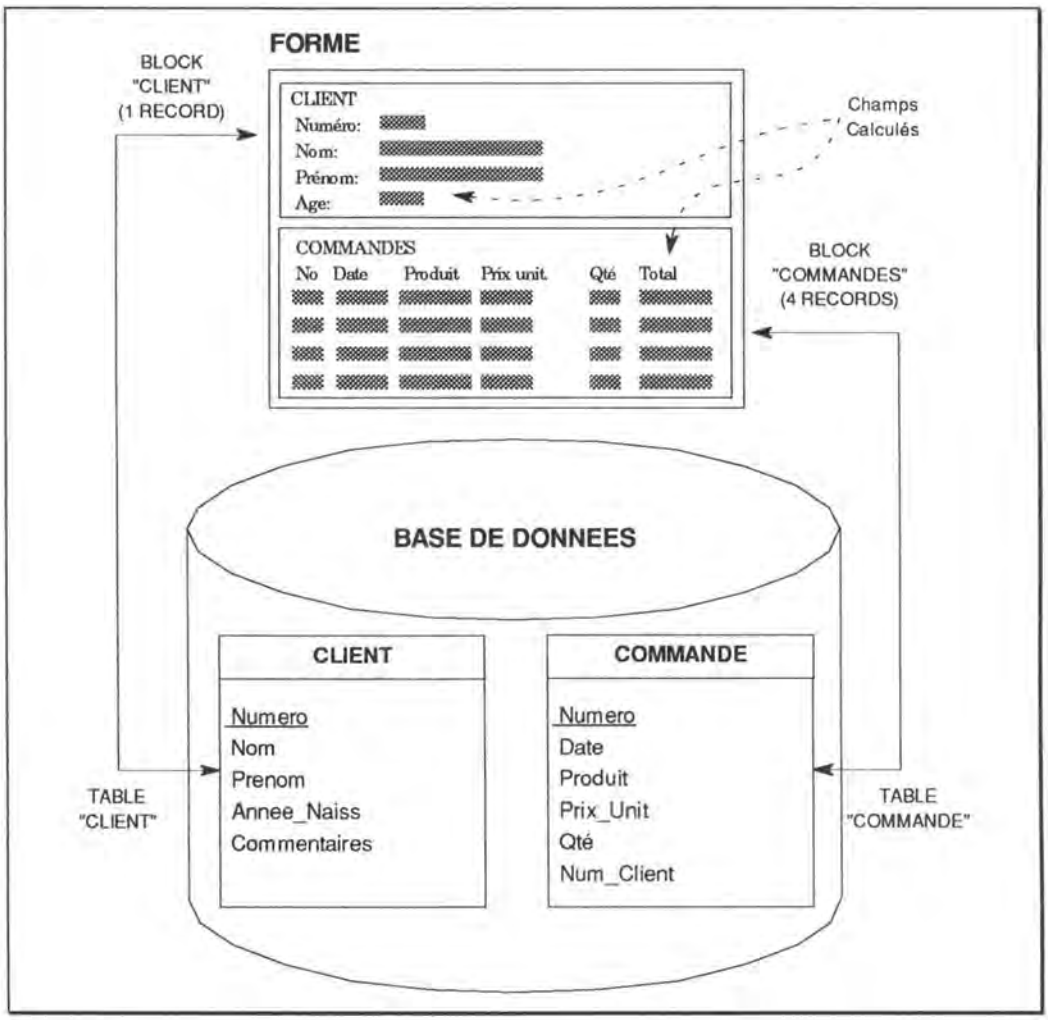

Fig. 6.4.: Les différents éléments de la forme et leur liens avec la base de données

Les formes, vu leurs liens avec la base de données, permettent de manipuler les enregistrements de la base de données. L'utilisateur peut ainsi ajouter, consulter, modifier ou supprimer des enregistrements de la base via les formes.
Toute modification n'est réellement effectuée dans les tables de la base de données que lorsque l'utilisateur les confirme en utilisant la touche COMMIT. Tant que ce n'est pas fait, il est toujours possible pour lui de "revenir en arrière" jusqu'au moment du dernier COMMIT grâce au processus de ROLLBACK.
La sélection d'enregistrements particuliers peut être faite en passant en mode de sélection (QUERY MODE) et en entrant des valeurs de recherche dans les différents champs. On effectue à ce moment là l'exécution de la sélection (EXECUTE QUERY) pour voir à l'écran l' (les) enregistrement(s) correspondant(s) aux critères de recherche.

6.1.3. Programmation événementielle: les événements et les triggers associés

SQL*forms permet de gérer, lors de l'exécution d'une forme, toute une série d'événements: pression d'une touche, action sur la base de données, erreur signalée par Oracle lors de l'accès à la base de données...
Ces événements portent chacun un nom assez représentatif, par exemple:

Key-Insert..... Quand la touche Insert est enfoncée
Key-NxtFld..... Quand une touche permettant d'aller sur le champ suivant est enfoncée
Key-DelRec ..... Quand la touche d'effacement du record courant est enfoncée
On-Insert ..... Quand on insère un nouvel enregistrement
On-Delete ..... Quand on supprime l'enregistrement courant
On-Validate-Field..... Lorsque le curseur quitte un champ
On-Error..... Lorsqu'une erreur est survenue
Post-Commit..... Lorsque l'opération Commit a été effectuée sans erreur

A chaque type d'événement géré par SQL*forms, on peut associer une procédure rédigée en PL/SQL et certains types d'événements ont, d'office, une procédure par défaut qui leur est associée (par exemple, l'événement On-Insert se voit associée la procédure lancant l'instruction d'insertion). Rien n'empêche alors le développeur de définir une procédure plus complexe que celle existant par défaut. Ces procédures sont appelées "triggers" (gachette), ce qui exprime bien l'aspect "événementiel" de l'exécution d'une forme.

PL/SQL est un langage similaire à SQL, mais offrant des possibilités se rapprochant de celles des langages de programmation procéduraux: procédures, boucles, tests... Ce langage est un produit développé par Oracle. Il permet également la gestion de variables fort utiles dans les triggers.

Les triggers peuvent être associés à un block (block level triggers) comme le trigger Post-block, ou bien à un block et à un de ses champs en particulier (field level triggers) comme le trigger Post-fld ou bien rester associés au niveau de la forme (form level triggers).
Ainsi, lorsque l'on veut gérer la frappe d'une touche déterminée (la touche Next Field, par exemple, avec le trigger Key-NxtFld), on peut différencier le cas où le curseur se trouve

dans un block particulier, sur un champ particulier voire quel que soit l'endroit où il se trouve dans la forme. Voici 3 exemples d'utilisation du trigger Key-NxtFld:

Cas	Trigger	Block	Champ
Lorsque la touche Next Field est frappée où que soit le curseur dans la forme	Key-NxtFld	(quelconque)	(quelconque)
Lorsque la touche Next Field est frappée si le curseur se trouve dans le block "Client"	Key-NxtFld	Client	(quelconque)
Lorsque la touche Next Field est frappée si le curseur se trouve sur le champ "Nom" et dans le block "Client"	Key-NxtFld	Client	Nom

La programmmation événementielle telle qu'elle existe dans SQL*forms permet donc une gestion très fine de l'exécution des formes et ainsi de définir des interfaces répondant parfaitement aux besoins du développeur.

Il est également intéressant de remarquer que les touches gérées par SQL*forms portent des noms génériques: Next field, Previous Record, Help, Print, Exit, Edit... Ceci est fait dans le souci de définir un code source indépendant de toute plateforme. Ce n'est que lors de la génération du code exécutable (qui elle, par contre, est dépendante d'une plateforme donnée) qu'une correspondance ("mapping") est faite entre touche générique et touche réelle du clavier de la plateforme visée. D'ailleurs, lors de l'éxécution de la forme, l'utilisateur peut, à tout moment, appeler un écran d'aide affichant le tableau correspondance.
Par exemple, la touche générique Commit est la touche DO sur un clavier de terminal VT, la touche F10 sur un PC...

6.1.4. Procédures de niveau général (form level procedures)

Il est enfin possible de définir des procédures PL/SQL de niveau général (non associées à un événement particulier). Ces procédures assurent souvent des fonctions qui doivent être accessibles à tout moment et depuis n'importe quel point de la forme.
Par exemple, une procédure de niveau général pourrait servir à effectuer une mise à jour globale des variables intervenant dans les triggers.
Une forme peut même utiliser des procédures de niveau général définies dans une autre forme. Ce mécanisme de référence permet d'éviter la redondance de code et a été utilisé à bon escient dans le projet LANDB (cfr couche commune de procédure - section 6.4.1).

La figure 6.5. représente l'intégration du mécanisme de programmation événementielle et des procédures de niveau général. Les flèches y représentent les appels faits aux procédures de niveau général et entre elles.

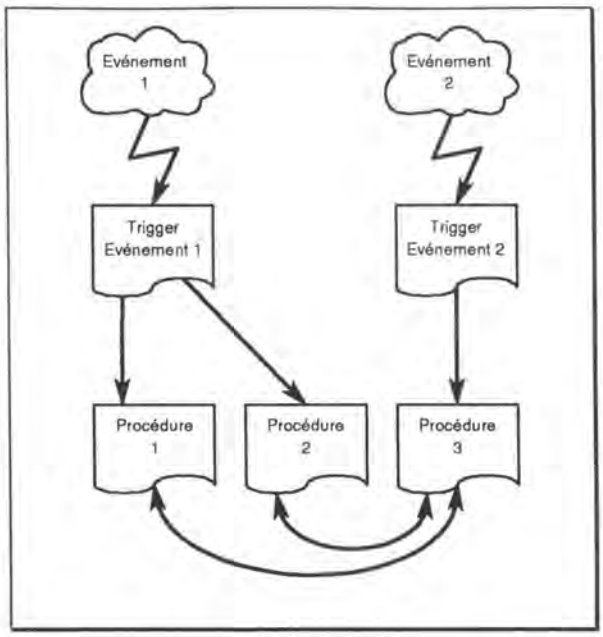

*Fig. 6.5.: la programmation événementielle avec SQL*forms*

6.2. Présentation générale des interfaces de gestion de l 'adressage[1]

Les interfaces de gestion de l'adressage ont été développées pour les gestionnaires TCP/IP et DECnet qui utilisaient déjà l'application CSNET. On peut donc qualifier les utilisateurs d'experts.
Le développement de ces interfaces a été effectué dans le cadre d'un stage de fin d'études au CERN. Une enquête a été effectuée auprès de leurs premiers utilisateurs au moment de la rédaction de ce mémoire. On pourra trouver à l'annexe 3 le formulaire d'évaluation des interfaces ainsi qu'une réponse parvenue au moment de l'impression de ce mémoire.

6.2.1. Une forme principale

Les interfaces de gestion de l'adressage des machines s'articulent autour d'une forme principale appelée "Network Device". Depuis cette forme, l'utilisateur peut consulter les informations générales sur les machines du réseau: nom, type, système d'exploitation, localisation, responsable, utilisateur principal, groupe gestionnaire... En plus de ces informations, l'utilisateur peut consulter, sur la même forme, l'essentiel des informations sur les adresses TCP/IP, DECnet version 4 et Hardware de la machine. Cette forme est donc composée de 4 blocks correspondant aux tables NETWORK_DEVICE, TCPIP_ADDRESS, DECNET_ADDRESS et HARDWARE_ADDRESS. Elle est reprise à la figure 6.6. Le block maître est bien évidemment NETWORK_DEVICE, les trois autres blocks étant blocks esclaves de ce dernier.

---

[1] Le lecteur qui désire avoir une information complète sur le fonctionnement des interfaces de gestion de l'adressage peut se rapporter au manuel d'utilisation intitulé "Landb Database Network - Management Screen - User Guide" repris en annexe.

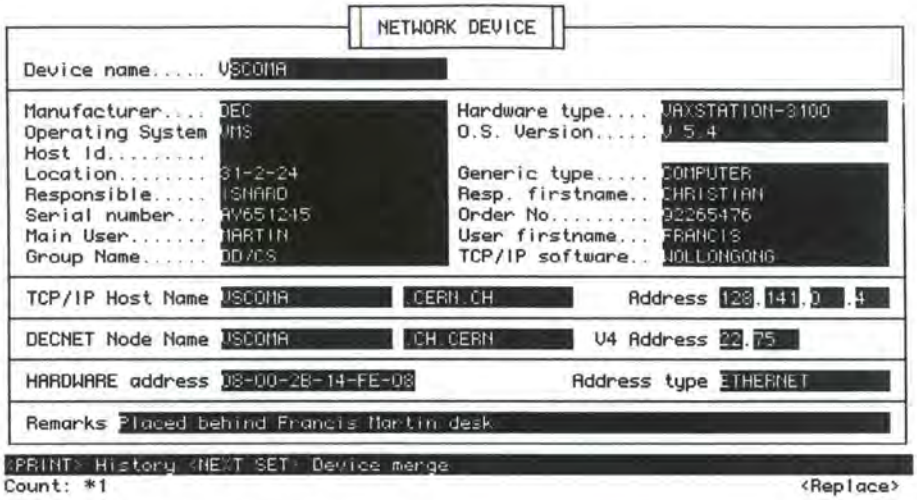

Fig. 6.6.: L'écran principal de gestion des adresses: Network Device

Cette forme permet la consultation, la modification, l'insertion et la suppression à la fois de machines, mais aussi de leurs adresses TCP/IP, DECnet version 4 et Hardware. Plusieurs outils, que nous détaillerons ci-après, sont mis à la disposition de l'utilisateur afin de lui permettre d'effectuer un maximum d'opérations liées à sa tâche depuis cette seule forme: listes de valeurs, valeurs par défaut...

6.2.2. Des formes spécialisées

Malgré cela, l'utilisateur qui désirerait avoir plus d'information peut accéder depuis la forme Network Device à des formes spécialisées lui donnant toute l'information sur les adresses, les alias, les areas TCP/IP, les réseaux TCP/IP, les historiques... La figure 6.7. représente l'écran des adresses TCP/IP.
Ces formes lui permettent également de consulter, modifier, insérer ou supprimer de l'information des tables associées à ces formes spécialisées qui sont attachées à une table en particulier.

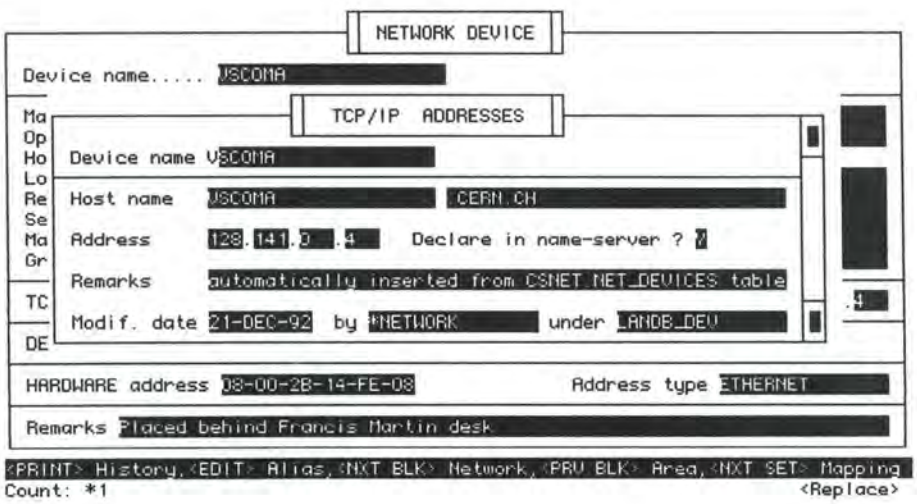

Fig. 6.7.: Un exemple de forme spécialisée: les adresses TCP/IP (par dessus la forme Network Device)

6.2.3. Remarques spécifiques sur le problème de transition de la version 4 à la version 5 de DECnet

Comme nous l'avons déjà signalé lors de la présentation de la base de données, le protocole DECnet est en période de transition de la version 4 à la version 5. Ce passage étant planifié sur plusieurs années et le CERN gérant des machines utilisant les deux versions (les DEC Alpha fonctionnent déjà avec la version 5 alors que les VAX ne devraient y venir que plus tard), il est nécessaire de gérer parallèlement les deux versions.
La solution prise au CERN au moment de la rédaction de ce mémoire était de considérer les adresses de la version 5 comme un miroir parfait des adresses de la version 4. Ainsi, on ne gérait que les adresses de la version 4 et toute opération sur ces dernières donnait lieu à la même opération dans les tables de la version 5. Les tables des adresses version 5 sont donc *esclaves* des tables de la version 4. La figure 6.8. illustre le principe de miroir des versions 4 et 5.

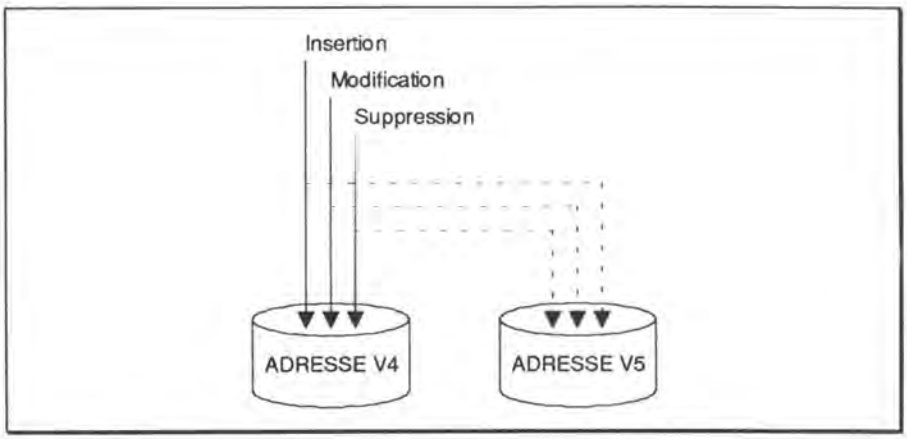

Fig. 6.8.: Principe de miroir entre les versions 4 et 5 de DECnet

Les adresses de la version 5 n'ayant pas le même format que les adresses de la version 4, il est nécessaire de calculer l'adresse version 5 à partir des valeurs de la version 4 au travers d'un algorithme de transformation comme représenté à la figure 6.9.

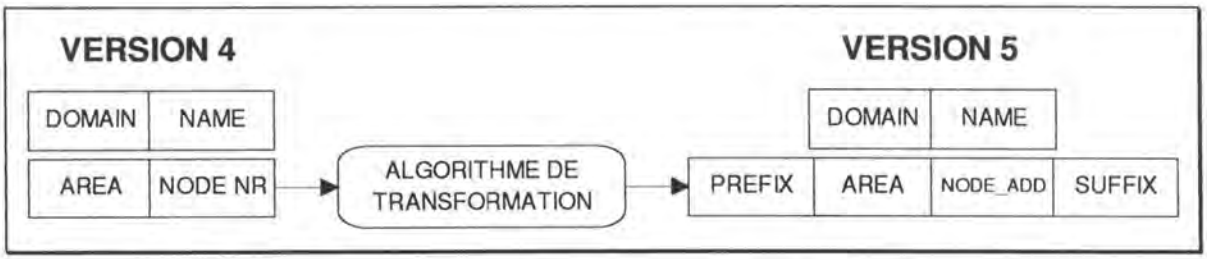

Fig. 6.9.: L'algorithme de transformation d'une adresse V4 en une adresse V5

Les nouvelles propriétés offertes par la version 5 (alias, noms multi-adresses...) ne seront pas utilisées tant que cette nouvelle version sera le miroir de la version 4. Ainsi, malgré qu'il soit possible d'associer plusieurs adresses à un seul nom dans la version 5, tant que le miroir sera assuré, une seule adresse de la version 5 sera associée à un seul nom. De même, aucun alias ne sera défini.

Cette solution explique qu'on ne puisse modifier que les adresses de la version 4 (seule visible, d'ailleurs, dans la forme Network Device). Il est néanmoins possible de consulter les adresses version 5 sans aucune possibilité d'insertion, de modification ou de suppression.

6.3. Modularité

La modularité des interfaces est nécessaire pour deux raisons: l'hétérogénéité de la population des utilisateurs et l'adaptabilité nécessaire aux changements intervenant dans la structure de la base et dans la définition des tâches des utilisateurs.
Nous allons maintenant voir les principes de développement de formes permettant la plus grande modularité possible.

6.3.1. Définition des formes: le principe utopique "une forme par table"

Comme nous l'avons déjà signalé, il a été décidé d'implémenter la modularité des interfaces en créant de préférence un maximum de formes utilisables par plusieurs utilisateurs de façon à minimiser le travail lors de changements au niveau de la structure de la base de données. Si on applique ce principe jusqu'au bout, l'idéal serait de créer une forme par table ni plus ni moins. Ainsi, toute modification à la structure des données n'impliquerait que des modifications dans LA forme associée à chaque type de données ayant subi une modification.
Ainsi, il y aurait une forme pour les machines du réseau, une forme pour les adresses TCP/IP, une forme pour les alias TCP/IP, une forme pour chacun de leurs historiques...
Le principe "une forme par table" est représenté à la figure 6.10.

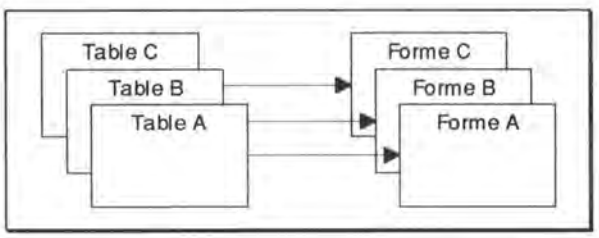

Fig. 6.10.: Le principe "Une forme par table"

En suivant le principe "Une forme par table", on est à l'extrême opposé du travers de l'application CSNET, à savoir, quatre écrans étroitement associés à une tâche particulière et extrêmement difficiles à réutiliser ou à modifier en cas de modification de la structure de la base ou de la tâche des utilisateurs, car intégrant des données provenant de toute la table.

Mais le principe "Une forme par table" est utopique. En effet, il fait table rase du principe communément admis (voir pour cela la plupart des ouvrages concernant le développement d'interfaces homme-machine dont [Bro88], [Rav89] et [Shn87]) selon lequel c'est plus la définition de la tâche des utilisateurs qui détermine la conception des interfaces proprement dites que la structure des données manipulées. En effet, l'utilisateur ne doit pas voir la manière de résoudre sa tâche dépendre de la structure sous-jacentes des données qu'il manipule. Cette structure est probablement différente voire complètement étrangère à la

structuration du réel qu'il perçoit. Ce problème est un exemple concret du modèle des "deux golfes" présentés par D.A. Norman [Bod88].

Or, si on se limite à la règle "Une forme par table", le design des interfaces est uniquement dépendant de la structure de la base de données.
Il est donc nécessaire de ne pas perdre de vue l'importance de la définition de la tâche des utilisateurs. Ainsi, certaines formes doivent éventuellement combiner des données de plusieurs tables en vue de rencontrer les attentes des utilisateurs au risque de demander un travail plus fourni en cas de modification de la structure de la base de données.

Afin de prendre en compte ces deux aspects, il a été décidé de limiter les formes associées à une tâche bien précise à un nombre minimum et aux cas ne pouvant s'en passer et d'essayer d'utiliser au maximum les formes "simples". Le principe est ainsi revu sans être tout à fait abandonné. Il est illustré à la figure 6.11.

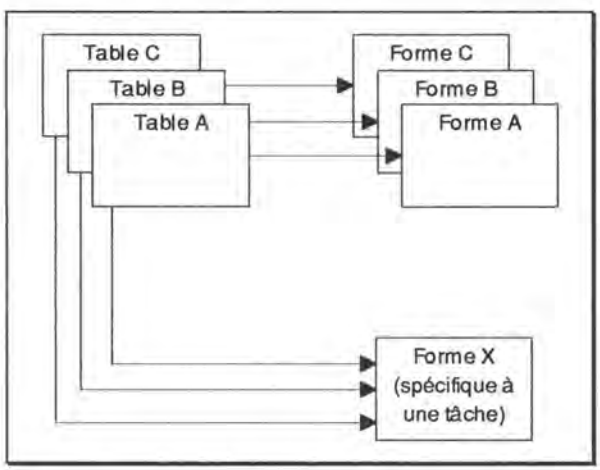

Fig. 6.11.: Le principe "Une forme par table" revu

Dans le cas précis de la gestion de l'adressage des machines, les formes suivantes ont été définies[2]:

Forme	*Table(s) associée(s)*	*Description*
NETDEV_C	Network_Device Decnet_Addres TCPIP_Address Hardware_Address	Forme récapitulative affichant les informations sur les machines et l'essentiel de leurs adresses DECnet, TCP/IP et Hardware.
NETDEV_H	Network_Device_History	Forme affichant l'historique des machines
IPADD_C	TCPIP_Address	Forme affichant la totalité des informations sur les adresses TCP/IP
IPADD_H	TCPIP_Address_History	Forme affichant l'historique des adresses TCP/IP

[2] Le suffixe C (pour "Common") signale que la forme est définie pour une utilisation standard, c'est-à-dire consultation et modification de tables standards (pas historiques, pas listes des valeurs...). Le suffixe H (pour "History") spécifie que la forme est associée à une table d'historique.

Forme	*Table(s) associée(s)*	*Description*
IPALI_C	TCPIP_Alias	Forme affichant la totalité des informations sur les alias TCP/IP
IPALI_H	TCPIP_Alias_History	Forme affichant l'historique des alias TCP/IP
DECADD_C	Decnet_Address Decnet_V5_Address	Forme affichant la totalité des informations sur les adresses DECnet version 4 et 5 (l'adresse V5 étant le miroir de la V4)
DECADD_H	Decnet_Address_History	Forme affichant l'historique des adresses DECnet version 4
DEV5AD_H	Decnet_V5_Address_History	Forme affichant l'historique des adresses DECnet version 5 (sans intérêt pour le moment car miroir de DECADD_H)
DEV5NA_C	Decnet_V5_Name Decnet_V5_Alias	Forme affichant la totalité des informations sur les noms DECnet version 5 et leurs alias (pas d'alias pour le moment)
DEV5AL_H	Decnet_V5_Alias_History	Forme affichant l'historique des alias DECnet version 5
HWADD_C	Hardware_Address	Forme affichant la totalité des informations sur les adresses hardware
HWADD_H	Hardware_Address_History	Forme affichant l'historique des adresses hardware

Comme on peut le constater dans le tableau ci-dessus, le nombre et le nom des formes suit de très près la structure de la base de données à l'exception de la forme NETDEV_C étant donné son rôle de forme *récapitulative*. Tout en rencontrant les attentes de l'utilisateur en lui proposant un écran récapitulatif (NETDEV_C) propre à l'aider à mener sa tâche à bien, on a également réussi à respecter la règle "Une forme par table" pour la majorité des formes. Ainsi une modification d'une quelconque table de la base (apparition d'un nouveau protocole, modification d'un protocole existant...) demandera un minimum de travail: modification de LA forme associée et de la forme NETDEV_C.

NETDEV_C affiche à la fois des informations générales sur les machines (et donc une partie seulement des champs de la table Network_Device) mais également l'essentiel des informations sur les adresses TCP/IP (table TCPIP_Address), DECnet (Decnet_Address) et Hardware. Cette forme est destinée aux utilisateurs chargés de la gestion des parcs d'adresses TCP/IP et DECnet et ne serait pas très intéressante pour un utilisateur ne désirant que les informations sur les machines ou sur les adresses. Elle est littéralement "truffée" d'outils aidant les gestionnaires TCP/IP et DECnet dans leurs tâches de tous les jours. Elle a fait l'objet de l'attention des développeurs en parallèle avec les remarques des futurs utilisateurs. Cette forme suit donc bien les principes de développement d'interfaces définis dans la littérature.

On peut concrètement justifier la philosophie de modularisation des interfaces visant à limiter le nombre de formes comme NETDEV_C en sachant que le développement de cette dernière a pris 6 mois à un développeur alors que les formes IPADD_C, DECADD_C... demandaient chacune 1 seule semaine de travail à la même personne.

Afin d'éviter d'avoir trop de formes spécifiques à une tâche et lourdes en développement, le développeur peut essayer de combiner les appels entre plusieurs formes élémentaires (associées à une table de la base de données) en vue d'obtenir une interface répondant au mieux aux attentes de l'utilisateur.

6.3.2. Appels entre formes: répercussion des clés primaires

Comme on vient de le voir, les interfaces utilisateur sont composées d'une multitude de formes assez petites en tailles et de quelques unes plus importantes liées à des tâches particulières. Il est donc nécessaire de pouvoir passer des unes aux autres de manière aisée.
SQL*forms permet aux formes de s'appeller les unes les autres en fonction des besoins. Ainsi, à partir de la forme concernant les machines, il est possible d'appeler la forme concernant les adresses TCP/IP ou la forme concernant les adresses DECnet.
Lorsqu'un utilisateur quitte une forme, on revient à la forme appelante si celle qu'on quitte était appelée (sinon, on quitte l'application). Les appels entre formes se font donc toujours de manière *séquentielle*. C'est SQL*forms qui gère la liste des appels effectués à un moment donné.
Les possibilités d'appel entre les formes sont définies par le développeur qui détermine une touche d'appel pour chaque forme "appelable". Il n'est donc nécessaire de définir les appels que dans le sens appelant-appelé, mais pas dans le sens retour, ce qui limite le travail du développeur.
Les possibilités d'appel entre les différentes formes que nous avons définies pour la gestion des adresses sont schématisées à la figure 6.12. Les flèches représentent la possibilité pour une forme (à l'origine de la flèche) d'en appeler une autre (au bout de la flèche).

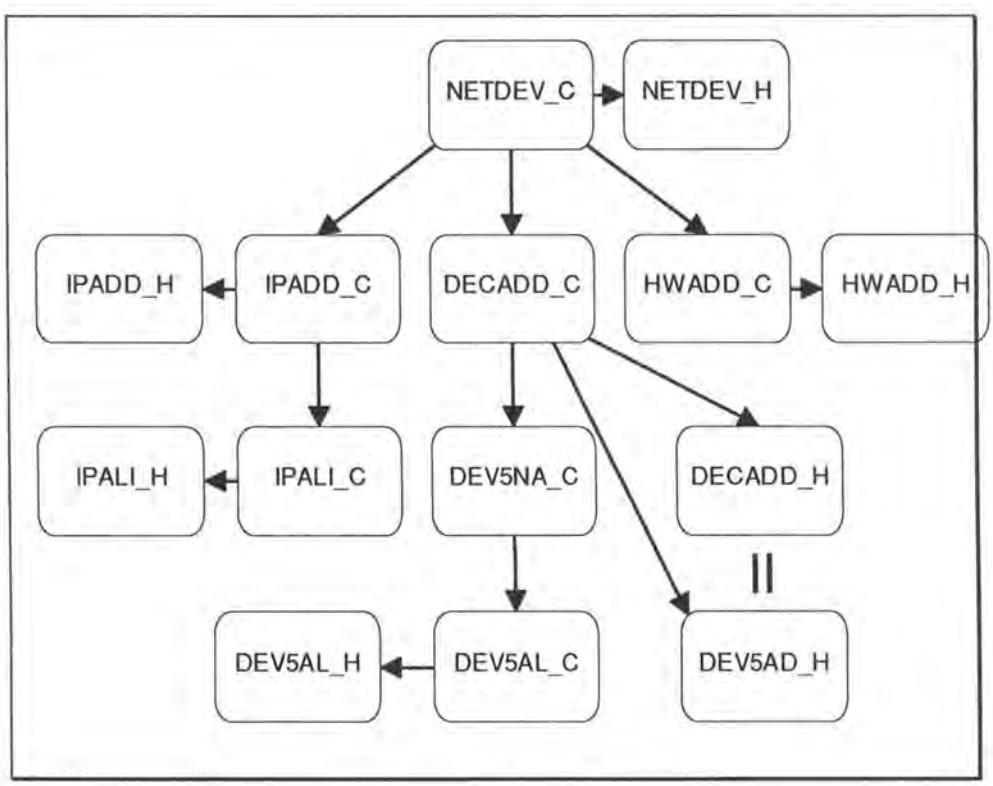

Fig. 6.12: Organisation des appels des formes gérant l'adressage des machines

La question se pose alors de savoir si le passage entre les différentes formes est cohérent au niveau de l'information affichée. En effet, imaginons que l'utilisateur consulte les informations sur la machine M dans la forme NETDEV_C. S'il appelle la forme DECADD_C, on peut supposer qu'il désire connaître la totalité des informations sur les adresses DECnet de la machine M. Le contraire est également vrai: si l'utilisateur effectue alors la sélection d'une adresse DECnet A depuis DECADD_C, lorsqu'il revient vers NETDEV_C, on peut supposer qu'il désire voir affichées les informations concernant la machine dont A est l'adresse. Cet exemple est illustré à la figure 6.13.
Ce principe est celui qu'on appellera de la "répercussion de l'information".

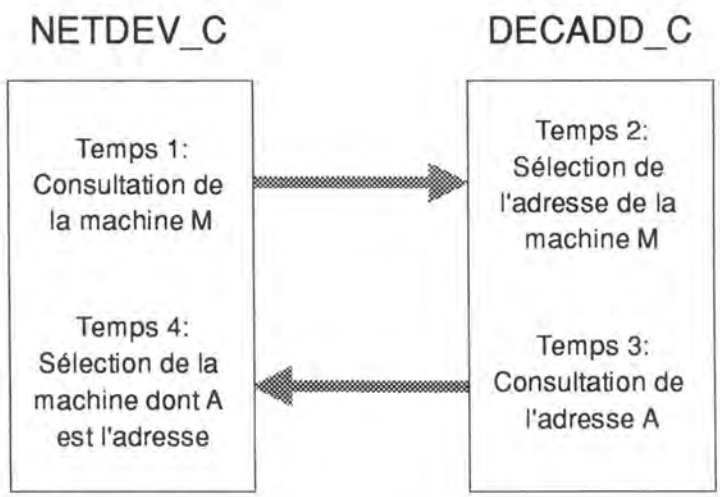

Fig. 6.13.: Un exemple de répercussion d'information entre les formes

Le problème est de savoir comment, d'une forme à l'autre, spécifier quel enregistrement de la table associée on désire consulter lorsqu'on "entre" dans une forme, et cela, sans limiter ni spécifier une fois pour toutes les possibilités d'appels entre les formes.

La solution adoptée a le mérite d'être assez transparente. SQL*Forms 3.0 permet la manipulation de variables. Il suffit dès lors de maintenir une variable par champ participant à n'importe quelle clé primaire d'une table. Dans notre exemple de la gestion des adresses, nous avons comme variables:

Table	***Champs de la clé primaire donnant lieu à une variable***
Network_Device	Usual_Name
Network_Device_History	Usual_Name
TCPIP_Address	Name Domain
TCPIP_Address_History	Name Domain
TCPIP_Alias	Alias
TCPIP_Alias_History	Alias
Decnet_Address	Name Domain

Table	*Champs de la clé primaire donnant lieu à une variable*
Decnet_Address_History	Name Domain
Decnet_V5_Address	Name Domain
Decnet_V5_Address_History	Name Domain
Decnet_V5_Name	Name Domain
Decnet_V5_Alias	Alias
Decnet_V5_Alias_History	Alias
Hardware_Address	Hardware_Address
Hardware_Address_History	Hardware_Address

Lorsqu'une forme en appelle une autre, il suffit de mettre à jour les variables correspondant à la clé primaire de la table de la forme appelée et elle seule (les autres variables sont donc non instanciées). La forme appelée dès qu'elle est ouverte vérifie la valeur de la clé primaire de *sa* table principale et sélectionne l'enregistrement correspondant. Ce principe est illustré à la figure 6.14.

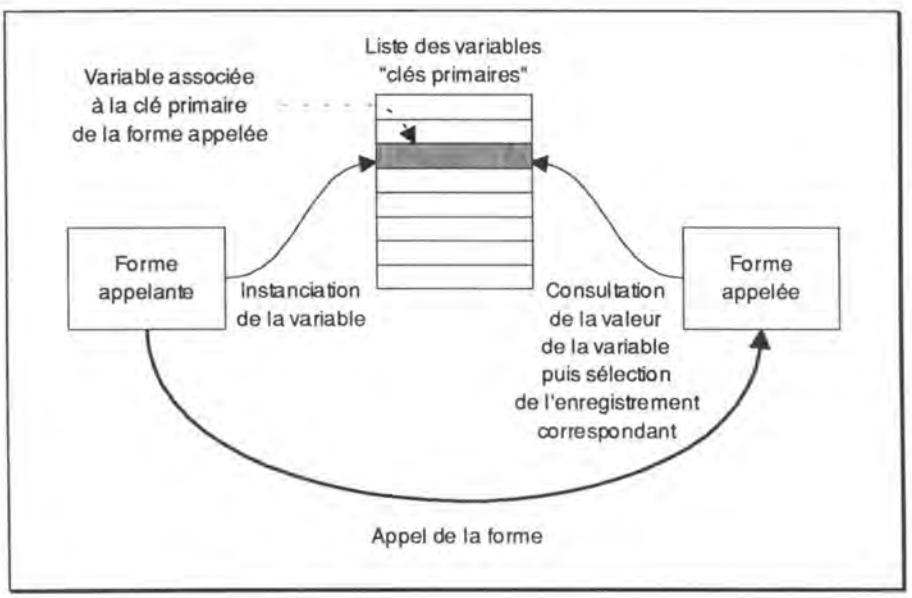

Fig. 6.14.: Mise à jour puis consultation des variables "clés primaires" à l'appel

Lorsqu'on quitte une forme, par contre, ne pouvant savoir vers quelle forme on retourne (c'est en effet SQL*forms qui gère la liste des appels), on instancie toutes les variables dont on connaît une valeur. La forme appelante ayant à nouveau la main vérifiera si les variables correspondant à sa clé primaire sont instanciées et sélectionnera alors l'enregistrement correspondant. Ce principe est repris à la figure 6.15.

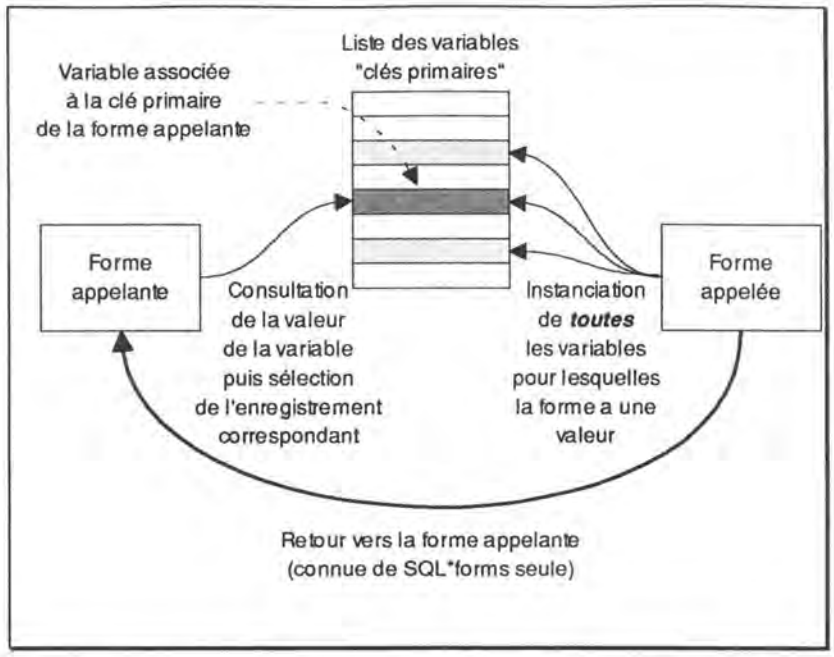

Fig. 6.15.: Mise à jour puis consultation des variables "clés primaires" au retour

Ce principe a l'avantage de ne pas compliquer le système d'appel entre les formes. En effet, on devait déjà définir pour chaque forme, quelles autres formes étaient appelables depuis elle. Il suffit donc de signaler quelles variables globales doivent être mises à jour. Au retour de la forme appelée, il suffit de vérifier si les variables globales correspondant à la clé primaire de la table principale de la forme a été modifiée (dans ce cas, il faut resélectionner l'enregistrement affiché par la forme) ou non.

6.3.3. Affichage

La multiplicité de formes appelables par un utilisateur peut aussi créer des problèmes de cohérence d'affichage à l'écran.
Comme nous l'avons vu dans la section 6.1.1., l'outil SQL*Forms 3.0 permet de faire afficher les formes sur tout ou partie de l'écran à la manière d'un environnement graphique de type fenêtré. Cette propriété sera très utile pour afficher les différentes formes au fur et à mesure de leurs appels. Si l'utilisateur désire les renseignements sur les adresses DECnet d'une machine qu'il consulte depuis NETDEV_C, il appelle la forme DECADD_C, celle-ci s'affiche par dessus la forme NETDEV_C. Lorsqu'il quitte DECADD_C, elle disparaît de l'écran. SQL*Forms remet alors l'écran dans le même état qu'avant l'appel à DECADD_C.
Le parallèle avec les environnements multi-fenêtrés s'arrête là... Il est, en effet, impossible de déplacer une forme sur l'écran, sa position étant définie lors de sa création, de même qu'il est impossible pour l'utilisateur de modifier la taille d'une forme (pour voir un élément se trouvant derrière par exemple). Dès lors, le développeur de l'interface doit être particulièrement attentif à la manière dont il dispose les formes s'il désire que certaines informations soient toujours visibles.

6.4. Sécurité et contrôle

Le nombre d'utilisateurs étant très élevé et leurs profils tellement différents, il est nécessaire que les interfaces assurent au mieux le contrôle des données entrées par ces derniers et assurent la sécurité pour la récupération des erreurs. Elles doivent également veiller à détecter tout conflit[3]: modification de données entrées par un autre utilisateur...

6.4.1. Couche commune de procédures

Une particularité de l'application LANDB est la présence d'une couche commune de procédures de niveau général écrites en PL/SQL se situant entre les formes et la base de données. Elle sert de filtre entre ces deux points et permet d'assurer une plus grande sécurité et une vérification constante des accès à la base. Ainsi, aucune forme n'accède directement à la base au moyen d'instructions SQL, mais fait plutôt appel à une procédure particulière.
En règle générale, il y a trois procédures par table de la base:

- on_insert_<*nom de la table*>
- on_update_<*nom de la table*>
- on_delete_<*nom de la table*>

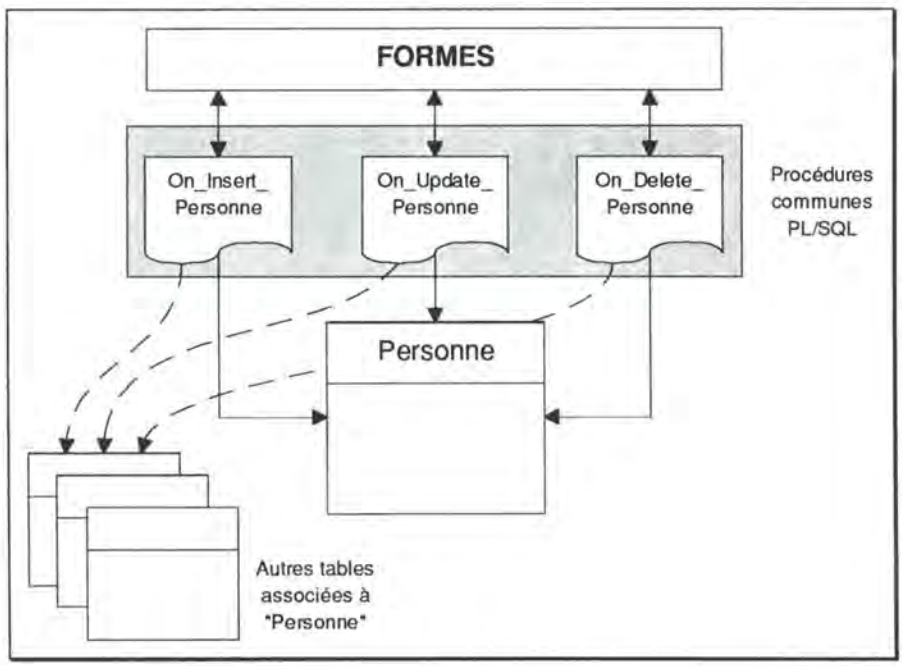

Fig. 6.16.: Le principe de l'accès vérifié à la base de données par la couche commune de procédures

Ces procédures effectuent toutes les vérifications nécessaires avant d'accéder à la base (ce mécanisme est représenté à la figure 6.16.):

- droit d'insertion, de modification ou de suppression de l'utilisateur sur la table concernée
- vérification que l'on ne va pas se trouver dans une situation incohérente

[3] Nous ne parlons pas ici de conflit d'accès simultané à la base de données. Ce problème est, en effet, géré par le SGBD Oracle.

- vérification de la validité des données (format, orthographes, existence des clés étrangères...)

Si toutes les conditions sont réunies, elles accèdent ensuite à la base en y apportant toutes les modifications nécessaires:

- modification de la table concernée par la procédure
- modification de toute autre table associée (historiques, tables associées par contrainte référentielle...)

Les procédures communes renvoient à la forme appelant un ou des messages appropriés: information que l'opération s'est bien passée, erreurs survenues...

Les situations incohérentes que pourrait détecter la couche commune de procédures sont principalement dues aux contraintes référentielles.

Par exemple, on ne peut supprimer un enregistrement de la table Network_Device si cet enregistrement est toujours référencé dans un enregistrement de la table TCPIP_Address. Le SGBD Oracle interdirait d'ailleurs cette opération étant donné la contrainte référentielle existant entre les deux tables.
L'avantage de la couche de procédure est qu'elle va gérer ce type de problèmes avant Oracle, détecter dans quel cas on se trouve précisément et peut être régler le problème sans intervention de l'utilisateur.
Reprenons l'exemple de la machine que l'on désire effacer alors qu'elle a toujours une adresse TCP/IP. La procédure On_Delete_Network_Device voyant que cette adresse existe toujours pourrait faire effacer l'adresse puis la machine afin qu'Oracle accepte l'opération de suppression de la machine. Un autre choix aurait pu être d'envoyer un message à l'utilisateur signalant que la suppression de la machine ne peut être faite car elle a toujours une adresse TCP/IP.
Ainsi, le développeur de l'application peut, selon les cas, faire effectuer des opérations en cascade sans en prévenir l'utilisateur ou bien annuler une opération en cours et prévenir l'utilisateur de la situation.

Dans la plupart des cas, LANDB signale à l'utilisateur les problèmes de cohérence dans la base et n'effectue aucune opération. L'utilisateur a alors le choix de modifier lui-même la situation afin de la débarasser de son incohérence.
Dans notre exemple, l'utilisateur désire supprimer une machine qui est toujours référencée dans la table des adresses TCP/IP.
Dans une première hypothèse, la couche commune de procédure, constatant cette situation, la signalera à l'utilisateur par un message du style: "Attention: impossible de supprimer la machine *<nom de la machine>*, car elle est toujours associée à l'adresse TCP/IP *<adresse TCP/IP>*". L'utilisateur sera alors libre de supprimer l'adresses TCP/IP puis seulement la machine.
Dans la seconde hypothèse, par contre, la couche commune de procédure se chargera de supprimer elle-même l'adresse TCP/IP puis la machine et signalera à l'utilisateur "Machine *<nom de la machine>* supprimée".

Cette couche commune de procédures sert donc de point d'entrée obligatoire pour toutes les formes et assure la cohérence de la base. Elle a été également concue dans l'optique de

l'évolution du SGBD Oracle qui, dans sa prochaine version (version 7), permettra d'associer des méthodes aux tables de la base, un premier pas vers une base de données orientée objet. On pourra ainsi associer toutes les procédures concernant une table à celle-ci. Le grand avantage de cette évolution est, qu'actuellement, un client peut toujours outrepasser l'utilisation de la couche commune de procédures et modifier directement la base de données à partir de SQL. Cela est rendu possible par le fait que la couche de procédures est localisée au niveau du client. Avec la version 7, par contre, cette couche commune de procédure se trouvant sur le serveur, il ne sera plus possible de ne pas en tenir compte. Ce mécanisme est illustré à la figure 6.17.

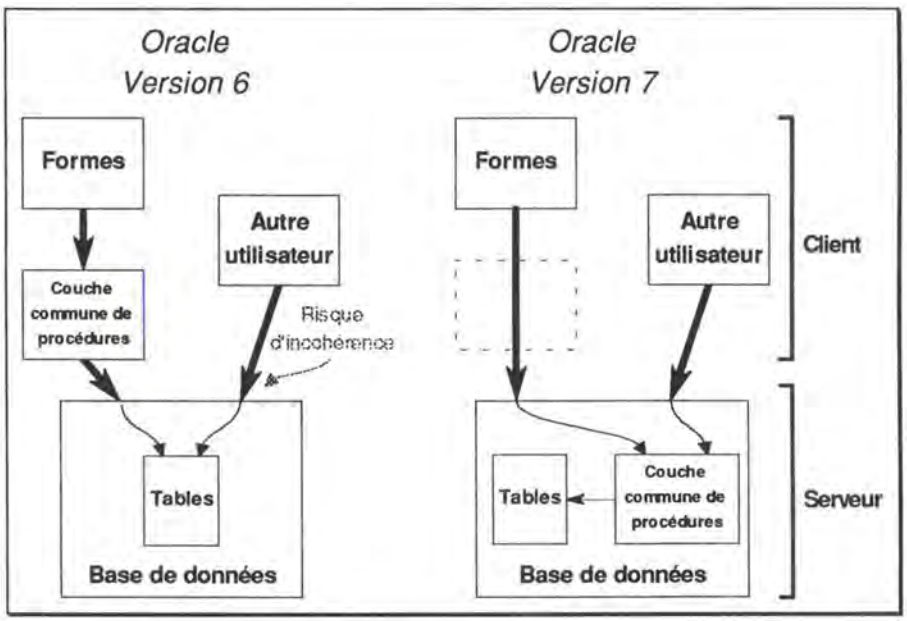

Fig. 6.17: Evolution entre les versions 6 et 7 de Oracle

Un autre avantage de la couche commune de procédure est de maintenir une certaine indépendance entre le dialogue et l'application elle-même. Ce principe, énoncé dans [Sac91] est essentiel pour permettre une évolution aisée et indépendante tant de la base de données que des interfaces.

Actuellement, la couche commune de procédures représente plus de 20.000 lignes de codes PL/SQL et s'intègre dans une forme n'ayant pour but que la définition de ces procédures: GLOBAL_F. Les autres formes utilisent alors la propriété de référence de procédures pour pouvoir appeler les procédures de cette forme GLOBAL_F.

6.4.2. Attributs de saisie

La technique la plus simple pour la vérification du format des données est la définition d'attributs de saisie. Cette technique existe sur la plupart des outils de développement d'interfaces; SQL*Forms la supporte également.
Plusieurs attributs peuvent être associés à chaque champ de la forme.

- Type du champ — Entier, chaîne de caractère, date (avec vérification)
- Longueur du champ — Nombre de caractères du champ
- Domaine de valeur — Valeurs minimale et maximale du champ

- Masque de saisie — Format alphanumérique (chiffres, majuscules, etc.) des données
- Attributs de modification — Caractère modifiable, affichable, sélectionnable, etc. du champ
- Valeur par défaut — Valeur la plus souvent rencontrée

Ce principe d'interface étant très répandu, nous ne nous étendrons pas plus sur le sujet.

6.4.3. Listes de valeurs

Si on désire retrouver un enregistrement sur base d'un nom, si on veut faire des statistiques ou des modifications à grande échelle, il est nécessaire de contrôler, plus encore que le format, la valeur de certaines données.
Prenons par exemple le cas des noms de types de machines (Device_Type). Comme nous l'avons déjà signalé, la base CSNET regorgeait de données redondantes en ce qui concerne les noms de types de machines. La station de travail DECstation 5000/120 pouvait se trouver référencée sous toutes les orthographes imaginables: DEC Station 5000-120, DecStation 5000 120, DECSTATION/5000/120, DEC 5000/120... L'utilisateur qui voulait alors sélectionner toutes les machines de ce type devait tenter de retrouver toutes les orthographes sans être certain d'en avoir fait le tour. Le cas du personnel est encore plus marqué par ce problème.
Pour cette raison, il a été décidé de mettre en place un système de listes de valeurs déjà introduit dans la description de la base de données.
Le principe des listes de valeurs est de proposer à l'utilisateur une liste des valeurs correctes. Lorsqu'il entre une donnée dans une forme, il la choisit dans la liste de valeurs et est ainsi certain de son format et de son orthographe. Un exemple de liste de valeurs est donné à la figure 6.18.

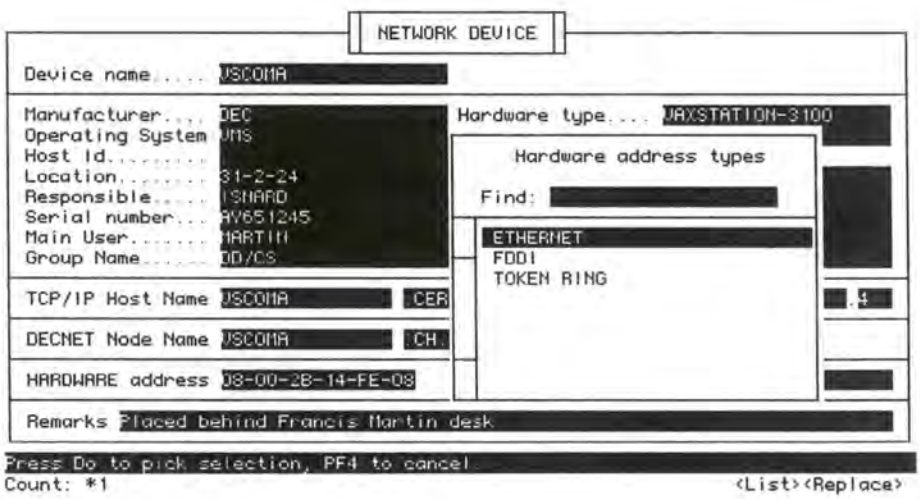

Fig. 6.18.: Un exemple de liste de valeurs: les types d'adresses hardware

L'utilisation des listes de valeurs est, en général, optionnelle, mais lors de l'introduction ou de la modification de la donnée dans la base de données, une vérification est faite par la couche commune de procédure (détaillée plus haut). Si la valeur est incorrecte, un message est donné à l'utilisateur et l'insertion ou la modification est annulée. En fait, l'utilisateur a

toujours intérêt à employer les listes de valeurs sauf s'il est absolument certain de l'orthographe de la valeur.
Le tableau suivant reprend l'ensemble des listes de valeurs et les tables associées en ce qui concerne la gestion des parc d'adresses:

Type de données	Table(s) associé(e)
Système d'exploitation	Operating_System
Type de machine	Device_Type Generic_Type
Information sur le personnel	Personal_Information
Groupes CERN	Logical_Group
Type d'adresse hardware	Hardware_Address_Type

Chaque liste de valeur contient une valeur "unknown" permettant à l'utilisateur de ne pas être bloqué si la valeur qu'il recherche ne se trouve pas dans la liste.

Seul un petit nombre d'utilisateurs a le droit d'accéder aux tables associées aux listes de valeurs. L'accès limité pour la modification des listes de valeurs est le garant d'une liste correcte, intérêt de ce système.

6.4.4. Listes de valeurs privées - communes

Une caractéristique a été ajoutée aux listes de valeurs trop longues pour une manipualtion aisée, à savoir les listes de types de machines et de systèmes d'exploitation dans le cadre de la gestion d'adresses.
Pour chacune de ces listes, chaque compte Oracle utilisateur peut définir une liste privée. La liste privée est un sous-ensemble de la liste de valeur complète que l'on appelle alors *liste commune* puisque *tous* les utilisateurs peuvent aller chercher les valeurs de leur liste privée dans celle-ci.
Il y a donc autant de listes privées qu'il y a de comptes Oracle.
A tout moment, l'utilisateur peut ajouter ou supprimer des éléments de sa liste privée. Il ne lui est pas permis de modifier les valeurs de sa liste privée pas plus que de modifier la liste commune, privilège qui reste réservé à un nombre restreint d'utilisateurs.
Le principe de la liste de valeurs privée est illustré à la figure 6.19.

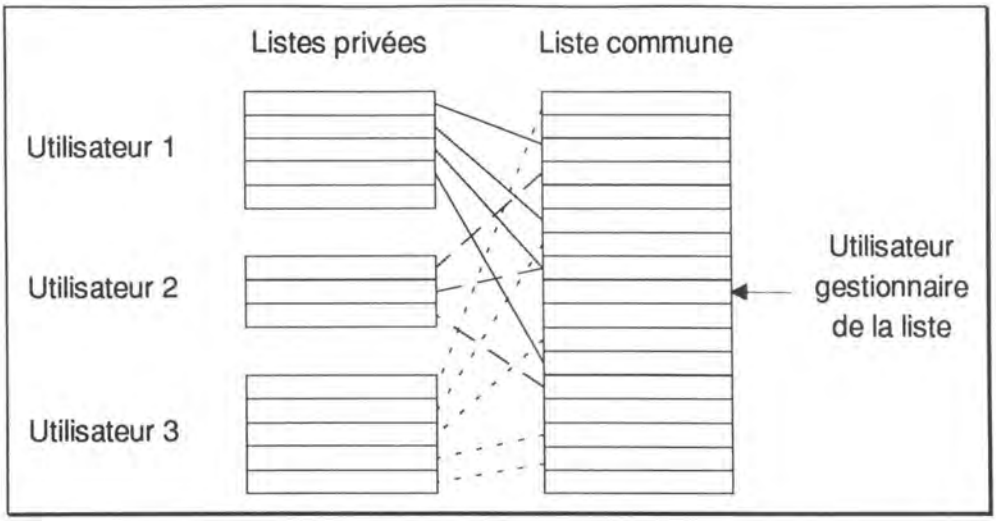

Fig. 6.19: Le principe des listes privées

Un exemple de liste de valeurs séparée en liste privée et liste commune est repris au figures 6.20. et 6.21. Les croix présentes à la gauche de la liste commune (figure 6.20.) signalent que les valeurs correspondantes font partie de la liste privée de l'utilisateur.

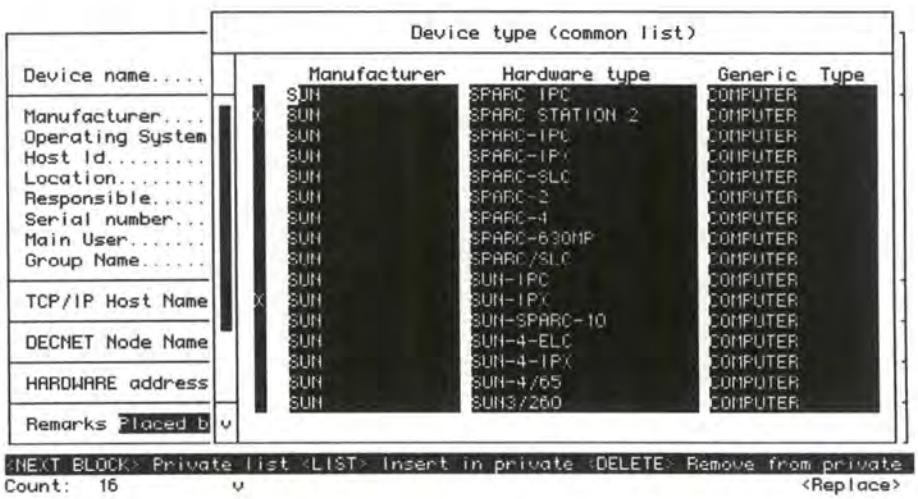

Fig. 6.20: Liste de valeurs commune pour les types de machines (Device Type)

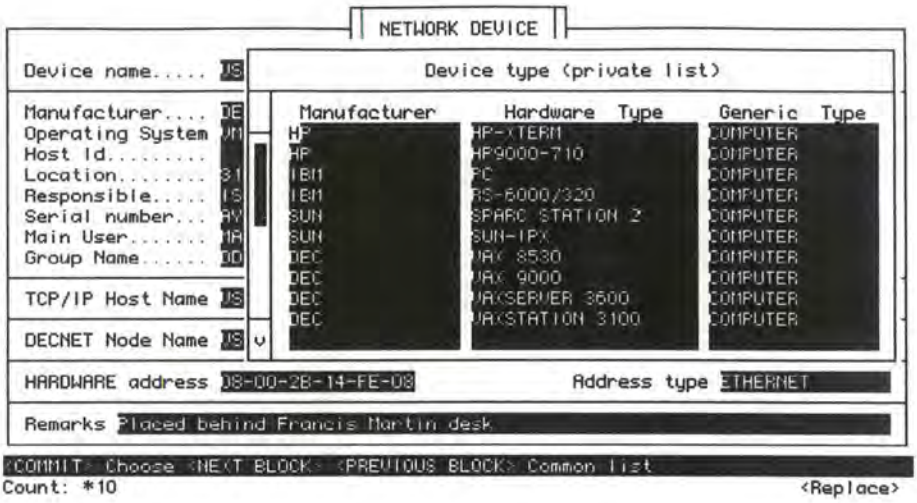

Fig. 6.21.: Une possibilité de liste de valeurs privée pour les types de machines (Device Type)

6.4.5. Historiques

Comme nous l'avons vu lors de l'analyse des tables de la base de données, un système d'historiques a été mis en place pour certaines tables jugées plus importantes.
Les historiques conservant les états successifs des enregistrements au fur et à mesure de leurs modifications, il serait intéressant de pouvoir "revenir en arrière" en rétablissant un état donné.
Les interfaces comprennent des formes affichant les historiques. Ces formes permettent à partir d'une forme concernant une table, de consulter les différents états qu'a connu l'enregistrement courant de cette forme dans l'ordre anti-chronologique des opérations qu'il a subie (de l'opération la plus récente à l'opération la plus ancienne encore gardée dans l'historique).
S'il le désire, l'utilisateur peut demander la récupération d'un état donné (sous réserve d'existence d'enregistrements associés, comme nous l'avons vu à la section 5.4.2).
Les tables d'historique étant accessibles depuis l'interface, cela permet à n'importe quel utilisateur de vérifier comment, quand et par qui des données auraient pu être modifiées puis de contacter la personne pour trouver un arrangement.
L'accès aux historiques s'inscrit donc parfaitement dans l'objectif de sécurité défini plus haut.
La figure 6.22. reprend un exemple de forme d'historique, celui de la table Network Device.

Fig. 6.22.: La forme d'historique de la table Network Device

6.4.6. Informations sur les risques de conflit

Le nombre de données dont l'accès est partagé étant tellement grand, l'application doit être à même de pouvoir prévenir l'utilisateur lorsqu'il accède à des données pouvant créer une situation de conflit avec un autre utilisateur.

Nous l'avons déjà signalé lors de la description de la base de données, le nombre d'adresses TCP/IP est tellement important, que le parc d'adresses a été subdivisé en zones (TCP/IP Areas). Chaque zone est confiée à la responsabilité d'un utilisateur. Certaines zones se recouvrant partiellement, il est posssible que deux utilisateurs différents (voire plus) soient capables d'accéder à une même série d'adresses, créant ainsi un risque de conflit, à savoir la modification d'une adresse TCP/IP par un autre utilisateur.
Bien que des outils aient été mis en place pour pouvoir détecter comment et par qui un conflit est né (par les historiques entre autres), il est intéressant que les interfaces offrent la possibilité d'être prévenu *à l'avance* d'un risque de conflit.

Pour la gestion des adresses, deux formes ont été prévues à cet effet. La première offre à l'utilisateur de voir quels comptes utilisateurs sont susceptibles de pouvoir modifier une adresse TCP/IP donnée. C'est la forme "TCP/IP Address Owners". La deuxième, la forme "TCP/IP Areas Mapping" reprend la même philosophie en affichant de manière semi-graphique (toujours en mode texte évidemment), les zones TCP/IP recouvrant tout ou partie d'un intervalle d'adresses TCP/IP.

6.4.7. Informations sur les utilisateurs

Comme nous l'avons vu, le nombre d'utilisateurs peut être élévé et un minimum d'information sur ces derniers doit être consultable. Il existe donc une forme permettant à tous les utilisateurs de consulter cette information: compte utilisateur, nom de la personne réelle, téléphone, e-mail.
Cette forme est entre autres très utile lorsque l'on ne dispose que du compte Oracle de l'utilisateur comme c'est souvent le cas et en particulier dans l'historique.
On peut déplorer le fait que les informations sur les personnes réelles utilisant un compte Oracle soient regroupées dans quelques lignes de texte (cfr. structure de la table User_Account). Une référence à la table des informations sur le personnel aurait été tout à

fait justifiée: la mise à jour des informations (téléphone, adresse e-mail...) aurait été automatique et les informations plus complètes (division, bureau...).

6.4.8. Détection de données enregistrées plusieurs fois

Des données concernant une même machine peuvent être entrées par des utilisateurs différents. Il n'est pas rare de voir des informations entrées plusieurs fois ou ne pas être mises en commun alors qu'elles devraient l'être. C'est un problème propre à l'aspect partagé et multi-utilisateurs de la gestion d'un parc machine. L'application doit pouvoir détecter dans la mesure du possible de pareils cas et offrir à l'utilisateur des solutions lui permettant de régler le problème.

Par exemple, un utilisateur peut être plus en charge des données administratives alors qu'un autre sera chargé des données de type protocolaire. Il n'est pas rare que deux personnes essayent ainsi d'entrer des données sur la même machine avec des noms différents, chacun ayant son point de vue propre (administratif ou protocolaire) de la machine.
L'application doit tant que faire se peut détecter toute tentative d'entrer des données sur des machines apparemment identiques, mais auxquelles on a donné des noms différents.
Ainsi, si un utilisateur tente de donner un nom à une adresse DECnet ou TCP/IP et que ce nom est utilisé par une autre machine, il est possible que ces deux machines n'en soient qu'une seule. L'application le signale alors à l'utilisateur et lui propose de fusionner les deux machines. Ce que peut accepter ou refuser l'utilisateur.
La fusion donnera alors priorité à la machine qui portait déjà le nom convoité. Mais pour tous les champs qui ne seraient pas instanciés, une copie de ceux de la machine qui sera "sacrifiée" est faite. On obtient ainsi un enregistrement unique composé du maximum d'informations possibles.

Cette fonction porte le nom de Merge (fusion) dans LANDB. La figure 6.23. représente l'écran de fusion pour les machines (Network Device Merge).

Fig. 6.23: L'écran de fusion des machines

6.5. Aide à l'utilisateur

Le domaine de la gestion des réseaux étant assez complexe et le niveau de formation des utilisateurs à ce sujet étant très différent, il est nécessaire d'offrir à ces derniers une aide complète et générale. Elle doit, en effet, s'adresser à un public hétérogène.

6.5.1. Liste de valeurs

Comme nous l'avons déjà signalé, les listes de valeurs assurent le contrôle et la sécurité au niveau des données entrées par l'utilisateur. Elles jouent le rôle supplémentaire de guide pour l'utilisateur qui n'est pas certain du format ou de l'orthographe des données qu'il doit encoder.

6.5.2. Listes de valeurs complexes

Il existe un autre type de listes de valeurs que nous appellerons *listes de valeurs complexes*. Ce type de listes de valeurs est spécifique à un cas particulier de valeur à choisir. Il s'agit en fait de formes ou de touches de fonction permettant de choisir une valeur répondant à certains critères parmi des valeurs possibles.
Nous les appelons listes de valeurs bien qu'il n'y ait pas véritablement de liste affichée, mais le choix d'une valeur dans une série de valeurs disponibles se rapproche assez bien du concept de liste.
Ces listes ont plus pour but d'aider l'utilisateur dans son choix que de vérifier le format ou la valeur des données qu'ils rentrent, bien que cela soit également fait.

Ce type d'outil est primordial dans une application multi-utilisateurs surtout au vu du nombre important de valeurs possibles de certains champs et de l'accès concurrent.

Une de ces listes est, par exemple, le choix d'une adresse TCP/IP libre. Depuis l'écran NETDEV_C, lorsque le curseur se trouve sur les champs concernant les adresses TCP/IP, l'utilisateur peut appeler la liste de choix d'une adresse libre. L'écran qui apparaît alors lui donne la liste des areas TCP/IP dans lesquelles il a le droit d'assigner des adresses à une machine. Pour chacune de ces areas, l'utilisateur a la possibilité de calculer la première adresse TCP/IP libre, mais également la plus haute adresse libre. Cette dernière est fort utile lorsque l'utilisateur veut être certain de pouvoir trouver plusieurs adresses libres qui se suivent.
Un deuxième exemple est la possibilité de faire calculer automatiquement le *node number* libre suivant dans une area DECnet donnée lorsque l'utilisateur se trouve sur le champ Node Number.
L'avantage de ces deux listes est d'assurer à l'utilisateur que les valeurs retournées sont disponibles. Ceci permet d'améliorer le travail de l'utilisateur, car le nombre de valeurs susceptibles d'être choisies peut être grand et cela évite à l'utilisateur de rentrer dans un fastidieux processus d'essais-erreurs.

6.5.3. Pages d'aides

Dans LANDB, l'utilisateur peut appeler une page d'aide depuis chaque block de chaque forme. La page d'aide donne ainsi des explications plus fournies sur chaque champ du block: but du champ, relations avec les autres champs, règles existant entre les différents champs...
La page d'aide donne également la liste des touches génériques ayant été rédéfinies, c'est-à-dire, ne donnant pas l'effet par défaut défini par SQL*Forms.
La figure 6.24. représente la page d'aide associée au block NETWORK_DEVICE de l'écran Network Device.

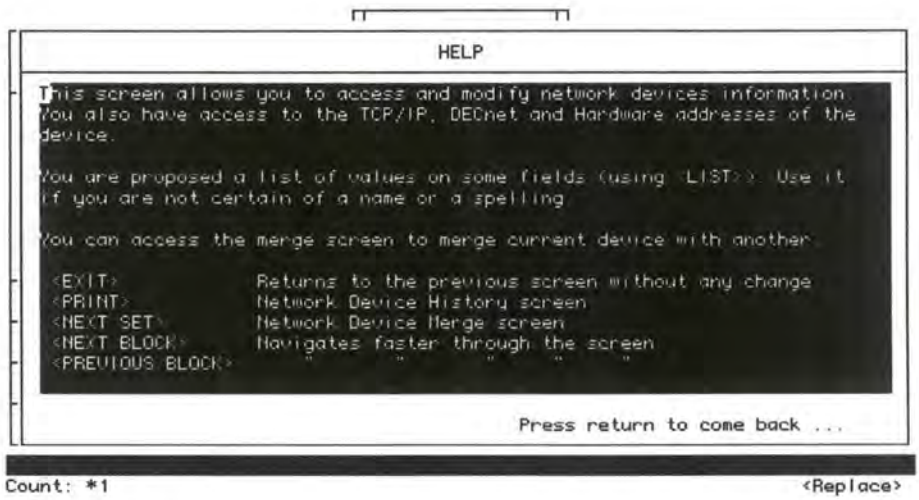

Fig. 6.24.: L'écran d'aide en ligne du block NETWORK_DEVICE

6.5.4. Quick Reference

En plus des pages d'aide plein écran, chaque champ peut se voir associé une ligne d'aide qui s'affiche en bas de l'écran. Elle s'appelle Quick Reference Line. Cette ligne a été utilisée pour afficher la liste des touches génériques ayant été redéfinies, c'est-à-dire ne donnant pas l'effet par défaut défini par SQL*Forms. Un exemple est donné à la figure 6.25.
Lorsqu'un champ peut prêter à confusion, la ligne d'aide est complétée par une description sommaire du champ, ce qui évite de devoir appeler l'aide en ligne pleine page.

Fig. 6.25.: Une ligne d'aide "Quick Reference"

6.5.5. Documentation

L'aide en ligne se doit d'être complétée par une documentation adéquate.

La documentation existant pour les écrans de gestion des adresses (et qui est reprise en annexe) apporte plusieurs choses qui ne sont pas reprises dans l'aide en ligne. Le but au moment de sa rédaction était de fournir un document à même de répondre aux questions élémentaires de l'utilisateur et surtout de manière pratique: comment effectuer telle ou telle opération typique... Cette documentation n'est donc pas une référence complète, mais plutôt

un manuel d'apprentissage. Ce choix a été fait sur base de la demande des premiers utilisateurs ayant une bonne maîtrise des notions d'adressage, mais requérant un apprentissage des interfaces créées par SQL*Forms 3.0.

La documentation comprend d'abord une explication du fonctionnement général des interfaces générées par SQL*Forms. Cette partie s'adresse tout particulièrement aux personnes utilisant LANDB pour la première fois.

Après une brève description des données affichées sur les différents écrans, elle reprend des exemples détaillés d'opérations types que l'utilisateur devra certainement accomplir tôt ou tard: modifier une adresse, ajouter une nouvelle machine, supprimer une adresse. Ces exemples tentent d'être représentatifs du travail quotidien des utilisateurs.

La documentation donne ensuite un aperçu des listes de valeurs complexes propres à certains champs de la forme: choix d'une adresse TCP/IP ou DECnet libre.

Malgré cette documentation fort utile à l'utilisateur qui débute, LANDB pêche par l'absence d'autre documentation. Un manuel exhaustif donnant une description de tous les écrans de l'application serait fort utile ou, au moins, une liste complète des significations des champs de l'ensemble des formes (qui ne sont pas toujours les mêmes que ceux de la base de données malheureusement).
Un tel ouvrage n'a pas été réalisé pour la seule raison que les utilisateurs actuels de LANDB ont un niveau de spécialisation assez élevé dans leur domaine (adressage, cablâge...) et ne requièrent qu'un simple guide introductif sur l'utilisation des écrans qui les intéressent particulièrement. Néanmoins, si l'application venait à s'ouvrir à des utilisateurs occasionnels désireux, par exemple, de consulter quelques renseignements sur les machines sur lesquelles ils travaillent (responsable, adresses TCP/IP...), un manuel d'utilisation plus général reprenant une présentation détaillée des informations reprises dans la base de données et de leur signification serait le bienvenu.

6.6. Conclusion

Le développement d'interfaces pour une application telle que LANDB demande au développeur d'observer quelques principes. Les développeurs de LANDB ont essayé d'appliquer au mieux ceux qui suivent.

Le nombre important d'utilisateurs, l'hétérogénéité de leurs fonctions et de leur formation ainsi que la complexité et l'instabilité de la structure de la base de données doit pousser le développeur à assurer une modularisation maximale aux interfaces utilisateur. Ceci dans le but de réutiliser au maximum des parties communes à tous les utilisateurs et éviter de développer des interfaces trop monolithiques et spécifiques à une fonction particulière. De cette manière, la maintenance des interfaces sera plus facile à assurer lors de changements tant du côté de la structure des données que du côté des utilisateurs en impliquant un minimum de modifications.
Néanmoins, les tâches que doivent remplir les utilisateurs ne peuvent être ignorées et doivent toujours être prises en compte dans l'élaboration d'interfaces plus spécifiques regroupant des données issues de plusieurs tables différentes.

Dès lors, le développement des interfaces doit prendre en compte la structure des données autant que la définition de la tâche des utilisateurs. Il faut néanmoins donner priorité aux formes associées à des tables simples sans quoi le coût de maintenabilité des interfaces serait trop lourd étant donné l'instabilité de la structure des données.

La modularité des interfaces ainsi assurée doit être accompagnée d'un système permettant un passage facile d'une interface à l'autre et, cela, de manière cohérente. Ce système doit enfin être défini de manière à pouvoir modifier la structure des possibilités d'appels entre formes avec un minimum de travail afin de pouvoir développer de nouvelles interfaces (combinaisons d'interfaces) à la demande.

L'hétérogénéité de la population des utilisateurs est aussi à l'origine d'un besoin d'assurer via les interfaces un contrôle dans la manipulation des données. Ce contrôle doit éviter la prolifération de redondance dû à des problèmes de format ou d'orthographe courante dans les applications multi-utilisateurs. Plusieurs outils sont utilisables pour assurer ce contrôle parmi lesquels les listes de valeurs correctes nous semblent parfaitement adaptées. Un contrôle peut également être effectué pour éviter la simple redondance de données (introduction multiple de données déjà existantes).

La sécurité doit être renforcée étant donné l'accès concurrent aux données. Pour assurer cette sécurité, un module obligatoire d'accès et de vérification aux données semble être la solution idéale. Enfin, un système gardant la trace des opérations effectuée facilite la détection et la récupération d'erreurs.

La différence de formation des utilisateurs en ce qui concerne le domaine de l'application doit impliquer le développement d'une aide s'adressant à tous les types d'utilisateurs avec une petite priorité aux utilisateurs novices.
Des outils d'aide peuvent également servir à réduire la charge de travail de l'utilisateur quand les possibilités d'actions sont nombreuses (choix automatique de valeurs...).

Parmi les reproches que l'on peut faire aux interfaces de l'application LANDB, citons l'utilisation d'une interface de type texte alors que la tendance actuelle est aux interfaces graphiques de type WIMP. Ce choix a été justifié par des impératifs de maintenabilité, comme nous l'avons déjà signalé.

Une certaine complexité découlant du nombre impressionnant de formes et des nombreuses possibilités d'appel entre elles peut aussi perturber l'utilisateur habitué à des interfaces plus monolithiques spécifiquement développées pour sa tâche.

Enfin, une documentation devrait être écrite pour à un public moins expert dans le domaine de l'adressage et des réseaux.

Conclusion

Conclusion générale

Les réseaux informatiques actuels sont caractérisés par une hétérogénéité importante tant au niveau matériel que logiciel. Cette hétérogénéité se retrouve également au niveau des utilisateurs qui sont de plus en plus originaires d'horizons professionnels très divers.

Les gestionnaires de réseaux, face à cette situation, attendent des outils de gestion qu'ils soient complets et souples afin de pouvoir gérer de manière aisée tous les types d'information pouvant concerner le réseau: matérielles et protocolaires tout d'abord, mais également administratives, logicielles, humaines et autres.

Pour la gestion on-line, c'est-à-dire la gestion du réseau en temps réel (gestion du trafic, des pannes, etc.), des applications ouvertes à l'hétérogénéité techniques apparaissent de plus en plus et permettent de gérer les aspects les plus techniques des réseaux: protocoles, trafic, pannes, état, topologie, etc.

Par contre, cruel est le manque d'applications de gestion off-line des réseaux. Ces applications devraient rassembler les informations ne pouvant être recueillies automatiquement: administration, utilisateurs, responsables, localisation, logiciels installés, licences d'exploitation et toute autre information qui n'est généralement pas gérée par les applications de gestion on-line des réseaux.

Alors que précédemment, la gestion de réseau était souvent confiée à une seule personne, les applications de gestion de réseaux actuelles doivent être attentives à permettre une certaine délégation de la gestion. L'information de gestion étant, en effet, plus complète, les personnes concernées par cette dernière sont plus nombreuses.

Les standards de gestion de réseaux tels que SNMP et CMIS-CMIP permettent de définir une base d'informations de gestion. Malgré une certaine souplesse rendue possible par la manière de définir la base, ces standards se sont axés principalement sur les informations de type protocolaires laissant libre choix à l'utilisateur de définir une base d'information plus complète. Ce silence n'aide donc pas beaucoup le gestionnaire soucieux d'utiliser un standard pour la gestion d'informations sur le réseau qui soient complètes.

Le cas du CERN est remarquable du fait de la taille et de l'hétérogénéité de ses réseaux locaux. Il préfigure probablement les réseaux de demain et il est intéressant de s'attarder à observer la manière dont l'information sur les réseaux y est gérée.

Très vite, la première application CSNET a montré des faiblesses à s'adapter aux différents changements techniques et organisationnels des réseaux. Elle était extrêmement liée aux fonctions de gestion de quatre types d'utilisateurs et ne pouvait supporter les spécificités techniques de nouveaux matériels.

La nouvelle application LANDB a permis de mettre en oeuvre quelques principes permettant de tenir compte de l'hétérogénéité du matériel et des utilisateurs ainsi que de l'instabilité tant technique qu'organisationnelle des réseaux. Les objectifs de centralisation de diverses sources d'informations sur les machines branchées sur les réseaux, de maintien de qualité des données et d'adaptation aux changements nous paraissent tout à fait transposables à d'autres cas que celui du CERN.

Un choix d'outils multiplateformes est nécessaire afin de permettre aux utilisateurs d'accéder à l'application depuis des environnements matériels différents, situation actuellement fréquente dans les réseaux multi-vendeurs.

La structure de la base de données doit être éclatée de manière à représenter la complexité du réseau de manière simple et de façon qu'elle puisse suivre facilement tout changement technique ou apparition de nouvelles informations. Toutefois, il faut éviter de verser dans l'excès et regrouper des informations trop anodines pour justifier l'existence d'une table propre. Ce choix s'effectue selon les cas et les priorités des gestionnaires du réseau.
Enfin, la base de données doit contenir certaines informations propres à améliorer le fonctionnement général de l'application en parallèle avec les interfaces utilisateurs. Nous pensons ici aux historiques et aux listes de valeurs autorisées.

Les interfaces utilisateur doivent quant à elle être développées dans un souci d'adaptabilité aux changements dans la structure de la base et à l'hétérogénéité de la population des utilisateurs.
Ainsi, une certaine modularité doit guider le développement des interfaces afin de rendre leur adaptation plus aisée.
De plus, les interfaces doivent permettre d'assurer une certaine sécurité et un contrôle au niveau des valeurs entrées ou modifiées par des utilisateurs ayant une connaissance très hétéroclite des informations de gestion et de leur signification particulièrement à cause de l'accès concurrent dont elles sont l'objet.
Enfin, les interfaces utilisateurs doivent permettre de rendre acccessible l'application à tous les types d'utilisateurs, du novice à l'expert.

Ainsi, la base de données rassemble toutes sortes d'informations autant techniques que non liées au fonctionnement direct du réseau. Elle est accompagnée d'une application complète et accessible par tous les acteurs impliqués par la gestion du réseau. C'est à cette condition seule que pourra être maintenue toute l'information nécessaire aux différents gestionnaires.

Critique de la démarche adoptée

Spécificité du cas LANDB

Il est évident que ce mémoire se base principalement sur le cas de l'application LANDB développée au CERN. On peut reprocher à cette démarche de s'attacher de trop près à une application propriétaire et par là même d'intégrer dans une réflexion générale sur les informations de gestion de réseaux des éléments trop spécifiques à un cas particulier.
Néanmoins, le cas du CERN, bien qu'étant un cas unique de par la taille et l'hétérogénéité de ses réseaux locaux, semble être représentatif de ce vers quoi on tend aujourd'hui: des réseaux extrêmement condensés et hétérogènes avec des utilisateurs nombreux et divers. De plus, les protocoles et les machines retrouvées au CERN représentent bien ceux qu'on retrouve actuellement dans la plupart des réseaux locaux.

Prépondérance de l'analyse des interfaces sur celle de la base de données

En ce qui concerne l'analyse de LANDB, elle s'est principalement portée sur les interfaces utilisateur. Il n'a pas été fait d'analyse en profondeur des motivations et des choix des développeurs en ce qui concerne la structure de la base de données. Plusieurs raisons sont à l'origine de ce choix. Tout d'abord, le développement de cette structure de données s'est basé sur le cas particulier du CERN et sur l'expérience acquise avec l'application CSNET: statistiques, volumes de données existantes, désirs des utilisateurs,... Mais nous avons surtout voulu nous attarder sur les principes généraux ayant inspiré le développement des interfaces car ils nous paraissaient plus transposables à d'autres applications du genre de LANDB.

Illustration restreinte des principes de développement des interfaces

L'analyse des interfaces utilisateur n'a, quant à elle, été illustrée que par les interfaces spécifiques à la gestion des informations protocolaires de la base de données. Comme nous l'avons déjà signalé, cette limitation était motivée par l'état d'avancement des interfaces utilisateurs au moment de la rédaction de ce mémoire ainsi que par le manque de place pour aborder d'autres exemples.

Méthode d'analyse empirique

Enfin, la méthode d'analyse des principes de développement des interfaces utilisateur peut à juste titre paraître fort empirique. La littérature en matière d'interfaces hommes-machines manque en effet, cruellement de références en ce qui concerne les applications destinées à être employées par un nombre important d'utilisateurs à des fins différentes. Nous nous sommes donc rabattus sur un mode d'analyse se basant sur l'expérience acquise durant le stage effectué au CERN au contact tant des développeurs que des utilisateurs. Enfin, on manque encore, à l'heure actuelle, de résultats et d'analyses permettant d'affirmer si les principes de développement tant de la base de données que des aspects interfaces sont payants.

Propositions pour la suite

Approfondir les principes de développement de la structure de la base de données

Afin de se détacher d'un environnement trop particulier et spécifique comme celui du CERN, il serait bon d'essayer de définir les principes de développement d'une application de gestion d'informations off-line sur les réseaux en ce qui concerne la structure de la base de données, et cela pour un réseau de n'importe quel type. Il serait, en effet, intéressant de définir une structure de données basée sur les types généraux de données définis dans la première partie et d'essayer de les associer aux principes de développement des interfaces qui ont été proposés ici. On éviterait ainsi l'attachement à une application particulière que l'on pourrait reprocher à ce travail.

Comparaison avec des applications non propriétaires

Il serait ensuite intéressant de comparer les principes définis dans ce travail à ceux observés dans des applications de gestion de réseaux existant sur le marché et similaires à LANDB, car la préoccupation actuelle des firmes proposant de telles applications est également d'offrir des outils de gestion ouverts et non propriétaires.

Continuer l'analyse

Ensuite, il serait adéquat de continuer l'analyse faite ici sur LANDB pour voir comment s'orientera l'application tant au niveau des fonctionnalités existant déjà qu'au niveau de celles qui seront implémentées dans un futur proche: gestion de l'aspect logiciel, gestion administrative complète, intégration avec les autres outils de gestion du réseau... Il sera alors intéressant de voir si les principes de développement se sont révélés payants.

Bibliographie

Ouvrages

[Bod89] François BODART, Yves PIGNEUR, *Conception assistée des systèmes d'information, méthode - modèles - outils*, Masson, 1989

[Bod90] François BODART, *Cours introductif aux interfaces homme-machine, Notes de cours provisoires du Chapitre I*, Institut d'Informatique, Namur, 1989-1990

[Bro88] C. Marlin BROWN, *Human-Computer Interface Design Guidelines*, Ablex Publishing, Norwood, New Jersey, 1988

[Com91] Douglas E. COMER, *Internetworking with TCP/IP, Volume I, Principles, protocols and architectures, Second edition*, Prentice Hall Internaltional Editions, 1991

[Dat86] C. J. DATE, *Relational Database, Selected Writings*, Addison-Wesley Publishing Company, Reading, Massachussets, 1986

[Det87] Bernard DETREMBLEUR, *Network management in an interconnected networking environment: the current state of art from the OSI point of view and a description of a practical implementation*, Facultés Universitaires Notre-Dame de la Paix, Institut d'Informatique, Namur, 1986-1987

[Gal89] GALACSI, *Conception de bases de données, du schéma conceptuel aux schémas physiques*, Dunod Informatique, Paris, 1989

[Hai86] Jean-Luc HAINAUT, *Conception assistée des applications informatiques, Tome 2, Conception de la base de données*, Masson, Presses Universitaires de Namur, 1986

[Hey91] Didier HEYVAERT, *Network Management Systems, The management of TCP/IP networks*, Facultés Universitaires Notre-Dame de la Paix, Institut d'Informatique, Namur, 1990-1991

[ISO7498-4] ISO/IEC 7498-4, *Information processing Systems - Open Systems Interconnection - Basic Reference Model - Part 4: Management Framework*, International Standards Organization, November 15, 1989

[ISO8824] ISO/IEC 8824, *Information processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)*, International Standards Organization, May, 1987

[Mar92] Olivier Francis MARTIN, *Proposition pour une base de données centralisée*, CERN, Genève, 1992

[Rav89] Susannah RAVDEN, Graham JOHNSON, *Evaluating Usability of Human-Computer interfaces, a practical method*, Ellis Horwood, 1989

[Rfc1156] Keith McCLOGHRIE and Marshall T. ROSE, *Management Information Base Network Management of TCP/IP Based Internets, Requests for Comments 1156*, DDN Network Information Center, SRI International, May, 1990

[Rfc1158] Marshall T. ROSE (editor), *Management Information Base Network Management of TCP/IP Based Internets, MIB-II Requests for Comments 1158*, DDN Network Information Center, SRI International, May, 1990

[Rose91] Marshall T. ROSE, *The Simple Book: an introduction to management of TCP/IP-based Internets*, Prentice Hall, Englewood Cliffs, New Jersey, 1991

[Sac91] B. SACRE, I. PROVOT, *Proposition d'un langage de spécification de l'interface homme-machine d'une application de gestion hautement interactive*, Institut d'Informatique, Namur, 1991

[Shn87] Ben SHNEIDERMAN, *Designing the user interface: strategies for effective human-computer interaction*, Addison-Wesley, Reading, Massachussets, 1987

Articles

[Ree90] David P. REED, *Guest Editorial*, IEEE Network Magazine, July 1990, Vol. 4, n°3, p. 4

[Emb90] Jack EMBRY, Peter MANSON, Dave MILHAM, *An Open Network Management: OSI/NM Forum, Architecture and Concepts*, IEEE Network Magazine, July 1990, Vol. 4, n°3, pp. 14-21

[Pin91] Edward L. PINNES, Sunil PODAR, *Guest editorial*, IEEE Network Magazine, March 1991, Vol. 5, n°2, p. 4

Annexe 1:

Schéma Oracle*Case de la base de données LANDB

LANDB DATABASE SCHEME

28-OCT-92

M-E.Manrique - CN/CE
O.F.Martin - CN/CS/IN

USER PROGRAM
USER ACCOUNT
ROLE_TAB _PRIVS
PERSONAL INFORMATION
DEVICE TYPE
GENERIC TYPE
OPERATING SYSTEM
PRODUCT
LICENSE
LICENSE HISTORY
INSTALLATION
PROTECTED SOFTWARE INSTALLATION
INSTALLATION HISTORY
DESCRIPTION
DATABASE
FINANCIAL
FINANCIAL ADMIN HISTORY
LOGICAL GROUP
CLUSTER
NETWORK DEVICE
DESKTOP COMPUTER
SUN WORKSTATION
PERSONAL COMPUTER
HP APOLLO
ETHERNET DEVICE
PERIPHERIC TYPE
PERIPHERIC
PERIPHERIC HISTORY
PARTITIONS DISK
CABLE
CABLE HISTORY
HARDWARE CONNECTION
HARDWARE CONNECTION HISTORY
HARDWARE ADDRESS TYPE
HARDWARE ADDRESS
HARDWARE ADDRESS HISTORY
NETWORK DEVICE HISTORY
ERRORS
HELP
SYSTEM
DECNET V5 NAME
DECNET V5 ALIAS
DECNET V5 ALIAS HISTORY
DECNET ADDRESS V5
DECNET ADDRESS V5 HISTORY
DECNET ADDRESS
DECNET ADDRESS HISTORY
TCPIP ALIAS
TCPIP ALIAS HISTORY
TCPIP AREA
NICCOMMENT
TCPIP ADDRESS
TCPIP ADDRESS HISTORY
TCPIP NETWORK
EXTERNAL TCPIP
EXTERNAL IP LINE HISTORY
APOLLO CONNECTION
APOLLO CONNECTION HISTORY
APOLLO NETWORK

Annexe 2

Manuel d'utilisation des interfaces de gestion des adresses

Cette annexe reprend le manuel d'utilisation rédigé à l'attention des premiers utilisateurs des interfaces de gestion de l'adressage sur LANDB.
Ce manuel est plus un guide d'apprentissage qu'un guide à consulter à tout moment.

CERN
European Laboratory for Particle Physics
Geneva, Switzerland

LANDB DATABASE

NETWORK MANAGEMENT SCREEN

User guide

François Briard / CN/CS/IN - Olivier Francis Martin / CN/CS/IN
January, 1993

Table of contents

0. General presentation of the LANDB database 2

1. General presentation of the screen 3

1.1. Main purpose of this screen 3
1.2. Different parts of the screen 4
1.3. Description of fields 6
1.4. General features 8
1.4.1. Quick reference 8
1.4.2. On-line help 8

2. Query, Insert, Update and Delete 9

2.1. Query 9
2.1.1. General use 9
2.1.2. Some tips 9
2.1.3. Examples 10
2.2. Insert 12
2.2.1. General Use 12
2.2.2. Some tips 13
2.2.3. Examples 15
2.3. Update 17
2.3.1. General Use 17
2.3.2. Some tips 17
2.3.3. Example 19
2.4. Delete 20
2.4.1. General Use 20
2.4.2. Examples 20

3. Some specific features 22

3.1. Lists of values 22
3.2. The DEVICE MERGE screen 26
3.3. The HISTORY 28
3.4. The addresses details screens (EDIT) 30

4. Specific application errors 31

Note

All keys described in this user guide will be SQL*forms generic function keys independent of which keyboard and software configuration you are using. They will appear between "<" and ">" symbols (like "<COMMIT>").

If you want to know which key on your keyboard correspond to a generic function key please refer to the key mapping of your specific software and hardware configuration. An on-line key-mapping (dependent on the operating system) is always callable from the application.

0. General presentation of the LANDB database

The LANDB database has been designed to register information related to the LANs (Local Area Networks) at CERN. It replaces the former CSNET database and allows one to manage more types of information in an easier way.

It is mainly used by people in charge of managing different aspects of the network: local manager, network managers and all people interested in querying information about the devices connected to the network.

Information recorded in the LANDB database is, for example, about:

- the computers and peripherals linked to them
- the connection devices (bridges, gateways,...)
- the cables and all devices connected to them
- the communication protocols (TCP/IP, DECnet, Hardware addresses)
- the persons responsible for and using the devices
- the software running on the devices
- etc...

1. General presentation of the screen

1.1. Main purpose of this screen

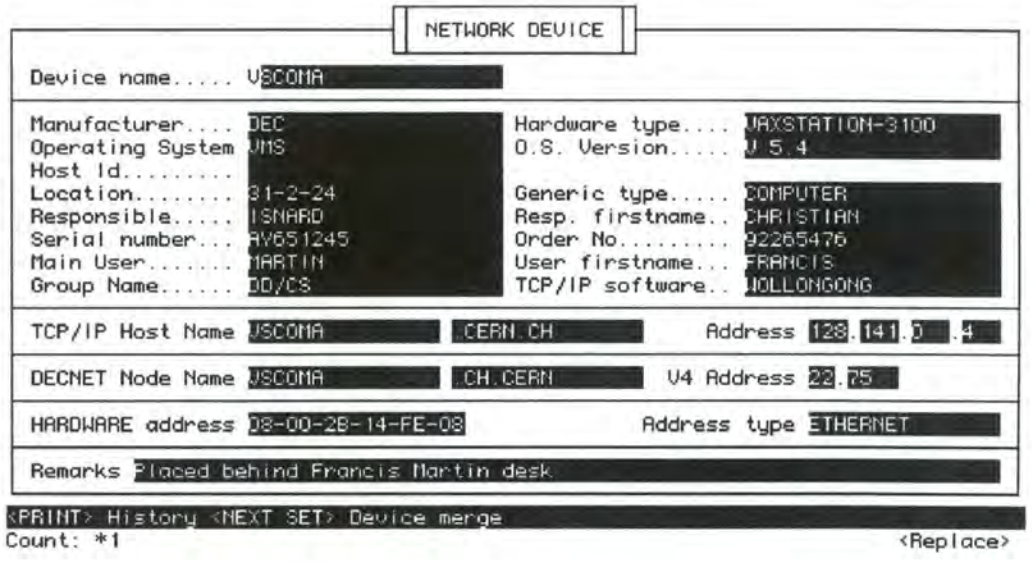

Fig. 1 : Network Devices screen

The NETWORK DEVICES screen's main purpose is to allow people in charge of protocol management to insert new network device information in the LANDB database .
It also allows them to query, update or delete such information.
This screen manages information about the network devices themselves, TCP/IP, DECnet and Hardware addresses of the devices.
It is a general information screen giving a summary of the protocol aspects of network devices. For more detailed information (aliases, areas,...) on a specific protocol please refer to other screens callable from this screen: TCP/IP addresses, DECnet addresses or Hardware addresses screens.

1.2. Different parts of the screen

The NETWORKS DEVICE screen is divided into 4 "blocks":

- The NETWORK DEVICE block: this block gathers some overall information about the network device itself (the "box"), the people responsible for it or using it, the group it belongs to and the TCP/IP software running on it (note that the REMARKS field belongs to the Network Device block though it is shown as separate).

Fig. 2 : the network device block

- The TCP/IP ADDRESS block: this block specifies a TCP/IP address which is associated with the network device, i.e. a TCP/IP Hostname (as declared in the name-server; note that it can be different to the device name) and address. There may be more than one TCP/IP address associated with the same device. To view the other TCP/IP addresses, press <PREVIOUS RECORD> or <NEXT RECORD> when the cursor is in the TCP/IP address part.

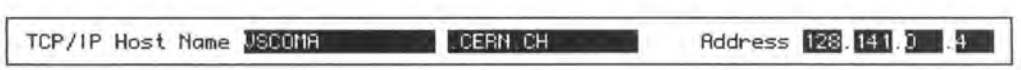

Fig. 3 : the TCP/IP address block

- The DECNET ADDRESS block: this block describes a (phase IV) DECnet address which is associated with the network device, i.e. NCP Node name (as declared in the NCP base; note that it can be different to the device name) and address. There may be (exceptionally) more than one DECnet address associated with the same device. To view the other DECnet addresses, press <PREVIOUS RECORD> or <NEXT RECORD> when the cursor is in the DECnet address part.

Fig. 4 : the DECnet address block

- The HARDWARE ADDRESS block: this block describes a hardware address which is associated with the network device. There may be (exceptionally) more than one DECnet address associated with the same device. To view the other hardware addresses, press <PREVIOUS RECORD> or <NEXT RECORD> when the cursor is in the hardware address part.

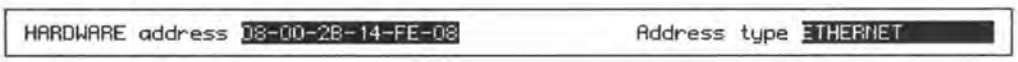

Fig. 5 : the hardware address block

As can be seen, there are 4 differents "objects" on this screen which are linked together (all the addresses belong to the same network device). Therefore, some operations, such as DELETE RECORD, INSERT RECORD,..., may only have local effects on the current block of the screen. Deleting a TCP/IP address will not delete the network device it belongs to BUT deleting a network device will delete all addresses linked to it (see chapter 2 on insert and delete operations).

1.3. Description of fields

NETWORK DEVICE

Field Name	Description	Redefined keys
DEVICE NAME	Usual name of the device This name identifies the device, not an address, even if the address names (hostname and node name) are usually the same.	<PRINT> <NEXT SET>
MANUFACTURER	Manufacturer of the device (use list of values)	<LIST> <PRINT> <NEXT SET>
HARDWARE TYPE	Hardware type of the device Related to the manufacturer (use list of values)	<LIST> <PRINT> <NEXT SET>
OPERATING SYSTEM	Name of the operating system (use list of values)	<LIST> <PRINT> <NEXT SET>
OPERATING SYSTEM VERSION	Version of the operating system Related to the operating system name (use list of values)	<LIST> <PRINT> <NEXT SET>
HOST ID	Unix machine unique identifier	<PRINT> <NEXT SET>
LOCATION	Location of the device Use bldg-floor-room format	<PRINT> <NEXT SET>
GENERIC TYPE	Generic device type Related to the manufacturer and the hardware type (use list of values)	<LIST> <PRINT> <NEXT SET>
RESPONSIBLE	Surname of person responsible for the device As it will not be allowed to leave the responsible field with invalid values (unknown person), use list of values.	<LIST>
RESP. FIRSTNAME	Firstname of person responsible for the device As it will not be allowed to leave the responsible field with invalid values (unknown person), use list of values.	<LIST>
SERIAL No	Manufacturer's serial number	<PRINT> <NEXT SET>
ORDER No	CERN order number	<PRINT> <NEXT SET>
MAIN USER	Surname of main user of the device As it will not be allowed to leave the user field with invalid values (unknown person), use list of values.	<LIST>

USER FIRSTNAME	First Name of main user of the device As it will not be allowed to leave the user field with invalid values (unknown person), use list of values.	<LIST>
GROUP NAME	Administrative group wich owns the device or cluster name to which the device belongs	<PRINT> <NEXT SET>
TCP/IP SOFTWARE	TCP/IP software running on the device	<PRINT> <NEXT SET>

TCP/IP ADDRESS

Field Name	Description	Redefined keys
TCP/IP HOST NAME (name and domain)	TCP/IP Host name as declared in the name server Often the same as the device name	<PRINT> <EDIT> <LIST>
TCP/IP ADDRESS (4 bytes)	TCP/IP Address (last byte must be greater than 0 and all bytes must be less than 255)	<PRINT> <EDIT> <LIST>

DECnet ADDRESS

Field Name	Description	Redefined keys
DECNET NODE NAME (name and domain)	DECnet phase IV Node name as declared in the NCP base Often the same as the the device name	<PRINT> <EDIT>
DECNET ADDRESS (area and node number)	DECnet phase IV Address Area must be in 1-63 range Node number must be in 1-1023 range	<PRINT> <EDIT> <LIST>

HARDWARE ADDRESS

Field Name	Description	Redefined keys
HARDWARE ADDRESS	Hardware address Must contain 6 hex bytes with separators	<PRINT> <EDIT>
ADDRESS TYPE	Hardware address type (e.g. Ethernet) (use list of values)	<PRINT> <EDIT> <LIST>

1.4. General features

1.4.1 Quick reference

When not in ENTER QUERY mode (see 2.1.1.), the message line will display all the keys which have been redefined for the current field and a short description of what each redefined key does.

Fig. 5 : the quick reference line

1.4.2. On-line help

When not in ENTER QUERY mode (see 2.1.1.), you can access on-line help by pressing the <HELP> key. The HELP screen will give you some information on the purpose of the current screen and remind you of the redefined keys for the current block.

Help is always divided in two parts: a general presentation of the block and the list of redefined keys.

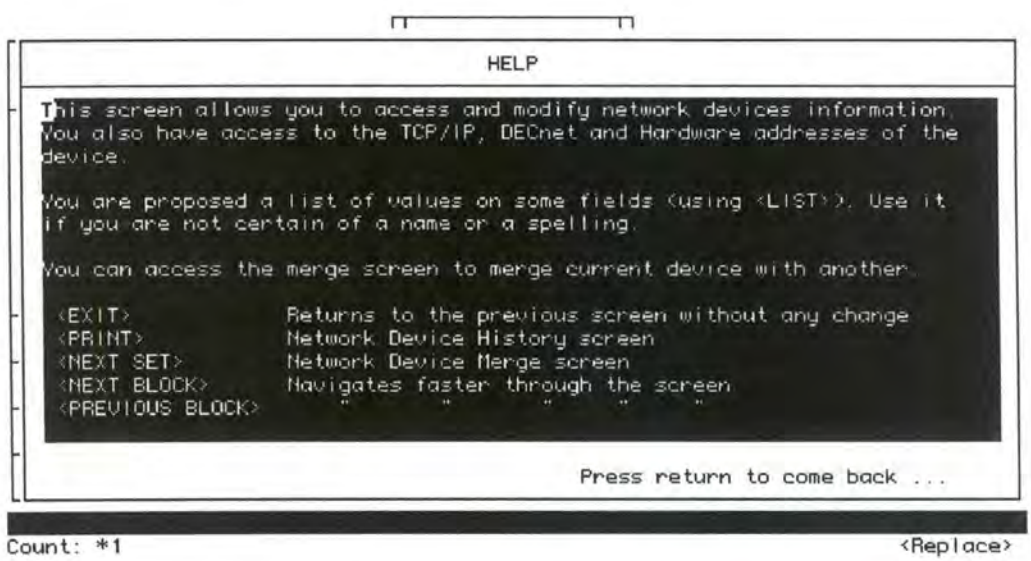

Fig. 6 : the help screen for network device block

2. Query, Insert, Update and Delete

2.1. Query

2.1.1. General use

Query allows you to select records from the database according to field values you have entered.

To enter a query you must first be in ENTER QUERY mode. It is sometimes automatically the case, so check the message line at the bottom of the screen. If it begins with "Enter a query;" then you are already in ENTER QUERY mode. If it does not then press <ENTER QUERY>.
When you are in ENTER QUERY mode, the normal keys don't work anymore. You can just enter search field values and navigate through the screen using <NEXT FIELD> and <PREVIOUS FIELD>.
When you have entered all the field values you are looking for, press <EXECUTE QUERY> to fetch all the records matching those values. You quit ENTER QUERY mode as soon as the query has executed. The message "Query caused no records to be retrieved" means that there are no records corresponding to your search. In such a case, check all values you have entered to see if there are any spelling errors.

2.1.2. Some tips

Wildcards

In non-numeric field, you are allowed to use wildcards. These are represented by the percent symbol (%). They mean "any character string".

Last query

When entering the search values in ENTER QUERY mode, if you press <ENTER QUERY> again then search values from the last query you entered will be displayed automatically.

Count query hits

If you want to know how many records match the search values, press <COUNT QUERY HITS>. The number of records which would be retrieved by the query will be displayed on the message line. You remain in ENTER QUERY mode so you can change your search field values and retry using <EXECUTE QUERY>.

Multiple purpose field

In this special screen you are allowed to enter different kinds of field values in the DEVICE NAME field. You can of course enter a Device Name value but also values for:
- TCP/IP hostname
- TCP/IP address
- TCP/IP alias
- DECnet V4 NCP node name
- DECnet V4 address
- DECnet V5 DNS name
- DECnet V5 node address
- DECnet V5 alias
- Hardware address

Wildcards are allowed for those values.
Multipurpose field is only possible in this screen and for this particular field. In other fields and screens, you are supposed to enter values in fields according to the field name. This has been done to avoid you to having navigate through the whole screen before being in the right field.
When using this special feature, you are warned by a message if the value you have entered in the DEVICE NAME field was not considered as a Device Name value and what it is (Example: "GIFT is a TCP/IP alias of VXGIFT").

2.1.3. Examples

Example 1

Imagine you are looking for all Network Devices with a name begining with "VSCS", in the TCP/IP area 128.141.254... and with TCP/IP software JNG.

1. Press <ENTER QUERY> if you are not already in ENTER QUERY mode (i.e. message line prompting "Enter a query;...").

2. Enter "VSCS%" in the DEVICE NAME field.
3. Press <NEXT FIELD> until you are on the TCP/IP SOFTWARE field.
4. Enter "JNG".
5. Press <NEXT FIELD> until you are on the TCP/IP ADDRESS first field (i.e. first byte).
6. Enter 128, 141 and 254 (the cursor will shift automatically; note that IP digits fields shift if they are fully entered, otherwise <NEXT FIELD>).

7. Press <EXECUTE QUERY>.
8. You can see all the records queried by pressing <NEXT RECORD> and <PREVIOUS RECORD>.

Example 2

If you want to query all records for TCP/IP addresses of the form 128.141.13.X

1. Press <ENTER QUERY> if you are not already in ENTER QUERY mode (i.e. message line prompting "Enter a query;...").

2. Enter "128.141.13%" in the DEVICE NAME field (multipurpose field).

3. Press <EXECUTE QUERY>.

2.2. Insert

2.2.1. General Use

Insert allows you to create new devices and new addresses in the database.

You must first be in INSERT MODE (i.e. in the NETWORK DEVICE block all fields are empty and it is not ENTER QUERY mode; in the ADDRESS blocks the address fields are empty). If you are in ENTER QUERY mode (when entering the screen, for example), press <EXIT> to enter INSERT mode.

If you are not in INSERT mode, just move the cursor on the block where you want to insert a record (by pressing <NEXT BLOCK> or <PREVIOUS BLOCK>) then press <INSERT RECORD>.

Note that the <INSERT RECORD> key only affects the current block.

You can insert several addresses for the same device before pressing <COMMIT> but you are not allowed to insert more than one device at a time without COMMITing the changes.

You must always have inserted or queried a network device before inserting any new address associated with it.

When in INSERT mode, enter field values and move from one field to another by pressing <NEXT FIELD> and <PREVIOUS FIELD> (or <NEXT BLOCK> and <PREVIOUS BLOCK> to go faster).

When you have finished entering all known field values, press <COMMIT> to insert all new records in the database.

If INSERT performs correctly, you will then have a "Transaction complete -- all records posted and committed." message.

If there is any mistake or problem, there will be messages explaining what was wrong and you will have to correct the mistakes before pressing <COMMIT> again or pressing <CLEAR FORM> or <EXIT> to cancel the operation. In case of any error, refer to the errors list (see chapter 4 for any error message ranging from 20000 to 20999).

If you were trying to insert an address which was already associated with another device, you will automatically enter the DEVICE MERGE screen (see below).

2.2.2 Some tips

- Inserting a new network device

If there was a network device displayed before you press <INSERT RECORD>, some field values will be duplicated in the new record assuming you're inserting the sqme type of machine, located in the same place,... In that case, be sure you give distinct device name, serial and order numbers.
It is recommended to use the lists of values (key <LIST>) for the mandatory fields MANUFACTURER and OPERATING SYSTEM (see section 3.1. for more information about the way to use list of values).
The list of value is also recommended when entering any personal field (responsible, contact name and firstname) as you will not be allowed to leave those pair of fields with unacceptable values, e.g. person unknown at CERN (see section 3.1. for more information about the way to use list of values).

- Inserting a new TCP/IP address

When you press <INSERT RECORD> in the TCP/IP block there will be default values displayed: you may accept them or change them. Note that accepting the values does NOT guarantee that <COMMIT> will work without error.

Hostname

Usually, the hostname is the same as the network device name. However if it has already been used by another TCP/IP address then you must use a different one. It is especially the case when default hostname is "?".

Domain

The default domain value is always ".CERN.CH". You can change it by going "backward" into that fiel using <PREVIOUS FIELD> from the first TCP/IP ADDRESS byte.

Address

The default address (if any) is usually the next address from the last displayed. It is not guaranteed that this address is free.
It is better to use <LIST> key to call the AVAILABLE TCP/IP AREA screen. That screen will allow you to choose a free TCP/IP address from the TCP/IP areas you own (see chapter 3 for more information on the way to use list of values).

- Inserting a new DECnet address

When you press <INSERT RECORD> in the DECnet block, there will be a default node name displayed: you may accept it or change it. Note that accepting the name does NOT guarantee that <COMMIT> will work without error.

Node name

Usually, the node name is the same as the network device name. However if it has already been used by another DECnet address, then you must use a different one. It is especially the case when the default node name is "?".

Domain

The default domain value is always ".CH.CERN". You can change it by going "backward" into that field using <PREVIOUS FIELD> from the first DECNET ADDRESS field (AREA).

Address

When you have entered the AREA, use <LIST> to see what is the next free NODE NUMBER in that AREA. You can press <LIST> several times to be shown different free node numbers.

- Inserting a new hardware address

Address

There is no default hardware address.
The hardware address must be made up of 6 hexadecimal bytes separated by any separator character (blank, hyphen,...) e.g. 08-00-2B-78-90-AB.

Address type

The default address type is always ETHERNET. You can change it by pressing <LIST> and choosing a new value into the list.

2.2.3. Examples

Example 1

If you want to insert a new device and its TCP/IP and DECnet addresses:

1. Press <CLEAR FORM> to be sure you begin with an empty form (no default values in the NETWORK DEVICE block)
2. Enter the device name
3. Press <NEXT FIELD>
4. Press <LIST> to open the MANUFACTURER list of values
5. With <NEXT RECORD> and <PREVIOUS RECORD> choose the manufacturer of the device and the device itself then press <COMMIT> to accept your choice.
6. Press <LIST> to open the OPERATING SYSTEM list of values
7. With <NEXT RECORD> and <PREVIOUS RECORD> choose the operating system and its version then press <COMMIT> to accept your choice.
8. Enter all the network device fields you have a value for, changing field with <NEXT FIELD> (if the field is not changed automatically).
9. Once in the TCP/IP ADDRESS block, press <INSERT RECORD> to create a new TCP/IP address (there will be default values).
10. Accept the default hostname or change it before pressing <NEXT FIELD>
11. On the first TCP/IP ADDRESS field, press <LIST> to have the list of all TCP/IP areas from where you can choose an address.
12. With <NEXT RECORD> and <PREVIOUS RECORD> choose the area where you want to choose the TCP/IP address and press <COMMIT> to accept your choice.
13. Press <NEXT FIELD> four times to move to the DECNET ADDRESS block.
14. Press <INSERT RECORD> to create a new DECnet address.
15. Accept the default node name or change it before pressing <NEXT FIELD>
16. Enter the AREA value.
17. Press <LIST> until you find a good node number in that area.
18. Press <COMMIT> to insert the 3 new records you have just created (one device and two addresses).
19. If there are any error messages, go back to point 3 or press <CLEAR FORM> or <EXIT> to cancel the operation.

Example 2

If you want to insert a new TCP/IP address for an existing device.

1. Press <CLEAR FORM> to be sure you begin with an empty form.
2. Query the network device to which you want to add a new TCP/IP address (see 2.1 to know how to query a device).
3. Press <NEXT BLOCK> until the cursor is in the TCP/IP block.
4. Press <INSERT RECORD> to create a new record.
5. Accept the default hostname or change it before pressing <NEXT FIELD>
6. On the first TCP/IP ADDRESS field, press <LIST> to have the list of all TCP/IP areas from where you can choose an address.
7. With <NEXT RECORD> and <PREVIOUS RECORD> choose the area where you want to choose the TCP/IP address then press <COMMIT> to accept your choice.
8. Press <COMMIT> to insert the 3 new records you have just created.
9. If there are any error messages, go back to point 2 or press <CLEAR FORM> or <EXIT> to cancel the operation.

2.3. Update

2.3.1. General Use

Update allows you to modify information about devices and addresses already recorded in the database. You must, of course, have queried existing records or inserted (and committed) new records before updating any information.

You can modify field values and move from one field to another by pressing <NEXT FIELD> and <PREVIOUS FIELD> , <NEXT BLOCK> and <PREVIOUS BLOCK> and from one device to another (or one address to another depending on the current block) by pressing <NEXT RECORD> and <PREVIOUS RECORD>.

As with INSERT mode, it is recommended to use <LIST> whenever possible.

When you have updated all the records, press <COMMIT> to accept the changes you have made.

If UPDATE performs correctly, you will have a "Transaction complete -- all records posted and committed." message.

If there is any mistake or problem, there will be messages explaining what was wrong and you will have to correct any mistakes before pressing <COMMIT> again or pressing <CLEAR FORM> or <EXIT> to cancel the operation. Refer to chapter 4 for any error message ranging from 20000 to 20999.

If you were trying to update an address by giving it a value which was already associated with another device, you will automatically enter the DEVICE MERGE screen (see chapter 3 for more information about this screen).

2.3.2 Some tips

- Updating an existing network device

It is recommended to use the lists of values (key <LIST>) for the mandatory fields MANUFACTURER and OPERATING SYSTEM (see section 3.1 for more information about the way to use list of values).

- Updating an existing TCP/IP address

Hostname

The hostname is usually the same as the network device name. However, if it has already been used by another TCP/IP address then you must use a different one.

Domain

The default domain value is always ".CERN.CH". You can change it by going "backward" into that field using <PREVIOUS FIELD> from the first TCP/IP ADDRESS byte.

Address

It is better to use <LIST> key to call the AVAILABLE TCP/IP AREA screen. That screen will allow you to choose a free TCP/IP address from your TCP/IP areas (see section 3.1 for more information on the way to use list of values).

- Updating an existing DECnet address

Node name

The node name is usuqlly the same as the network device name. However, if it has already been used by another DECnet address then you must use a different one.

Domain

The default domain value is always ".CH.CERN". You can change it by going "backward" into that field using <PREVIOUS FIELD> from the first DECNET ADDRESS field (AREA).

Address

When the AREA field is specified, use <LIST> to see what is the next free NODE NUMBER in that AREA. You can press <LIST> several times to be shown different free node numbers.

- Updating an existing hardware address

Address

The hardware address must be made up of 6 hexadecimal bytes separated by any separator character (blank, hyphen,...) e.g. 08-00-2B-78-90-AB.

Address type

The default address type is always ETHERNET. You can change it by pressing <LIST> and choosing a new value into the list.

2.3.3. Example

If you want to update an existing device and its TCP/IP and DECnet addresses:

1. Press <CLEAR FORM> to be sure you begin with an empty form.
2. Query the network device you want to update (see 2.1 to know how to query a device).
3. Modify all the network device, TCP/IP and DECnet fields you want, changing field with <NEXT FIELD>, <PREVIOUS FIELD>, <NEXT BLOCK> and <PREVIOUS BLOCK>.
4. When you have updated all the records, press <COMMIT> to accept the changes you have made.
5. If there are any error messages, go back on point 3 or press <CLEAR FORM> or <EXIT> to cancel operation.

2.4. Delete

2.4.1. General Use

Delete allows you to erase devices and addresses already recorded in the database. You must, of course, have queried existing records or inserted (and committed) records before deleting them.

Move the cursor onto the record you want to delete by using <NEXT FIELD>, <PREVIOUS FIELD>, <NEXT BLOCK>, <PREVIOUS BLOCK>, <NEXT RECORD> and <PREVIOUS RECORD>. Then press <DELETE RECORD>. The current record will be removed from screen. If it is a network device you want to remove, all addresses associated with it will be removed as well.

When you have removed all the records you wanted to, press <COMMIT> to delete them from the database.

If DELETE performs correctly, you will have a "Transaction complete -- all records posted and committed." message.

If there is any mistake or problem, there will be messages explaining what was wrong and you will have to correct any mistakes before pressing <COMMIT> again or pressing <CLEAR FORM> or <EXIT> to cancel the operation. Please refer to chapter 4 for any error message ranging from 20000 to 20999.

2.4.2. Examples

Example 1

If you want to delete an exisiting device and all its addresses:

1. Press <CLEAR FORM> to be sure you begin with an empty form.
2. Query the network device you want to delete (see section 2.1 to know how to query a device).
4. Press <DELETE RECORD> to remove the network device and all associated addresses from screen.
5. Press <COMMIT> to remove the device and all its adresses from the database.

Example 2

If you want to delete a TCP/IP address but not the device it is associated with:

1. Press <CLEAR FORM> to be sure you begin with an empty form.
2. Query the network device owning the TCP/IP address you want to delete (see section 2.1 to know how to query a device from its TCP/IP address).
3. Move the cursor onto the TCP/IP ADDRESS block by using <NEXT FIELD>, <PREVIOUS FIELD>, <NEXT BLOCK> and <PREVIOUS BLOCK>.
4. Press <NEXT RECORD> or <PREVIOUS RECORD> until you get displayed the TCP/IP address you want to delete.
4. Press <DELETE RECORD> to remove TCP/IP address from screen (note that network device and other addresses have not been removed).
5. Press <COMMIT> to remove the TCP/IP address from the database.

3. Some specific features

3.1. Lists of values

To help you enter correct values in certain fields, you will be shown a list of values. It is highly recommended to use them, since wrong spellings are not allowed in certain fields.

You can open a list of values by pressing key <LIST> but before doing so, when inserting a new record, you will be allowed to enter a search value, which will restrict the number of elements in the list of values (you can specify the first letter for example). If you have used a search value (then you have a restricted list), you can see ALL values again by pressing <EXECUTE QUERY>.

In the list, you can shift from one choice to another by using <NEXT RECORD> and <PREVIOUS RECORD>.

To accept a value and come back in the main screen, press <COMMIT>.

To exit a list of values without validating any choice, press <EXIT>.

Queries are possible in a list of values: press <ENTER QUERY> then enter you search values and <EXECUTE QUERY> like in any other screen.

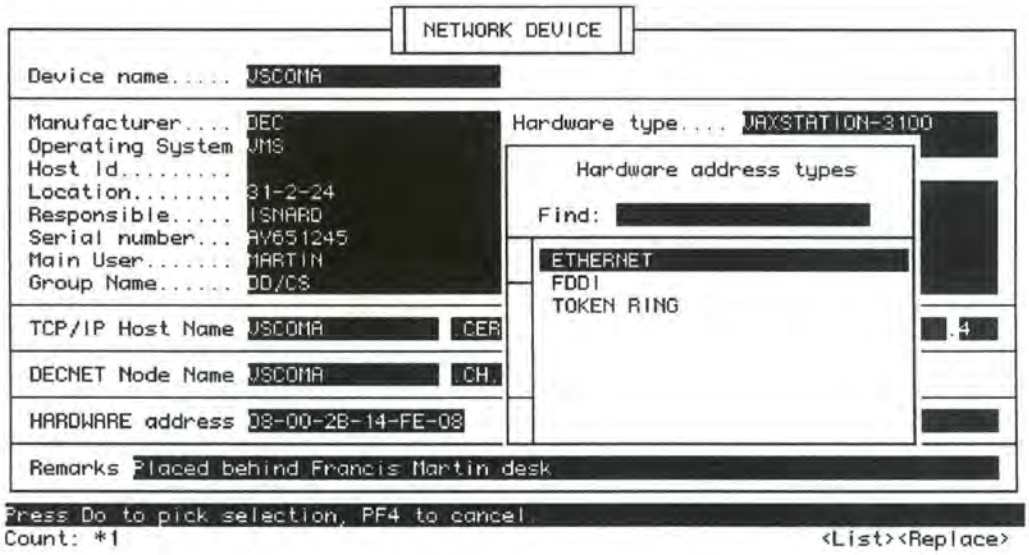

Fig. 7 : a list of values for hardware address type

Here are some explanations of three different types of list of values with special features:

- For HARDWARE TYPE and OPERATING SYSTEM:

Fig. 8 : a possible private list of values for Hardware type

The first list displayed is your *private list* of values. It contains the values you use most of the time. It is short and easy to use but not complete.

The "private" values are taken from a *common list* which contains all known values. It is big and not very easy to manipulate. You can call the common list from the private list by pressing <NEXT BLOCK> or <PREVIOUS BLOCK>. The private list is the only one from which you can choose values for the fields.

"Search values"

Before entering the private list of values, you can, if inserting a new record, enter a search value: for example, entering "DE" (wildcards are not needed at end) in the manufacturer field, then pressing <LIST> will restrict the private list of values to all values beginning with "DE" (all DEC machines in our example screen). Once in the private list of values, if you want to see all values, press <EXECUTE QUERY>.

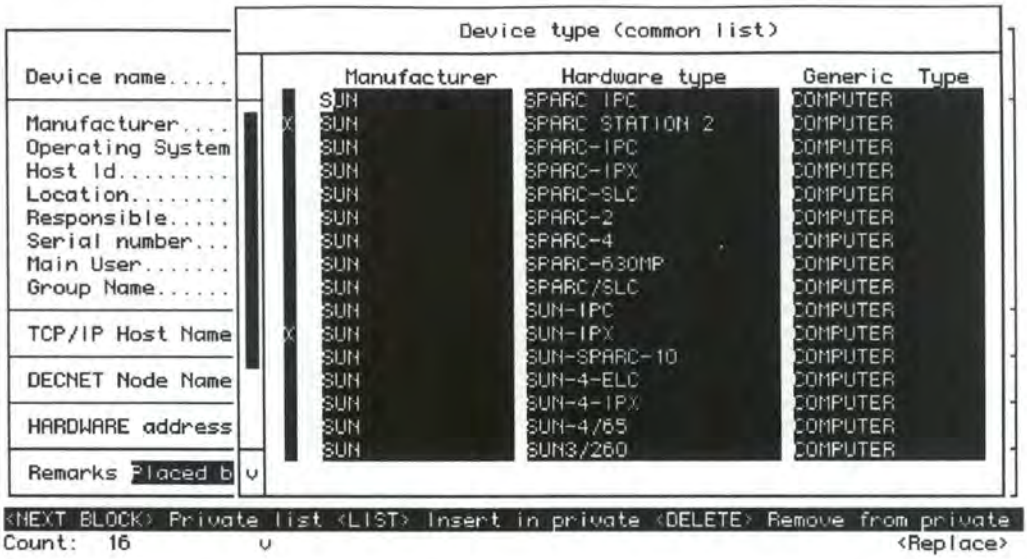

Fig. 9 : the common list of values for Operating Systems

From the common list, you will be able to insert or delete values in your private list. Values which are already in your private list are marked with a "X" sign. Values you've just removed are marked with a "-" sign and values you've just added are marked with "+". To accept the changes you've made in your private list from the common list, press <COMMIT>.

To go back to the private list without validating any choice, press <EXIT>, <NEXT BLOCK> or <PREVIOUS BLOCK>.

Note that all values are displayed in a common list of values even if the private list you come from is restricted.

- For RESPONSIBLE and CONTACT fields you will not be allowed to leave those pair of fields with unacceptable values (e.g. person not registered at CERN). It is therefore recommended to use the list of values.

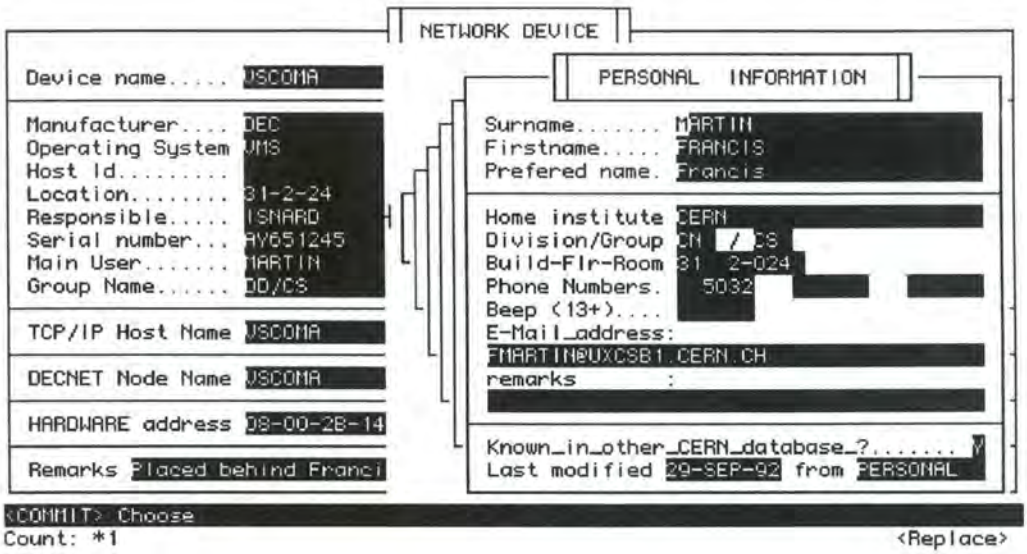

Fig. 10 : the PERSONAL INFORMATION list of values

- When inserting a new TCP/IP address, you will be proposed a list of values. It will show you all the TCP/IP areas you own. In this list of values, you will be able to compute the next free address of an area and the highest free address of an area (i.e. there are no used address above that one in the specific area). The highest free address is useful when entering a group of devices that must have consecutive addresses.

Pressing <COMMIT> in that list of values will select highest free address in the current area if there is one (even if it has not been computed) and pressing <PRINT> will select the next free address.

Computing the next free address can be done for ALL areas (using <LIST>) or only for current area (using <NEXT SET OF RECORDS>). Computing the free addresses of an area takes about 7 seconds.

3.2. The DEVICE MERGE screen

When you try to insert a new address (or update an existing one by giving it a value) which is already associated with another network device, the DEVICE MERGE screen will be displayed, assuming that you are perhaps trying to enter a device which is recorded twice in the database with different names.

That screen is divided into two columns:

- The column on the left, contains informations about the device you are inserting (updating) and which is going to be replaced by the existing device.
- The column on the right contains informations about the exisiting device which may or may not be the same as the one you are inserting (updating).

Fig. 11 : the DEVICE MERGE screen

If you accept to merge those two devices by pressing <COMMIT>, the device on the left will be replaced by the one on the right and ALL existing information associated with it will be associated with the right device (addresses, software, peripherals,etc).

Empty fields on the device on the right will be filled by the corresponding left hand device information. Priority is given to information for the device on the right.

If you do not wish to merge those two devices (in case of doubt especially), press <EXIT>. All "recordable" information about the new (or updated) device will be recorded in the database i.e. the address you were trying to insert (or update) initially will stay associated with the existing device, not the new (or updated) one.

3.3. The HISTORY

A history screen is associated with every block of the NETWORK DEVICES screen:

- Network device history
- TCP/IP address history
- DECnet address history
- Hardware address history

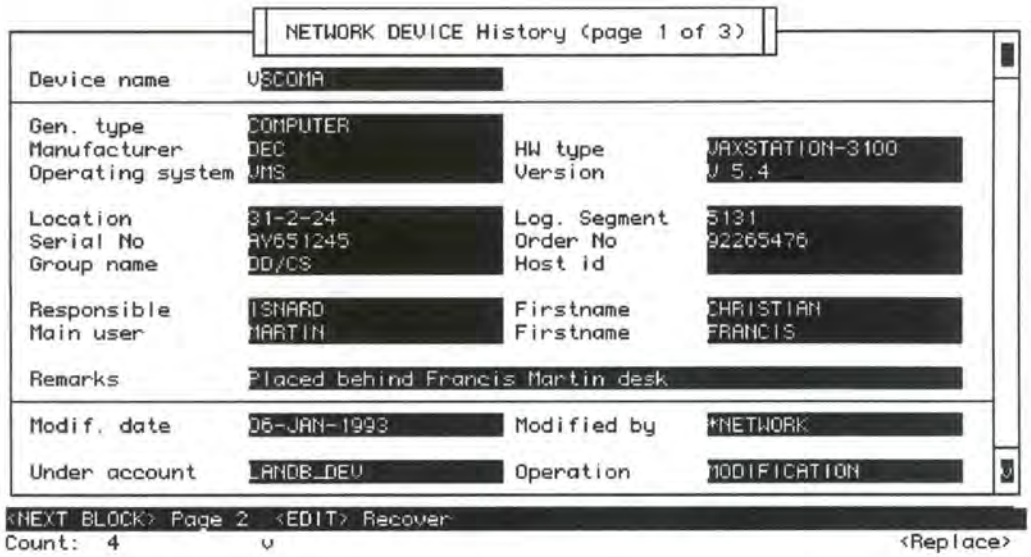

Fig. 12 : the NETWORK DEVICE HISTORY screen

Depending on the block where the cursor is, you can call the history screen by pressing <PRINT>. The history records every state of the device (or address) in the database. It is therefore possible to "go backward in time" to see all operations which have been made on a device (or an address). The history displays the operation made on the device (or address) and also the date of the operation, under which account and by who it has been made.

There are four types of operations: INSERT, UPDATE, DELETE and KEY UPDATE (change of identifier).

The first operation displayed is the most recent one. By pressing <PREVIOUS RECORD> (backward in time) and <NEXT RECORD> (forward in time), you can access all operations recorded in the database and see how the device (address) has changed.

To quit the history, press <EXIT>.

You are allowed to recover a specific state of the device (address) by pressing <EDIT>, but note that it is not always possible to recover a specific situation! For example, it will not be possible to recover a deleted address if the device it was associated with has also been deleted. In such a case you should first recover the network device before recovering its addresses.

NOTE: Before recovering a historical situation contact the user who has made the displayed operation to check if that will not bring any problem.

3.4. The addresses detail screens (EDIT)

For every ADDRESS block (TCP/IP, DECnet and Hardware), there is a special EDIT screen. That screen gives you more information about addresses than is displayed in the NETWORK DEVICE screen.
For example, in those screens, you can access information about aliases, areas and so on.

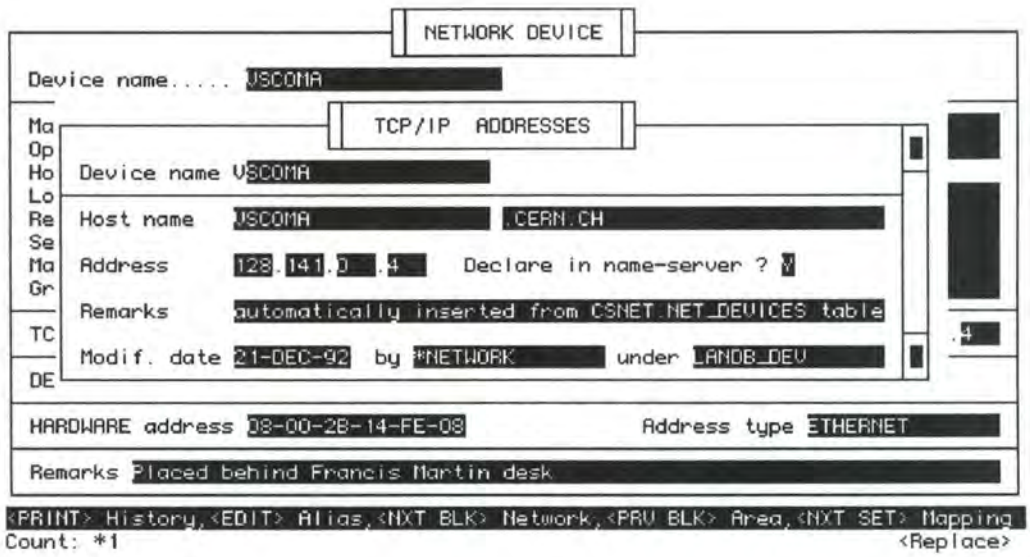

FIg. 13 : the TCP/IP ADDRESS EDIT screen

4. Specific application errors

Note: these errors (in range 20000-20999) are added to normal ORACLE and SQL*forms 3.0 errors. Any other error code should be referenced in ORACLE reference manuals.

ORA-20000: <any text>

Cause: When not preceded by any other message, the application raised an unexpected error which should not occur. May follow another application error (errors which number is between 20000 and 20999).

Action: If preceded of followed by any other application error, treat the cause of the other message. When not preceded by any other error message, fix the sequence of operations which raised the error and contact the application manager.

ORA-20001: Unable to insert <object>.

Cause: Generic error. An unspecified problem occured while inserting a record.

Action: If the message was not preceded by another message, fix the sequence of operations which raised the error and contact the application designer. If preceded of by any other application error, treat the cause of the other message.

ORA-20002: Unable to update <object>.

Cause: Generic error. An unspecified problem occured while updating a record.

Action: If the message was not preceded by another message, fix the sequence of operations which raised the error and contact the application designer. If preceded of by any other application error, treat the cause of the other message.

ORA-20003: Unable to delete <object>.

Cause: Generic error. An unspecified problem occured while deleting a row.

Action: If the message was not preceded by another message, fix the sequence of operations which raised the error and contact the application designer. If preceded of by any other application error, treat the cause of the other message.

ORA-20004: Unable to insert into history table.

Cause: Generic error, probably designer error. An internal and unspecified problem occured while inserting a record into a history table.
Action: Fix the sequence of operations which raised the error and contact the application designer.

ORA-20005: Record to be updated does not exists.

Cause: Generic error, probably a designer error. An internal and unspecified problem occured while updating a record.
Action: Fix the sequence of operations which raised the error and contact your application designer.

ORA-20006: Record to be deleted does not exists.

Cause: Generic error, probably a designer error. An internal and unspecified problem occured while deleting a record.
Action: Fix the sequence of operations which raised the error and contact your application designer.

ORA-20007: Unable to recover <object>.

Cause: Generic error. An unspecified problem occured when trying to recover an object from history table.
Action: If the message was not preceded by another message, fix the sequence of operations which raised the error and contact your application designer.

ORA-20008: Unable to merge <objects>.

Cause: Generic error. You are trying to merge two network devices, operating systems, device types or persons, but an internal error occured.
Action: When not preceded by any other message, fix the sequence of operations which raised the error and contact the application designer.

ORA-20009: <object> can not be merged with itself.

Cause: Generic error. You are trying to merge two network devices, operating systems, device types or persons, but you selected the same device twice !
Action: Select two different devices.

ORA-20101: Boot node <device> does not exists.

Cause: Invalid reference. Your are referring an incorrect machine while defining a logical group (cluster).
Action: Correct the machine you want to define as the boot node or principal device. Clear the record to cancel.

ORA-20102: Mail router <device> must be an existing machine.

Cause: Invalid reference. Your are referring an incorrect machine as the mail router for the device you are inserting or updating.
Action: Correct the machine you want to define or clear the field. Clear the record to cancel.

ORA-20103: Unknown network device <device>.

Cause: Invalid reference. Your are referring an unknown device.
Action: Check spelling or define the device first. Clear the record to cancel.

ORA-20104: Top-of-range computer <device> does not exists.

Cause: Invalid reference. Your are referring an incorrect machine while defining a top-of-range computer for an operating system.
Action: Correct the machine you want to define as the top-of-range computer, or define this device first. Clear the record to cancel.

ORA-20110: No TCP/IP address with such host name : <name>.

Cause: Invalid reference. Your are referring an incorrect TCP/IP address (when inserting a TCP/IP alias, for example).
Action: Correct the hostname. domain you are referring. Clear the record to cancel.

ORA-20120: DECnet V5 name <name> must correspond to at least one address.

Cause: Invalid reference. You are trying to directly insert or update a DNS (DEC phase V) name.
Action: No action required, DEC DNS name are automatically inserted when defining a DECnet phase V address. Clear the record to cancel.

ORA-20121: DECnet phase V name <name> does not exist.

Cause: Invalid reference. You are trying to define or modify a DNS alias, but the DNS name you are referring does not exist.
Action: Check spelling or clear the record to cancel.

ORA-20130: Unknown cable <cable>.

Cause: Invalid reference. You are trying to define or modify a connection of a device to a cable but the cable you are referring does not exist.
Action: Check spelling or insert cable first. Clear the record to cancel.

ORA-20140: Incorrect hardware address type <type>.

Cause: Invalid reference. You are defining or modifying an hardware address but the type you have specified is unknown.
Action: Check spelling, use a list of value or register the address type first. Clear the record to cancel.

ORA-20141: Contact person does not exists.

Cause: Invalid reference. Your are referring a person who is not registered in the CERN database.
Action: Use the list of known persons or check spelling. Clear the record to cancel.

ORA-20142: Incorrect device type <type>.

Cause: Invalid reference. You are trying to specify directly the device type (maker, hardware type and generic type) when inserting or updating a network device.
Action: Use the list of valid types or check spelling. Clear the record to cancel.

ORA-20143: Unknown operating system <operating system>.

Cause: Invalid reference. You are trying to specify directly the operating system (operating system and version) when inserting or updating a network device.
Action: Use the list of valid O/S or check spelling. Clear the record to cancel.

ORA-20144: Financial account <account> does not exists.

Cause: Invalid reference. You are trying to specify directly the financial account when inserting or updating a network device.
Action: Use the list of valid accounts or check spelling. Clear the record to cancel.

ORA-20145: User does not exists.

Cause: Invalid reference. You are referring a person who is not registered in one of the CERN databases.
Action: Use the list of known persons or check spelling. Clear the record to cancel.

ORA-20146: Unknown generic type <type>.

Cause: Invalid reference. You are defining a device type but the generic type is unknown.
Action: Check spelling or contact the application manager to enter a new generic type. Clear the record to cancel.

ORA-20147: Unknown ORACLE account <account>.

Cause: Invalid reference. you are trying to select a value in a private list or to give a privilege to a user whose account is not registered as a LANDB user.
Action: Correct spelling or contact your application manager to register the new user's account. Clear the record to cancel.

ORA-20148: Unknown external program <program>.

Cause: Invalid reference. You are trying to select an external program in your private list, but this program is not registered in the database.
Action: Correct spelling or define the external program first. Clear the record to cancel.

ORA-20149: Unknown APOLLO network <network>.

Cause: Invalid reference. You want to declare a node as a host of an APOLLO network, but the network is not registered in the database.
Action: Correct spelling or define the network first. Clear the record to cancel.

ORA-20150: This group (or cluster) <group> does not exist.

Cause: Invalid reference. You are refering a logical group (CERN group, cluster name or logical group) but this group is not registered in the database.
Action: Correct spelling or define the group first. Clear the record to cancel.

ORA-20160: Unknown product <product>.

Cause: Invalid reference. You are trying to reference a product which is not registered in the database.
Action: Check spelling or define product first. Clear the record to cancel.

ORA-20161: Unknown license number <license number>.

Cause: Invalid reference. You are trying to reference a licence which is not registered in the database.
Action: Check spelling or define licence first. Clear the record to cancel.

ORA-20162: Unknown package product <product>.

Cause: Invalid reference. You are trying to define a subproduct but the package product is not registered in the database.
Action: Check spelling or define package product first. Clear the record to cancel.

ORA-20170: Unknown product installation <installation>.

Cause: Invalid reference. You are trying to reference an installation which is not registered in the database (working on description or installation database table).
Action: Check spelling or define product installation first. Clear the record to cancel.

ORA-20180: Unknown peripheral or incompatible peripheral type.

Cause: Invalid reference. You are trying to define partitions for a peripheral which does not exists or the peripheral is not a hard disk.
Action: Check spelling, define the peripheral first or change peripheral type. Clear the record to cancel.

ORA-20181: Unknown peripheral type <type>.

Cause: Invalid reference. You are trying to define a peripheral but the peripheral type does not exist.
Action: Check spelling or define the peripheral type first. Clear the record to cancel.

ORA-20201: Bad hardware address length : <address>.

Cause: Domain error. The hardware address you are trying to insert or update is not seventeen characters length.
Action: Correct the corresponding hardware address (format is xx-xx-xx-xx-xx-xx, where x is a hexadecimal character.), or clear the record to cancel.

ORA-20202: Hardware address <address> must only contain hex. char.

Cause: Domain error. The hardware address you are trying to insert or update is not correct.
Action: Correct the corresponding hardware address (format is xx-xx-xx-xx-xx-xx, where x is a hexadecimal character.), or clear the record to cancel.

ORA-20203: Do not use special characters : !_@ ~` ^&*()+=[]{}:;"' |\. /?,<>

Cause: You use special characters when defining a host name or alias. Only alplhanumeric characters and '-' are allowed.
Action: Check the spelling or clear the record to cancel.

ORA-20204: Blanks are not allowed in a device name.

Cause: You used blanks when defining a device name. All characters (excpet blanks) are allowed in a device name.
Action: Check the spelling or clear the record to cancel.

ORA-20210: TCP/IP address byte must be between 1 and 254.

Cause: Domain error. The address you are trying to insert or update is not valid.
Action: Correct the TCP/IP address, or clear the record to cancel.

ORA-20211: Lower boundary is higher than upper boundary.

Cause: Domain error. You've made a mistake when defining a TCP/IP area.
Action: Correct the TCP/IP area, or clear the record to cancel.

ORA-20212: Address length incompatible with network class.

Cause: Domain error. You have specified too much or not enough address bytes for the class you specified when defining a TCP/IP network. For class A : ip1, class B : ip1. ip2, class C or S : ip1. ip2. ip3 .
Action: Correct the TCP/IP address or change the class; clear the record to cancel.

ORA-20213: TCP/IP network class must be in (A,B,C or S).

Cause: Domain error. You specified an invalid TCP/IP network class.
Action: Correct the class or clear the record to cancel.

ORA-20220: DECnet V4 node-number must be between 1 and 1023.

Cause: Domain error. The DECnet phase IV address you are trying to insert or update is not correct.
Action: Correct the DECnet address, or clear the record to cancel.

ORA-20270: Internal field must be either I or E.

Cause: Domain error. The internal field is an incorrect value.
Action: Correct field or clear the record to cancel.

ORA-20301: Can not delete Top-of-range computer.

Cause: Cascade error. You are trying to delete a device defined as a top-of-range computer for an ORACLE product (device used to calculate ORACLE licence price).
Action: Contact the application manager.

ORA-20325: DNS name <name> is still associated to addresses : can"t delete.

Cause: Cascade error. You are trying to delete a DNS name when addresses are still associated to.
Action: Nothing. DNS names will be automatically deleted when not in use (not associated to at least one DECnet V5 address).

ORA-20330: Devices are still connected to cable <cable> : cannot delete.

Cause: Cascade error. You are trying to delete a cable when devices are still connected to it.
Action: Delete connections first, or use a macro (special functionality) to delete the cable and connected devices.

ORA-20340: Hardware address of this type (<type>)still exists : cannot delete.

Cause: Cascade error. You are trying to delete a hardware address type when addresses of this type still exists.
Action: Delete addresses first, or use a macro (special functionality) to delete the type and corresponding addresses.

ORA-20341: Devices of this type still exist : cannot delete.

Cause: Cascade error. You are trying to delete a device type (from global list) when device of this type still exists.
Action: Change the type of correponding devices first, or use the device type merge facility.

ORA-20342: Devices type still selected in private lists : cannot delete.

Cause: Cascade error. You are trying to delete a device type (from global list) still selected in user's private list.
Action: Clean user's list first, or use the device type merge facility.

ORA-20343: O/S still selected in private lists : cannot delete.

Cause: Cascade error. You are trying to delete an Operating System (from global list) still selected in user's private list.
Action: Clean user's list first, or use the operating system merge facility.

ORA-20344: Devices with this O/S still exists: cannot delete.

Cause: Cascade error. You are trying to delete an Operating System (from global list) still referenced by devices.
Action: Correct devices first, or use the operating system merge facility.

ORA-20345: Products with this O/S still exists: cannot delete.

Cause: Cascade error. You are trying to delete an Operating System (from global list) still referenced by products.
Action: Correct obsolete info in product table first, or use the operating system merge facility.

ORA-20355: Hosts still exist. Can't delete APOLLO network <network>.

Cause: Cascade error. You are trying to delete an APOLLO network when devices are still connected to it.
Action: Delete connections first, or use a macro (special functionality) to delete the network and connections.

ORA-20361: This product (<product>) is still installed on devices.

Cause: Cascade error. You are trying to delete a product still installed on devices.
Action: Remove product installations first or use a macro.

ORA-20362: This product (<product>) is still selected in user's private lists.

Cause: Cascade error. You are trying to delete a product still selected in user's private lists of values.
Action: Remove product from private lists first or use a macro (if one exists).

ORA-20363: This product (<product>) is still associated to a license.

Cause: Cascade error. You are trying to delete a product but it is still associated to a license.
Action: Remove product-license link or use a macro (if one exists).

ORA-20371: Peripherals of this type still exist : cannot delete.

Cause: Cascade error. You are trying to delete a peripheral type when peripherals of this type still exists.
Action: Correct peripherals first.

ORA-20500: Name or alias <name> is already used by another machine.

Cause: Dupplicate error. The name (device name, host name or other) is already used by another device (as a device name, a host name or other).
Action: Choose another name, correct the other machine or merge the two devices.

ORA-20501: <object> <value> already exists.

Cause: Duplicate error. The object you are entering or updating is already defined. This application error corresponds to the ORACLE error : ORA-00001 : Duplicate value in index.
Action: Correct current data, correct the other information or merge the two devices.

ORA-20510: Alias cannot be the name itself.

Cause: You are trying to insert a TCP/IP or DNS alias already defined as the host name.
Action: Clear the record to cancel operation.

ORA-20511: Address <address> is already defined as an IP line local interface.

Cause: Duplicate error. This address is already defined (perhaps for the same machine) as an external TCP/IP line local interface (private address).
Action: Choose another address, correct the other machine or merge the two devices.

ORA-20512: Address <address> is already defined as an IP line distant addr.

Cause: Duplicate error. This address is already defined (perhaps for the same machine) as an external TCP/IP line distant interface (private address).
Action: Choose another address, correct the other machine or merge the two devices.

ORA-20513: Local address <address> already defined as a host address.

Cause: Duplicate error. This address is already defined (perhaps for the same machine) as an normal TCP/IP address.
Action: Choose another address, correct the other machine or merge the two devices.

ORA-20514: Distant address <address> already defined as a host address.

Cause: Duplicate error. This address is already defined (perhaps for the same machine) as an normal TCP/IP address.
Action: Choose another address, correct the other machine or merge the two devices.

ORA-20800: You are not allowed to modify TCP/IP address <address>.

Cause: You are trying to insert or modify a TCP/IP address or alias, but this address does not belong to one of yours TCP/IP area (address range associated to your ORACLE account).
Action: Contact your TCP/IP coordinator to have the access to this address.

ORA-20901: WARNING : This machine was the principal node for cluster <cluster>.

Cause: Cascade warning. The device you are deleting is defined as a main node for a cluster (Logical group).
Action: No action required, but note that the logical group (cluster) has no longer a principal node.

ORA-20902: WARNING : <object> already selected in your private list.

Cause: Information warning. From a common list of values, you are trying to choose a value already selected in your private list.
Action: No action required.

ORA-20903: WARNING : <object> already removed from your private list.

Cause: Information warning. From a common list of values, you are trying to remove a value which is not currently selected in your private list.
Action: No action required.

ORA-20904: INFORMATION : Record inserted into <table>.

Cause: Information message. This message is isued when trying to recover a record from history screens. It means that a deleted record was restored.
Action: No action required.

ORA-20905: INFORMATION : <object> updated.

Cause: Information message. This message is isued when trying to recover a record from history screens. It means that a previous state of a record was restored.
Action: No action required.

ORA-20906: WARNING : <object> is no longer valid - not recovered.

Cause: reference warning. This message is issued when trying to recover a record from history screens. A reference could not be restored because the corresponding information does not exist any longer.
Action: No action required, but reference is lost.

ORA-20910: WARNING : No free TCP/IP address in area <area>.

Cause: The TCP/IP area within which you are trying to allocate an address is full.
Action: Choose another TCP/IP area.

ORA-20911: WARNING : This TCP/IP area overlaps others areas.

Cause: The TCP/IP area you are creating or updating overlaps another area. It may be what you want to do.
Action: No action required. Use the TCP/IP area mapping screen to see overlaps.

ORA-20920: WARNING : No free DECnet address in <area> greater than <number>.

Cause: The DECnet V4 area within which you are trying to allocate an address is full, or there are no free address greater than the start number you specified.
Action: Decrease the start number or choose another DECnet V. 4 area.

5. VT220 Keyboard mapping

NAVIGATION

Next Field	Tab	<NEXT FIELD>
Down	Down	<DOWN>
Next Block	NextScreen	<NEXT BLOCK>
Next Set of Records	PF1 F9	<NEXT SET>
Next Primary Key	PF1 F10	
Previous Field	PF1 Return, PF1 TAB	<PREVIOUS FIELD>
Up	Up	<UP>
Previous Block	PrevScreen	<PREVIOUS BLOCK>
Next Record	Down	<NEXT RECORD>
Previous Record	Up	<PREVIOUS RECORD>
Scroll Down	PF1 Down	
Scroll Up	PF1 Up	

QUERY

Enter Query	F11	<ENTER QUERY>
Execute Query	F12	<EXECUTE QUERY>
Count Query Hits	PF3	

INSERT,UPDATE,DELETE

Insert Record	Insert	<CREATE RECORD>
Delete Record	Remove	<DELETE RECORD>
Duplicate Field	PF1 F11	
Duplicate Record	PF1 F12	

VALIDATION,ROLLBACK,CLEAR,EXIT

Accept/Commit	Do, <ctrl>O	<COMMIT>
Clear Field	F17	<CLEAR FIELD>
Clear Record	F18	<CLEAR RECORD>
Clear Block	F19	<CLEAR BLOCK>
Clear Form/Rollback	F20	<CLEAR FORM>
Exit	PF4, <ctrl>B	<EXIT>

EDIT

Edit	PF2	<EDIT>
Insert/Replace	<ctrl>A	
Left	Left	
Right	Right	
Scroll Left	PF1 Left	
Scroll Right	PF1 Right	
Beginning of Line	PF1 PF1 Left	
End of Line	PF1 PF1 Right	
First Line	PF1 PF1 Up	
Last Line	PF1 PF1 Down	
Cut	PF1 Remove	
Copy	PF1 Find	
Paste	PF1 Insert	
Delete Backwards	Backspace	
Delete Character	<ctrl>D	
Delete Line	PF1 Delete	
Clear End-of-Line	PF1 PF1 Remove	
Return	Return	

OTHERS

Block Menu	PF1 F14	
Menu	F14, <ctrl>G,<ctrl>N	<MENU>
List	Find, <ctrl>F	<LIST >
Display Error	PF1 Help	
Print	<ctrl>P	<PRINT>
Refresh	<ctrl>R	<REFRESH>
Show Keys	<ctrl>K	
Select	Select, <ctrl>V	
Help	Help, <ctrl>W	<HELP>

5. Macintosh emulation of VT220 keyboard

(using NCSA-BUY - Telnet 2.5)

NAVIGATION

Next Field	Tab, Return, Enter	<NEXT FIELD>
Down	Down	<DOWN>
Next Block	PageDown	<NEXT BLOCK>
Next Set of Records	NumLock F4	<NEXT SET >
Next Primary Key	NumLock F5	
Previous Field	NumLock Return	<PREVIOUS FIELD>
Up	Up	<UP>
Previous Block	End	<PREVIOUS BLOCK>
Next Record	Down	<NEXT RECORD>
Previous Record	Up	<PREVIOUS RECORD>
Scroll Down	NumLock Down	
Scroll Up	NumLock Up	

QUERY

Enter Query	F6	<ENTER QUERY>
Execute Query	F7	<EXECUTE QUERY>
Count Query Hits	/ (on keypad)	

INSERT,UPDATE,DELETE

Insert Record	Home	<CREATE RECORD>
Delete Record	PageUp	<DELETE RECORD>
Duplicate Field	NumLock F6	
Duplicate Record	NumLock F7	

VALIDATION,ROLLBACK,CLEAR,EXIT

Accept/Commit	F11	<COMMIT>
Clear Field	F12	<CLEAR FIELD>
Clear Record	F13	<CLEAR RECORD>
Clear Block	F14	<CLEAR BLOCK>
Clear Form/Rollback	F15	<CLEAR FORM>
Exit	* (on keypad)	<EXIT>

EDIT

Edit	= (on keypad)	<EDIT>
Insert/Replace	<ctrl>A	
Left	Left	
Right	Right	
Scroll Left	NumLock Left	
Scroll Right	NumLock Right	
Beginning of Line	NumLock NumLock Left	
End of Line	NumLock NumLock Right	
First Line	NumLock NumLock Up	
Last Line	NumLock NumLock Down	
Cut	NumLock PageUp	
Copy	NumLock Help	
Paste	NumLock Home	
Delete Backwards	Backspace	
Delete Character	<ctrl>D	
Delete Line	NumLock Backspace	
Clear End-of-Line	NumLock NumLock PageUp	
Return	Return	

OTHERS

Block Menu	NumLock F9	
Menu	F9, <ctrl>G,<ctrl>N	<MENU>
List	Help, <ctrl>F	<LIST >
Display Error	NumLock F10	
Print	<ctrl>P	<PRINT>
Refresh	<ctrl>R	<REFRESH>
Show Keys	<ctrl>K	
Select	Delete, <ctrl>V	
Help	F10, <ctrl>W	<HELP>

Annexe 3:

Evaluation des interfaces de gestion des adresses auprès des utilisateurs

Cette annexe reprend le formulaire d'enquête qui a été envoyé aux premiers utilisateurs des interfaces de gestion de l'adressage de LANDB, à savoir 6 utilisateurs.
Le formulaire est suivi des réponses reçues au moment de l'impression de ce mémoire.
La méthode d'évaluation est celle décrite dans [RAV89] a quelques modifications près de manière à l'adapter aux spécificités de LANDB.

Dear User,

Olivier Francis Martin has sent me a list of some of the first users of the system and that's the reason I send you this mail. I would like to ask you to fill in the evaluation you can find at the end of this mail. I know you have a lot of work but if you can spend some of your precious time it would help greatly.

I think it would be also useful to the LANDB developpers to improve the User Interface in a future version of LANDB and I will send the results of this evaluation to Olivier Francis Martin.

The goal of this evaluation is to judge the User Interface of the LANDB Database. It is important to stress that it is the LANDB User Interface that is under evaluation, not yourself.

As you will see the evaluation is divided into several sections, each focusing in the issue that stands as its title. These issues try to cover the aspects of the user interface I think are relevant for the LANDB User Interface.

If you have any doubts either on how to perform the task or on how to fill in the checklist, feel free to ask (fbi@info.fundp.ac.be). One important request I have about the checklist is: if you don't understand some question or think it is confusing write in the comments area.

You can either put crosses (X) in the forms below or send me a new mail containing your answers like this:

```
A. 1. S
   2. M   Question A.2 comments
   3. N
   4. N   Question A.4 comments
   5. S

and so on...
```

When done, please send it back to fbi@info.fundp.ac.be.

If you do not have any time to spend on this evaluation, please let me know. But if you do, would you please send it back as soon as possible.

I thank you very much for your cooperation.

Francois Briard

(Aug '92 - Jan '93 technical student)
e-mail: fbi@info.fundp.ac.be

The user interface evaluation is based on the method presented in Ravden S., Johnson G., "Evaluating usability of human-computer interfaces: a practical method",Ellis Horwood. It has been adapted to fit LANDB specific environnement.

LANDB USER INTERFACE EVALUATION
==========

GENERAL INFORMATION

What is your function as user of the LANDB application (TCP/IP manager, responsible for entering specific information...)?

What is your experience concerning that function's domain?

Have you ever used other applications to perform the same function?

Describe briefly the tasks you perform on LANDB.

How often do you use the LANDB application?

How do you connect the LANDB database (from/through which kind of computer)?

Under which environnement do you run LANDB (Terminal, Operating system...)?

What is your experience on systems with SQL*forms interfaces?

A. VISUAL CLARITY

Information displayed on the screen should be clear, well-organized, unambiguous and easy to read.

Key: (A)lways, (M)ost of the time, (S)ometime, (N)ever

	A	M	S	N	Comments
1. Is each form (screen/box) identified with an informative title or description?					
2. When the user enters information on the screen, is it clear:					
a) where the information should be entered?					
b) in what format it should be entered?					
3. Does information appears to be organized logically in the screens?					
4. Are different types of information clearly separated from each other on the screen?					
5. Where a large amount of information is displayed on one screen, is it clearly separated into sections on the screen?					
6. Do forms appear to confuse by crowding?					
7. Is the information on the screen easy to see and to read?					
8. Is it easy to find the required information on a screen?					

9. Are there any comments (good or bad) you wish to add regarding the above issues?

10. Overall, how would you rate the system in terms of visual clarity?

( ) Very satisfatory
( ) Moderately satisfatory
( ) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

B. CONSISTENCY

The way the system looks and works should be consistent at all times.

	A	M	S	N	Comments
1. Are abbreviations, acronyms, codes and other alphanumeric information used consistently throughout the system?					
2. Is the same type of information (e.g. menus, messages, titles) displayed:					
a) in the same location on the screen?					
b) in the same layout?					
3. Is the method of entering information consistent throughout the system?					
4. Is the method of selecting options (e.g. calling other forms) consistent throughout the system?					
5. Is the way the system responds to a particular user action consistent all the time?					
6. Are the same keys used for the same functions throughout the system?					
7. Is the guidance provided to the user consistent throughout the system?					
8. Is the same action (e.g. create, update, delete) done in a consistent way for different elements?					

9. Are there any comments (good or bad) you wish to add regarding the above issues?

10. Overall, how would you rate the system in terms of consistency?

( ) Very satisfatory
( ) Moderately satisfatory
( ) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

C. COMPATIBILITY

The way the system looks and works should be compatible with user conventions and expectations.

	A	M	S	N	Comments
1. Where abreviations, acronyms and others alphanumeric information are displayed: a) are they easy to recognize and understand? b) do they follow conventions when these exist?					
2. Where jargon and terminology is used within the system, is it familiar to the user?					
3. Are established conventions followed for the format in which particular types of information are displayed? (e.g. layout of dates and telephone numbers)					
4. Is the format of displayed information compatible with the form in which it is entered into the system?					
5. Is information presented in a way which fits the user's view of the task?					
6. Does the sequence of activities required to complete a task follow what the user would expect?					
7. Does the system work in the way the user thinks it should work?					

8. Are there any comments (good or bad) you wish to add regarding the above issues?

9. Overall, how would you rate the system in terms of compatibility?

( ) Very satisfatory
( ) Moderately satisfatory
( ) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

D. INFORMATIVE FEEDBACK

Users should be given clear, informative feedback on where they are in the system, what actions they have taken, whether these actions have been successful and what actions should be taken next.

	A	M	S	N	Comments
1. Are instructions and messages displayed by the system concise and positive?					
2. Are messages displayed by the system relevant?					
3. Do instructions/prompts clearly indicates what to do?					
4. Is it clear what actions the user can take at any stage?					
5. Is it clear what the user needs to do in order to take a particular action? (e.g. which options to select, which key to press)					
6. Is it made clear what changes occur on the screen as a result of a user input action?					
7. Is there always an appropriate system response to a user input or action?					
8. Are status messages (e.g. indicating what the system is doing, or has just done): a) informative b) accurate					
9. Does the system clearly inform the user when it completes a requested action (successfully or unsuccessfully)?					
10. Do error messages explain clearly: a) where the errors are? b) what the errors are? c) why they have occured?					
11. Is it clear to the user what should be done to correct an error?					
12. Does the system clearly indicates which mode the user is currently in? (e.g. query)					

13. Are there any comments (good or bad) you wish to add regarding the above issues?
14. Overall, how would you rate the system in terms of informative feedback?

( ) Very satisfatory
( ) Moderately satisfatory
( ) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

E. EXPLICITENESS

The way the system works and is structured should be clear to the user.

	A	M	S	N	Comments
1. Is it clear what stage the system has reached in a task?					
2. Is it clear what the user needs to do in order to complete a task?					
3. Where the user is presented with a list of options, is it clear what each option means?					
4. Is it clear what part of the system the user is in?					
5. Is it clear what the different parts of the system do?					
6. Is it clear how, where and why changes in one part of the system affect other parts of the system?					
7. Is it clear why the system is organized and structured as it is?					
8. Is it clear why a series of screens are sequenced as they are?					
9. Is the structure of the system obvious to the user?					
10. Is the system well-organized from the user's point of view?					
11. In general, is it clear what the system is doing?					

12. Are there any comments (good or bad) you wish to add regarding the above issues?

13. Overall, how would you rate the system in terms of expliciteness?

( ) Very satisfatory
( ) Moderately satisfatory
( ) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

F. APPROPRIATE FUNCTIONALITY

The system should meet the needs and requirements of users when carrying out tasks.

	A	M	S	N	Comments
1. Is the input device (i.e. keyboard) appropriate for the tasks to be carried out?					
2. Is the way in which information is presented appropriate for the tasks?					
3. Does each screen contain all the information which the user feels is relevant to the task?					
4. Are users provided with all the options which they feel are necessary at any particular stage in a task?					
5. Can users access all the information which they feel they need for the current task?					
6. Does the system allow users to do what they feel is necessary in order to carry out a task?					
7. Is system feedback appropriate for the task?					
8. Do the content of help facility make use of realistic task data and context?					
9. Is task-specific jargon and terminology defined at an early stage in the task?					
10. Where task sequences are particularly long, are they broken into appropriate subsequences?					

11. Are there any comments (good or bad) you wish to add regarding the above issues?

12. Overall, how would you rate the system in terms of appropriate functionality?

( ) Very satisfatory
( ) Moderately satisfatory
( ) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

G. FLEXIBILITY AND CONTROL

The interface should be sufficiently flexible in structure, in the way info is presented and in terms of what the user can do.
It should allow users to feel in control of the system.

	A	M	S	N	Comments
1. Is there an easy way for the users to 'undo' an action and step back to a previous stage or screen?					
2. Where the user can 'undo', is it possible to 'redo' (i.e. to reverse this action)?					
3. Are shortcuts available when required?					
4. Do users have control over the order in which they request information, or carry out a series of activities?					
5. Can the user move to the different parts of the system? And is it easy to move?					
6. Does the system prefill repeated information to save the user having to enter the same information several times?					
7. Can the user choose wether to enter information manually or to let the computer generate information automatically? (e.g. where there are defaults)					
8. Can the user override computer-generated (e.g. default) information, if appropriate?					
9. Can the user tailor certain aspects of the interface for their own preferences?					

10. Are there any comments (good or bad) you wish to add regarding the above issues?

11. Overall, how would you rate the system in terms of flexibility and control?

( ) Very satisfatory
( ) Moderately satisfatory
( ) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

H. ERROR PREVENTION AND CORRECTION

The system should minimize the possibility of errors, facilitate their detection and handling. Users should be able to check their inputs and to correct errors before the input is processed.

	A	M	S	N	Comments
1. Does the system clearly and promptly inform the user when it detects an error?					
2. Is there some form of cancel key for the user to reverse an error situation?					
3. Is it easy for the user to correct errors?					
4. Does the system validate user inputs before processing wherever possible?					
5. Are users able to check what they have entered before it is processed?					
6. Is the system protected against common trivial errors (e.g. wrong keys hit)					
7. Does the system ensure that the user double checks any requested actions which may be catastrophic if requested unintentionnaly? (e.g. large-scale deletion)					
8. Does the system prevent users from taking actions they are not authorized to take?					
9. In general, is the system free from errors and malfunctions?					

10. Are there any comments (good or bad) you wish to add regarding the above issues?

11. Overall, how would you rate the system in terms of error prevention and correction?

( ) Very satisfatory
( ) Moderately satisfatory
( ) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

I. USER GUIDANCE AND SUPPORT

Informative, easy-to-use and relevant guidance and support should be provided (via an on-line help facility and in hard-copy document form) to help the user understand and use the system.

	A	M	S	N	Comments
1. Can the user request the help facility easily from any point in the system?					
2. Is it clear how to get in and out of the help system?					
3. Is the help info presented clearly, without interfering with the user's current activity?					
4. When the user requests help, does the system clearly explain the possible actions which can be taken, in the context of what the user is doing?					
5. Does the help allow the user to browse through information about other parts of the system?					
6. Does the user guide an in-depth, comprehensive description?					
7. Is it easy to find a particular section in the user guide?					
8. Is the organization of all forms of support related to the tasks which the user can carry out?					
9. Do user support facilities adequately explain user and system errors and how these should be corrected?					
10. Are all forms of user support maintained up-to-date?					

11. Are there any comments (good or bad) you wish to add regarding the above issues?

12. Overall, how would you rate the system in terms of user guidance and support?

( ) Very satisfatory
( ) Moderately satisfatory
( ) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

J. SYSTEM USABILITY PROBLEMS

When using the system, did you experience problems with any of the following?

Key: (N)o problems, (Mi)nor problems, (Ma)jor problems

	N	Mi	Ma	Comments
1. Working out how to use the system				
2. Lack of guidance on how to use the system				
3. Poor system documentation				
4. Understanding how to carry out the tasks				
5. Knowing what to do next				
6. Understanding how the information on the screen relates to what you are doing				
7. Finding the information you want				
8. Information which is difficult to read clearly				
9. An inflexible, rigid structure				
10. An inflexible help (guidance) facility				
11. Losing track of where you are in the system or of what you are doing or have done				
12. Having to remember too much information while carrying out a task				
13. System response times that are too quick for you to understand what is going on				
14. Information which does not stay on the screen long enough for you to read				
15. System response times that are too slow				
16. Unexpected actions by the system				
17. An input device that is difficult or awckward to use				
18. Knowing where or how to input the information				
19. Having to spend too much time inputting information				
20. Having to be very careful in order to avoid errors				

21. Working out how to correct errors				
22. Having to spend to much time correcting errors				
23. Having to carry out the same type of activity in different ways				

K. GENERAL QUESTIONS ON SYSTEM USABILITY

1. What are the best aspects of the system for the user?

2. What are the worst aspects of the system for the user?

3. Are there any parts of the system which you found confusing or difficult to fully understand?

4. Were there any aspects of the system which you found particularly irritating although they did not cause major problems?

5. What were the most common mistakes you made when using the system?

6. What changes would you make to the system to improve it from the user's point of view?

7. Is there anything else about the system you would like to add?

L. GENERAL QUESTIONS ON THE DATABASE ITSELF

Do you have any comments on the LANDB database itself (content, structure...)?

Reponse reçue d'un utilisateur

A. VISUAL CLARITY

Information displayed on the screen should be clear, well-organized, unambiguous and easy to read.

Key: (A)lways, (M)ost of the time, (S)ometime, (N)ever

	A	M	S	N	Comments
1. Is each form (screen/box) identified with an informative title or description?	X				
2. When the user enters information on the screen, is it clear:					
a) where the information should be entered?	X				
b) in what format it should be entered?				X	
3. Does information appears to be organized logically in the screens?		X			
4. Are different types of information clearly separated from each other on the screen?		X			
5. Do forms appear to confused by crowding?				X	
6. Is the information on the screen easy to see and to read?		X			
7. Is it easy to find the required information on a screen		X			

8. Are there any comments (good or bad) you wish to add regarding the above issues?

An X-windows interfaces !

9. Overall, how would you rate the system in terms of visual clarity?

( ) Very satisfatory
(X) Moderately satisfatory
( ) Neutral
(X) Moderately unsatisfactory (comparing to other interfaces)
( ) Very unsatisfatory

B. CONSISTENCY

The way the system looks and works should be consistent at all times.

	A	M	S	N	Comments
1. Is the same type of information (e.g. menus, messages, titles) displayed:					
a) in the same location on the screen?	X				
b) in the same layout?		X			
2. Is the method of entering information consistent throughout the system?	X				
3. Is the method of selecting options (e.g. calling other forms) consistent throughout the system?		X			
4. Is the way the system responds to a particular user action consistent all the time?		X			
5. Is the guidance provided to the user consistent throughout the system?		X			
6. Is the same action (e.g. create, update, delete) done in a consistent way for different elements?	X				

7. Are there any comments (good or bad) you wish to add regarding the above issues?

8. Overall, how would you rate the system in terms of consistency?

(X) Very satisfatory (for a private application)
(X) Moderately satisfatory (in general)
( ) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

C. COMPATIBILITY

The way the system looks and works should be compatible with user conventions and expectations.

	A	M	S	N	Comments
1. Where abreviations, acronyms and others alphanumeric information are displayed:					
a) are they easy to recognize and understand?			X		cabling
b) do they follow conventions when these exist?		X			
2. Where jargon and terminology is used within the system, is it familiar to the user?	X				they are experts
3. Is the format of displayed information compatible with the form in which it is entered into the system?		X			
4. Does the sequence of activities required to complete a task follow what the user would expect?			X		
5. Does the system work in the way the user thinks it should work?			X		

6. Are there any comments (good or bad) you wish to add regarding the above issues?

Not easy to use. Was designed for experts.

7. Overall, how would you rate the system in terms of compatibility?

( ) Very satisfatory
(X) Moderately satisfatory
( ) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

D. INFORMATIVE FEEDBACK

Users should be given clear, informative feedback on where they are in the system, what actions they have taken, whether these actions have been successful and what actions should be taken next.

	A	M	S	N	Comments
1. Are instructions and messages displayed by the system concise and positive?		X			
2. Are messages displayed by the system relevant?		X			
3. Do instructions/prompts clearly indicates what to do?			X		
4. Is it clear what actions the user can take at any stage?		X			
5. Is it clear what the user needs to do in order to take a particular action? (e.g. which options to select, which key to press)				X	
6. Is it made clear what changes occur on the screen as a result of a user input action?			X		
7. Is there always an appropriate system response to a user input or action?		X			
8. Are status messages (e.g. indicating what the system is doing, or has just done):					
a) informative	X				
b) accurate	X				
9. Do error messages explain clearly:			X		
a) where the errors are?		X			
b) what the errors are?		X			
c) why they have occured?		X			

10. Are there any comments (good or bad) you wish to add regarding the above issues?

Not action oriented : a user may perform a lot of different actions and The system only gives a feedback for the internal actions the user has ready done (from its point of view). It does not explain how to perform a basic action from the user's point of view (like registering a new device, changing the location (with reponsible, physical connection...) of a device). The user guide does so (but it's off-line). There is a kind of logical gap.

11. Overall, how would you rate the system in terms of infomrative feedback?

( ) Very satisfatory
(X) Moderately satisfatory
( ) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

E. EXPLICITENESS

The way the system works and is structured shoula be clear to the user.

	A	M	S	N	Comments
1. Is it clear what stage the system has reached in a stage?			X		
2. Is it clear what the user needs to do in order to complete a task?			X		
3. Where the user is presented with a list of options, is it clear what each option means?	X				rare
4. Is it clear what the different parts of the system do?			X		
5. Is it clear how, where and why changes in one part of the system affect other parts of the system?			X		
6. Is it clear why the system is organized and structured as it is?		X			
7. Is it clear why a series of screens are sequenced as they are?		X			
8. Is the system well-organized from the user's point of view?		X			seems complex
9. In general, is it clear what the system is doing?			X		

10. Are there any comments (good or bad) you wish to add regarding the above issues?

same thing as above

11. Overall, how would you rate the system in terms of expliciteness?

( ) Very satisfatory
( ) Moderately satisfatory
(X) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

F. APPROPRIATE FUNCTIONALITY

The system should meet the needs and requirements of users when carrying out tasks.

	A	M	S	N	Comments
1. Is the input device (i.e. keyboard) appropriate for the tasks to be carried out?		X			use of emulators
2. Is the way in which information is presented appropriate for the tasks?		X			long study
3. Does each screen contain all the information which the user feels is relevant to the task?		X			long study
4. Can users access all the information which they feel they need for the current task?		X			
5. Does the system allow users to do what they feel is necessary in order to carry out a task?	X				sometimes more !
6. Is system feedback appropriate for the task?		X			see above
7. Do the content of help facility make use of realistic task data and context?		X			
8. Is task-specific jargon and terminology defined at an early stage in the task?					???
9. Where task sequences are particularly long, are they broken into appropriate subsequences?		X			

10. Are there any comments (good or bad) you wish to add regarding the above issues?
The system lets the users perform a lot of mistakes entering data (but does not allow the changes) It requires quite a long time to learn how to perform what you want.

11. Overall, how would you rate the system in terms of appropriate functionality?

(X) Very satisfatory
( ) Moderately satisfatory
( ) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

G. FLEXIBILITY AND CONTROL

The interface should be sufficiently flexible in structure, in the way info is presented and in terms of what the user can do.
It should allow users to feel in control of the system.

	A	M	S	N	Comments
1. Is there an easy way for the users to 'undo' an action and step back to a previous stage or screen?		X			
2. Where the user can 'undo', is it possible to 'redo' (i.e. to reverse this action)?				X	
3. Are shortcuts available when required?		X, but not procedural			
4. Do users have control over tasks that they would like to have? And over order in which they perform tasks (e.g. the order they request information)?			S		
5. Can the user move to the different parts of the system? And is it easy to move?	X				every user may read everything
6. Does the system prefill repeated information to save the user having to enter the same information several times?		x			
7. Can the user choose wether to enter information manually or to let the computer generate information automatically? (e.g. where there are defaults)	X				
8. Can the user override computer-generated (e.g. default) information, if appropriate?	X				
9. Can the user tailor certain aspects of the interface for their own preferences?				X	

10. Are there any comments (good or bad) you wish to add regarding the above issues?

11. Overall, how would you rate the system in terms of flexibility and control?

( ) Very satisfatory
(X) Moderately satisfatory
( ) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

H. ERROR PREVENTION AND CORRECTION

The system should minimize the possibility of errors, facilitate their detection and handling. Users should be able to check their inputs and to correct errors before the input is processed.

	A	M	S	N	Comments
1. Does the system clearly and promptly inform the user when it detects an error?	X				
2. Is there some form of cancel kay for the	X				

	user to reverse an error situation?					
3.	Is it easy for the user to correct errors?			X		lost
4.	Does the system validate user inputs before processing wherever possible?		X			
5.	Are users able to check what they have entered before it is processed?	X				
6.	Is the system protected against common trivial errors (e.g. wrong keys hit)	X				
7.	Does the system ensure that the user checks any requested actions which may be catastrophic if requested unintentionnaly? (e.g. large scale deletion)				X	1 record processed at a time in general
8.	Does the system prevent users from taking actions they are not authorized to take?		X	X		external protection system
9.	In general, is the system free from errors and malfunctions?		X			

10. Are there any comments (good or bad) you wish to add regarding the above issues?

11. Overall, how would you rate the system in terms of error prevention and correction?

    ( ) Very satisfatory
    (X) Moderately satisfatory
    ( ) Neutral
    ( ) Moderately unsatisfactory
    ( ) Very unsatisfatory

I. USER GUIDANCE AND SUPPORT

Informative, easy-to-use and relevant guidance and support should be provided (via an on-line help facility and in hard-copy document form) to help the user understand and use the system.

	A	M	S	N	Comments
1. Can the user request the help facility easily from any point in the system?	X				
2. Is it clear how to get in and out of the help system?	X				
3. Is the help info presented clearly, without interfering with the user's current activity?				N	24*80 screen !!
4. When the user requests help, does the system clearly explain the possible actions which can be taken, in the context of what the user is doing?	X				
5. Does the help allow the user to browse through information about other parts of the system?				N	
6. Does the user guide an in-depth, comprehensive description?		X			
7. Is it easy to find a particular section in the user guide?		X			
8. Is the organization of all forms of support related to the tasks which the user can carry out?	I don't understand the sentence				
9. Do user support facilities adequately explain user and system errors and how these should be corrected?		X			
10. Are all forms of user support maintained updated?		X			

11. Are there any comments (good or bad) you wish to add regarding the above issues?

12. Overall, how would you rate the system in terms of user guidance and support?

( ) Very satisfatory
(X) Moderately satisfatory
( ) Neutral
( ) Moderately unsatisfactory
( ) Very unsatisfatory

J. SYSTEM USABILITY PROBLEMS

When using the system, did you experience problems with any of the following?

Key: (N)o problems, (Mi)nor problems, (Ma)jor problems

	N	Mi	Ma	Comments
1. Working out how to use the system	X			
2. Lack of guidance on how to use the system	X			
3. Poor system documentation	X			
4. Understanding how to carry out the tasks		X		
5. Knowing what to do next		X		
6. Understanding how the information on the screen relates to what you are doing	X			
7. Finding the information you want		X		
8. Information which is difficult to read clearly	X			
9. An inflexible, rigid structure		X		
10. An inflexible help (guidance) facility				
11. Losing track of where you are in the system or of what you are doing or have done		X		
12. Having to remember too much information while carrying out a task	X			
13. System response times that are too quick for you to understand what is going on	X			too long
14. Information which does not stay on the screen long enough for you to read	X			
15. System response times that are too slow			X	
16. Unexpected actions by the system		x		
17. An input device that is difficult or awckward to use	x			
18. Knowing where or how to put the information		x		
19. Having to spend too much time inputting information		x		
20. Having to be very careful in order to avoid errors		x		
21. Working out how to correct errors		x		
22. Having to spend to much time correcting errors	x			
23. Having to carry out the same type of activity in different ways		x		

K. GENERAL QUESTIONS ON SYSTEM USABILITY

1. What are the best aspects of the system for the user?

security,
fits user requirements
shared tool

2. What are the worst aspects of the system for the user?
text mode
complex
susceptible
sometimes slow
needs a great effort to keep data up-to-date.

3. Are there any parts of the system which you found confusing or difficult to fully understand?

the merge facility,
the cabling part,even for the designer, who does not take part of the sect !)

4. Were there any aspects of the system which you found particularly irritating although they did not cause major problems?

Having to pre-declare what you are going to do :
ENTER QUERY, INSERT_RECORD

5. What were the most common mistakes you made when using the system?
Forgot to press enter query
or trying to insert in query mode -> loose your work

6. What changes would you make to the system to improve it from the user's point of view?

Use my mouse, multi-window

7. Is there anything else about the system you would like to add?