Cours No 01
Introduction aux bases de données relationnelles

Version 1.0

Informations importantes à donner sur le document. Cela peut-être un résumé du contenu, des infos sur les conseils de lectures ou tout autre information jugée utile

LYCEE STENDHAL MILAN
VIA LAVENO, 12
20151 MILANO
ITALIA

SOMMAIRE

NOTION DE BASE DE DONNÉES **4**

SYSTÈME DE GESTION DE BASE DE DONNÉES (SGBD) **4**

LA DÉFINITION DES DONNÉES 5
LA MANIPULATION DES DONNÉES 5
L'INTÉGRITÉ DES DONNÉES 5
LES ACCÈS CONCURRENTS 5
LA CONFIDENTIALITÉ 6
LA SÉCURITÉ DE FONCTIONNEMENT 6

MODÈLES DE REPRÉSENTATION DE DONNÉES ET TYPOLOGIE DES SGBD **6**

LE MODÈLE HIÉRARCHIQUE ET LES SGBD HIÉRARCHIQUES 6
LE MODÈLE RÉSEAU ET LES SGBD RÉSEAUX 6
LE MODÈLE RELATIONNEL ET LES SGBD RELATIONNELS 6
LES MODÈLES ET LES SYSTÈMES ORIENTÉS OBJETS 7

MÉTIERS DU DOMAINE **7**

UTILISATEURS DE BASES DE DONNÉES 7
CONCEPTEURS ET DÉVELOPPEURS 7
ADMINISTRATEUR DES BASES ET DES SYSTÈMES 7
RÉALISATEUR DE LOGICIELS DE GESTION ET DÉVELOPPEMENT DE BASE DE DONNÉES 7

UN PETIT HISTORIQUE **7**

LE MODÈLE RELATIONNEL DE DONNÉES **8**

INTRODUCTION 8
INTRODUCTION AU MODÈLE RELATIONNEL 8
VilleDépart *9*
EXERCICE 11
LES OPÉRATIONS PROPRES À L'ALGÈBRE RELATIONNELLE 12
Opération PROJECTION *12*
Opération SELECTION *12*
Opération DIVISION *13*
Opération JOINTURE (équijointure) *13*
LES OPÉRATIONS ENSEMBLISTES 14
Opération UNION *14*
Opération INTERSECTION *15*
Opération DIFFERENCE *15*
Opération PRODUIT CARTESIEN *16*
EXERCICE 16

LES LANGAGES DES SGBD RELATIONNELS **17**

INTRODUCTION 17
INTERROGATION EN SQL 17
Expression des projections *17*
Exercice : *17*
Expression des sélections *17*
Composition de la sélection et de la projection *18*
Expression du produit cartésien *18*
Expression de la jointure :Première formulation *18*
Expression de la jointure : Deuxième formulation *18*
Expression de l'union, de l'intersection et de la différence *18*
Nommage des objets et alias de relations *19*
Notion de valeur inconnue (ou NULL) *19*

Extension de la jointure *20*
Extension de l'union *20*
Ordonner les résultats en SQL *21*
Les fonctions d'agrégation dans SQL *21*
La clause GROUP BY ... HAVING *22*
Le mot clé EXIST *22*
Exercices *22*
INSTRUCTION SQL DE MISE À JOUR 23
INSERT : ajout de n-uplets *23*
UPDATE : Modification de n-uplets *24*
DELETE :Suppression de n-uplets *24*
MANIPULATION DE SCHÉMA DE RELATION 24
Instruction CREATE TABLE et DROP TABLE *24*
Vues et relations temporaires *26*
Index et clusters *26*

Notion de base de données

Les systèmes classique de gestion de fichiers se révèlent très rapidement inefficaces lorsqu'il s'agit de manipuler des quantités importantes de données surtout lorsqu'elles comportent une liaison entre elles :

- Pour des raisons de stockage
- De redondance de données
- D'intégrité de données

Apparut au début des années 60, le concept de base de données voulait palier à la fois aux insuffisances des systèmes de gestion de fichier et aux inconvénients liés au mode de développement des applications.

Définition :

Un base de données (BD) est une collection de données opérationnelles, enregistrées (sur un support adressable) et utilisées par des systèmes d'application(les programmes) d'une organisation (humaine) particulière. Cette collection de données :

- Est structurée indépendamment d'une application particulière,
- Est cohérente
- Est de redondance minimale
- Est accessible simultanément par différents utilisateurs

Une base de donnée doit donc être

- Structurée
- Mise en commun
- Non redondante
- Disponible

C'est le concepteur de la base qui veillera au 2 premiers points et le système de gestion de base de données qui veillera à l'intégrité et à la disponibilité des données.

Système de gestion de base de données (SGBD)

Un SGBD est une ensemble d'outils logiciels permettant la création et l'utilisation de bases de données.
Il est chargé de :

- Définir les données
- Permettre leur manipulation
- Garantir l'intégrité de ces dernières
- Gérer les accès concurrents
- Garantir la confidentialité des données
- Gérer la sécurité de fonctionnement

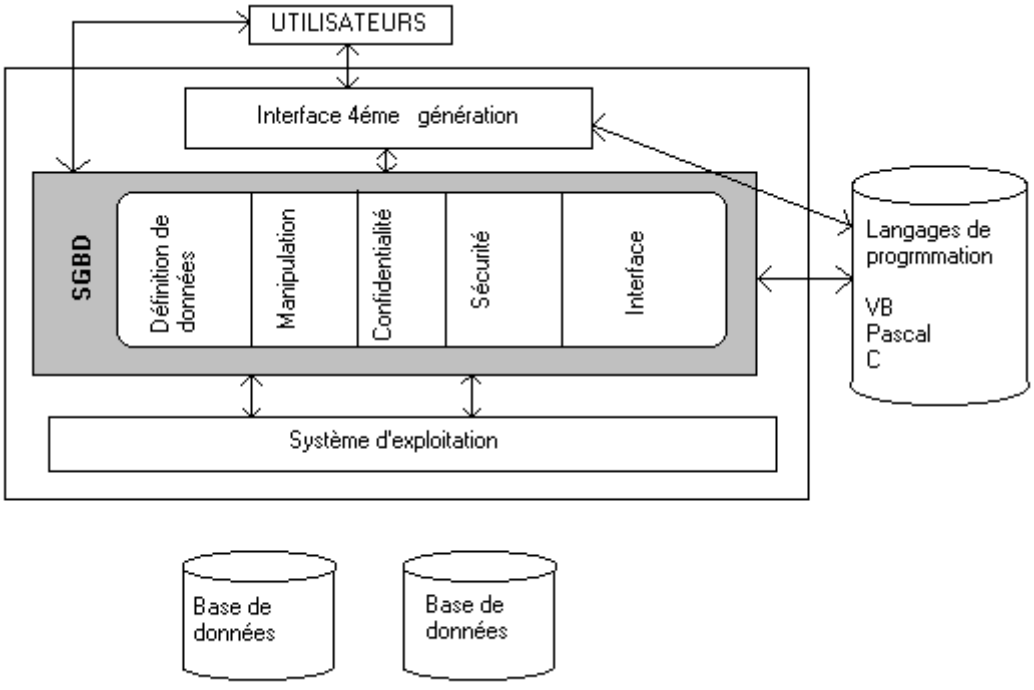


Figure 1:architecture fonctionnelle d'un SGBD

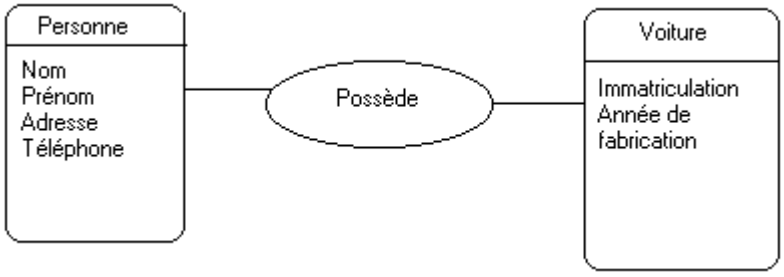
La définition des données

Le SGBD doit offrir à l'utilisateur pour décrire

- Des objets : personnes, voitures, etc..
- Leurs attributs :nom, couleur, marque, etc.,
- Leur liens : une personne possède une voiture, une voiture est d'une couleur, ect..
- Ainsi que d'éventuelles contraintes pouvant concerner ces objets, leur attributs ou leurs liens.

Ces moyens constituent généralement ce que l'on appelle de Langage de Description de Données (LDD).
Le schéma d'une base est la description dans le LDD des objets de la base, de leurs liens, et des contraintes associées.
Cette description est unique et commune aux diverses applications utilisatrices de la base.

Exemple :



Exercice :
Compléter le schéma précédent avec les objets couleur, constructeur, assureur et les relations associées.

La manipulation des données

La manipulation de donnée concerne les outils et les mécanismes qui permettent de faire communiquer une BD et ses clients (utilisateurs ou programmes).
Les SGBD offrent, souvent sous diverses formes, des capacités de recherche, de création, de modification et de suppression d'informations.
Une de ces formes est le langage de manipulation de données (LMD) qui permet de manipuler les données de la base de manière interactive.
L'action à effectuer sur la base est exprimée comme une phrase de ce langage : la requête qui est évaluée et exécutée par le SGBD.
Une autre forme est constituée par les interfaces que peut offrir le SGBD avec des langages de programmation classiques tels que C, C++, Visual Basic, Java ou Cobol.
Ces interfaces permettent l'accès et la manipulation des données d'une base de manière ergonomique et intuitive.

Ces divers moyens de manipulation des données d'une base cache totalement tous les aspect relatifs à l'organisation physique des données sur le support de stockage.

L'intégrité des données

Le concept d'intégrité des données est relatif à la qualité de l'information enregistrée.
Pour être fiable, celle-ci doit parfois vérifier certaines propriétés, comme l'appartenance à une liste de valeurs permises pour un attribut.
Ces propriétés sont appelées **contraintes d'intégrité.**
Elles sont spécifiées lors de la définition du schéma de la base, le SGBD se chargeant de les préserver pendant toute la vie de la base.

Les accès concurrents

Les données d'une base pouvant être accédées à tout moment par plusieurs utilisateurs, le SGBD doit offrir des mécanismes de gestion de conflit d'accès.
Ceux-ci sont similaires à ceux rencontrés dans les systèmes d'exploitation :

- Autorisation des accès multiples en consultation
- Verrouillages en modification,
- ….

La confidentialité

La mise en commun des données sous la forme de base de données accroît le besoin en confidentialité.
L'utilisation des sous-schémas fait partie des moyens qui permettent de l'assurer.
Mais de façon plus générale, elle est gérée par le biais de mots de passe et de niveaux d'accès.

La sécurité de fonctionnement

Le SGBD doit offrir des mécanismes permettant de remettre rapidement la base de données en état opérationnel en cas d'incident matériel ou logiciel qui en aurait altéré la qualité.
Ces mécanismes sont basés sur la journalisation des opérations réalisées sur la base et leur ré-éxecution automatique en cas de besoins.

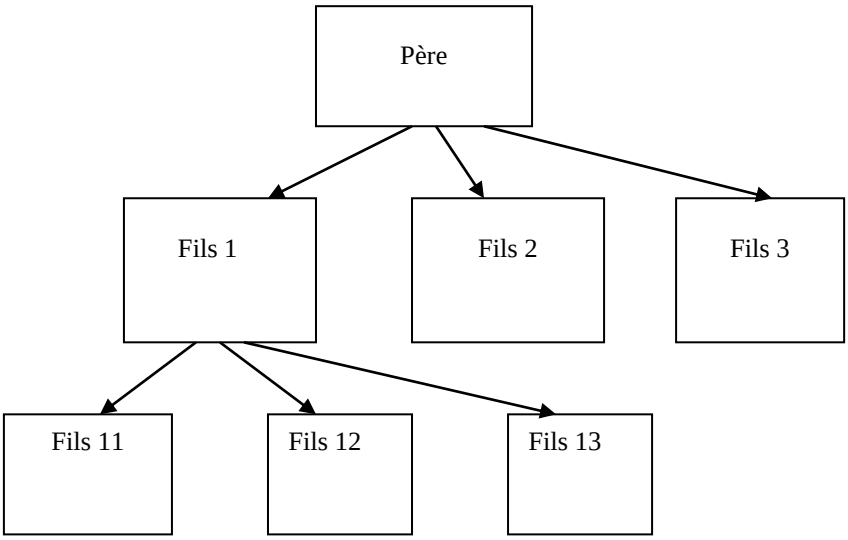
Modèles de représentation de données et typologie des SGBD

La typologie des SGBD dépend fortement des types de structures de données que les SGBD supportent.
Il existe 3 modèles de représentation de données :

- Le modèle hiérarchique
- Le modèle réseaux
- Le modèle relationnel
- Les modèles orientés objets

Le modèle hiérarchique et les SGBD hiérarchiques

Ce type de modèle permet de représenter des relations de type « père-fils » entre les objets :



Les SGBD supportant ce type de structure gère les liens entre un père et un fils.
Ils offrent des primitives pour manipuler et naviguer dans de telles structures (recherche d'un père, parcours des fils, etc..).
Ce modèle est de moins en moins utilisé à cause des limites qu'il présente en terme de relation.

Le modèle réseau et les SGBD réseaux

Ce modèle permet également de représenter des classes objets et des liens de type père-fils à la différence que l'on autorise une classe fille à posséder plusieurs classes mères.
Les langages de ces SGBD sont aussi navigationnels.

Le modèle relationnel et les SGBD relationnels

Ce modèle permet de voir une base comme un ensemble de tables.
Les langages relationnels de manipulation de données se caractérisent par leur nature déclaratives ; alors que chaque SGBD hiérarchique ou réseau possède sont propre langage de manipulation, les SGBD relationnels offrent un langage de manipulation de données standard, le SQL (Structured Query Language) qui fondé sur l'algèbre relationnelle.

Les modèles et les systèmes orientés objets

Le domaine des bases de données n'est pas resté insensible à l'explosion du phénomène objet !
Nombre travaux ont visé à intégrer des concepts issus des langages orientés objets (notion de classe, et de sous-classes, opération et méthodes associées, héritage, …) et des concepts issus des bases de données.
A la différence des SGBD relationnels, à l'heure actuelle, il n'existe pas de langage standard pour les SGBD orientés objets. OQL (Object Query Language) est en cours de définition.

Métiers du domaine

Les intervenants du domaine des bases de données peuvent être classifiés fonction de leur type d'activité et du degré de spécialisation requis pour assurer cette activité.

Utilisateurs de bases de données

On peut distinguer trois types d'utilisateurs de bases de données :

- Les utilisateurs occasionnels, qui ont généralement une technicité moyenne et qui accèdent à la base en utilisant le langage de manipulation de données
- Les utilisateurs « naïfs », qui accèdent et modifient la base par le biais de transactions préprogrammées (utilisation « presse boutons »).
- Les utilisateurs plus « spécialisés » qui ont généralement une technicité élevée leur permettant de mettre en œuvre les différents outils SGBD.

Concepteurs et développeurs

a) Le concepteur de base de données : il identifie et structure les types de données de la base ainsi que les divers traitements que ces données doivent subir. Ses compétences doivent s'étendre au-delà du domaine strict des bases de données et inclure une ou des méthodes de conception de logiciels.
b) Les développeurs d'applications : ils ont pour rôle de déterminer les besoins des utilisateurs, de spécifier et d'implanter les transactions et les programmes nécessaires à leur satisfaction. Les compétences requises incluent la construction de programmes (algorithmique et programmation) et la maîtrise des langages de manipulation de données.

Administrateur des bases et des systèmes

L'administrateur (personne ou équipe de personne) est responsable d'une ou de plusieurs bases de données. Il a notamment pour rôle la délivrance des autorisations d'accès à la base et à la coordination des activités. Il est également responsable des problèmes de performances et de sécurité de fonctionnement. De ce fait, sa compétence va au-delà de celles des deux compétences vues précédemment dans la mesure où il doit avoir connaissance de la façon dont est réalisé le SGBD qu'il doit administrer.

Réalisateur de logiciels de gestion et développement de base de données

a) Les développeurs d'outils : par outil, on entends des logiciels facilitant la conception, l'implémentation et l'utilisation de base de données ou de SGBD existants. Ces outils ne font pas, à proprement parler partie du SGBD.
b) Les concepteurs et implémenteurs de SGBD : une très grande technicité est requise pour cette classe d'acteur. Leur connaissance doit comprendre outre les techniques d'implémentation des SGBD, des compétences en compilation et langage, en systèmes d'exploitation, en réseaux, etc.

Un petit historique

- **1961 :** Apparition du système IDS (Integrated Data Storage Electric). La terminologie utilisée (record types et set types) est à la base du modèle réseau développé par la CODASYL DBTG (Conference On Data Systems and Languages Data Base Task Group)
- **1965-1970 :** Développement de systèmes de gestion de fichiers généralisés. Développement par IBM, du modèle hiérarchique et du système IMS (Information Management System).
- **1970 :** Apparition du modèle relationnel de données
- **1971 :** Publication du rapport CODASYL DBTG
- **1972 :** Première conférence internationale organisée pr ACM SIGMOD (Association of Computing Machinery, Special Interest Group on Management Of Data
- **1975 :** Première conférence internationale VLDB (Very Large Language). Publication du rappor9t ANSI-SPARC. Apparition du modèle individuel, dans le cadre de travaux ayant conduit à la méthode MERISE
- **1976 :** Publication du modèle Entit-Association
- **1975-1980** :Développement de systèmes relationnels expérimentaux :INGRE (Berkeley, University of California) et SYSTEM-R(IBM).
- **1980-… :** Apparition et commercialisation de nombreux SGBD relationnels qui ont supplanté les SGBD hiérarchiques et réseaux. Ces années ont également vu la disponibilité de SGBD relationnels sur micro-ordinateurs (Access) et la rlisation de systèmes de quatrième génération avec des outils et des interfaces multiples.

Le modèle relationnel de données

Introduction

Plusieurs étapes sont nécessaires à la mise en place d'une base de données, dès lors que l'on a précisément défini ses besoins (ce qui n'est déjà pas chose facile !) :

- **la création de la structure de la base** sous forme de tables (tableaux de données) reliées entre elles par des données clés,
- **la conception des requêtes** qui permettront d'extraire ou de mettre à jour les informations qu'elle contient,
- **la conception de l'interface** homme-machine (écrans et états) qui rendra plus conviviale la saisie et la restitution des informations. Le degré de difficulté dans la conception de l'interface varie beaucoup selon le logiciel utilisé qui peut d'ailleurs être différent du SGBDR.

La conception de la structure de la base de données, si elle est un peu complexe à appréhender, peut nécessiter, en amont, l'utilisation d'**outils de modélisation conceptuels** de type **schéma entités-associations** (emprunté à la méthode MERISE). Mais, même dans les cas les plus simples il faut obligatoirement connaître les concepts du **Modèle Relationnel**, sans quoi un utilisateur non averti pourra toujours arriver à créer une structure inadaptée et sera vite bloqué dans la conception des requêtes.
Il s'agit ici, d'étudier les principaux opérateurs de l'algèbre relationnelle servant de base à l'élaboration des requêtes.

- Elle est à l'origine du langage **SQL** (Structured Query Language) d'IBM, langage d'interrogation et de manipulation de tous les SGBDR actuels (Oracle, Sybase, Informix, Ingres, DB2, MS Access et tous les autres).
- Elle est également mise en œuvre de manière plus conviviale à travers le langage **QBE** (Query By Example) que l'on retrouve seulement sur certains SGBDR (Access par exemple) et qui n'est pas présenté ici.

Une bonne maîtrise de l'algèbre relationnelle permet de concevoir n'importe quelle requête aussi complexe soit elle avant de la mettre en œuvre à l'aide du langage QBE ou SQL.
Parmi les opérations de l'algèbre relationnelle, on dispose

- **Des opérations classiques sur les ensembles** (union, intersection, différence, produit cartésien)
- **D**es **opérations propres** (projection, sélection, jointure, division).
- **Des opérations de calcul, de regroupement**, de comptage et de tri, non définies à l'origine par Codd mais très utiles.

Tous les opérateurs sont présentés à l'aide d'exemples clairs. Pris séparément, ils sont faciles à appréhender. La rédaction de requêtes (combinaison d'opérateurs) est illustrée par des exercices concrets.

Le langage SQL n'est abordé que dans le cadre des opérations évoquées ci-dessus.

Introduction au Modèle Relationnel

L'exemple suivant, relatif à la gestion simplifiée des étapes du Tour de France 97, va nous servir à introduire le vocabulaire lié au modèle relationnel.

CodeEquipe	NomEquipe	DirecteurSportif
BAN	BANESTO	Eusebio UNZUE
COF	COFIDIS	Cyrille GUIMARD
CSO	CASINO	Vincent LAVENU
FDJ	LA FRANCAISE DES JEUX	Marc MADIOT
FES	FESTINA	Bruno ROUSSEL
GAN	GAN	Roger LEGEAY
ONC	O.N.C.E.	Manolo SAIZ
TEL	TELEKOM	Walter GODEFROOT
...	...	...

NuméroCoureur	NomCoureur	CodeEquipe	CodePays
8	ULLRICH Jan	TEL	ALL
31	JALABERT Laurent	ONC	FRA
61	ROMINGER Tony	COF	SUI
91	BOARDMAN Chris	GAN	G-B
114	CIPOLLINI Mario	SAE	ITA
151	OLANO Abraham	BAN	ESP
...	...	...	...

NuméroEtape	DateEtape	VilleDépart	VilleArrivée	NbKm
1	06-jul-97	ROUEN	FORGES-LES-EAUX	192
2	07-jul-97	ST-VALERY-EN-CAUX	VIRE	262
3	08-jul-97	VIRE	PLUMELEC	224
...	...	...	...	...

NuméroCoureur	NuméroEtape	TempsRéalisé
8	3	04:54:33
8	1	04:48:21
8	2	06:27:47
31	3	04:54:33
31	1	04:48:37
31	2	06:27:47
61	1	04:48:24
61	2	06:27:47
91	3	04:54:33
91	1	04:48:19
91	2	06:27:47
114	3	04:54:44
114	1	04:48:09
114	2	06:27:47
151	3	04:54:33
151	1	04:48:29
151	2	06:27:47
...	...	...

CodePays	NomPays
ALL	ALLEMAGNE
AUT	AUTRICHE
BEL	BELGIQUE
DAN	DANEMARK
ESP	ESPAGNE
FRA	FRANCE
G-B	GRANDE BRETAGNE
ITA	ITALIE
P-B	PAYS-BAS
RUS	RUSSIE
SUI	SUISSE
…	…

Comme nous pouvons le constater, le modèle relationnel est un modèle d'organisation des données sous forme de Tables (Tableaux de valeurs) ou chaque **Table** représente une **Relation**, au sens mathématique d'**Ensemble**.
C'est ainsi que dans l'exemple présenté, figurent

- l'ensemble des Equipes,

- des Coureurs,
- des Etapes,
- des Temps réalisés par les coureurs à chacune des étapes,
- et enfin l'ensemble des pays.

Les colonnes des tables s'appellent des **attributs** et les lignes des **n-uplets** (où n est le degré de la relation, c'est à dire le nombre d'attributs de la relation).
Un attribut ne prend qu'une seule valeur pour chaque n-uplet.
L'ordre des lignes et des colonnes n'a pas d'importance.

Chaque table doit avoir une **clé primaire** constituée par un ensemble minimum d'attributs permettant de distinguer chaque n-uplet de la Relation par rapport à tous les autres.
Chaque ensemble de valeurs formant la clé primaire d'un n-uplet est donc **unique** au sein d'une table.

C'est ainsi que dans la table COUREURS, chaque coureur a un NuméroCoureur différent.
Dans certains cas, plusieurs clés primaires sont possibles pour une seule table. On parle alors de **clés candidates**. Il faut alors en choisir une comme clé primaire.

Les liens sémantiques (ou règles de gestion sur les données) existants entre les ensembles sont réalisés par l'intermédiaire de **clés étrangères** faisant elles-mêmes référence à des clés primaires d'autres tables.
Exemple :
Dans la table COUREURS, la clé étrangère CodeEquipe (faisant référence à la clé primaire de même nom dans la table EQUIPES) traduit les deux règles de gestion suivantes :
Un COUREUR appartient à une EQUIPE
Une EQUIPE est composée de plusieurs COUREURS
Il existe deux grands types de liens :

- **Un - Plusieurs** (comme le précédent)
- **Plusieurs - Plusieurs**.

La réalisation de ce dernier type de liens, un peu plus complexe, passe par l'utilisation d'une table intermédiaire dont la clé primaire est formée des clés étrangères des tables qu'elle relie.

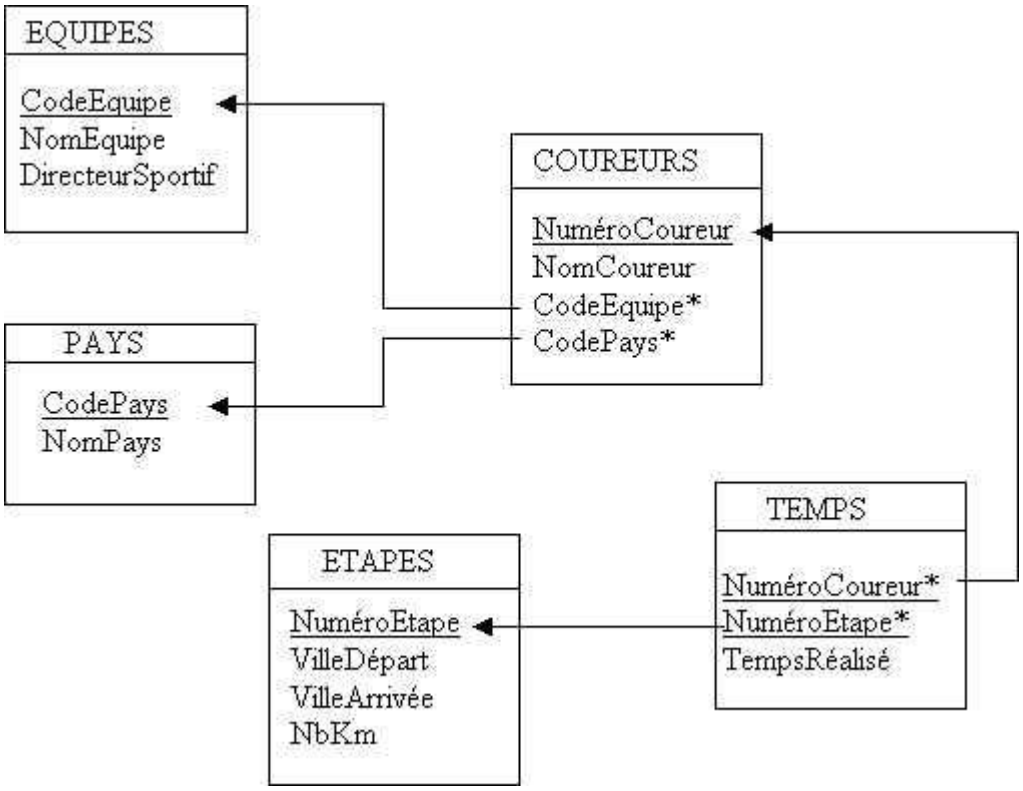
Exemple :
La table des TEMPS réalisés à chaque étape par chacun des coureurs exprime les deux règles de gestion suivantes :

Un COUREUR participe à plusieurs ETAPES
Une ETAPE fait participer plusieurs COUREURS

Le modèle relationnel est le plus souvent décrit sous la forme suivante, les clés primaires étant soulignées et les clés étrangères marquées par un signe distinctif (ici *).

EQUIPES (CodeEquipe, NomEquipe, DirecteurSportif)
COUREURS (NuméroCoureur, NomCoureur, CodeEquipe*, CodePays*)
ETAPES (NuméroEtape, VilleDépart, VilleArrivée, NbKm)
TEMPS (NuméroCoureur*, NuméroEtape*, TempsRéalisé)
PAYS (CodePays, NomPays)

On peut aussi le représenter sous forme graphique, de manière à mieux visualiser et interpréter les liens :



Un COUREUR appartient à une EQUIPE
Une EQUIPE est composée de plusieurs COUREURS
Un COUREUR est originaire d'un PAYS
Un PAYS est représenté par plusieurs COUREURS
Un COUREUR participe à plusieurs ETAPES
Une ETAPE fait participer plusieurs COUREURS

Dans le cadre d'un projet d'informatisation, la conception d'une base de données relationnelle passe d'abord par l'identification des objets de gestion (Coureurs, Etapes, …) et des règles de gestion du domaine modélisé (interviews des utilisateurs, étude des documents manipulés, des fichiers existants, …). Une fois énoncées et validées, ces règles nous conduisent automatiquement à la structure du modèle relationnel correspondant.

Exercice

On a la situation suivante en déduire le schéma relationnel optimal :

- Par la forme relationnelle
- Par un graphique

Les mots en italique désignent des tables ou des attributs de tables.

Description partielle de l'activité et de l'organisation d'un centre de recherche

- Les membres d'un centre de recherche sont regroupés en équipes de recherche :
- *Personne.No_equ* désigne le numéro de l'équipe d'appartenance d'une personne.
- Chaque équipe est dirigée par une personne : *equ.resp* désigne le responsable de l'équipe.
- Les équipes conduisent des projets (relation *développe*) sur un domaine de recherche : *projet.domaine* .
- Un projet est sous la direction d'une personne : *projet.resp* et il peut faire l'objet d'un contrat.
- Plusieurs organismes peuvent contribuer au financement d'un *contrat* : *financement.apport.*
- Des personnes peuvent participer à plusieurs projets.
- Les résultats de la recherche peuvent donner lieu à des *publications* dans des revues, journaux et autres congrès.
- Les publications peuvent avoir des *auteurs*.

Aide : la relation Personne se décrira par exemple de la manière suivante :
Personne(No_Pers*, Nom, Pom, Statut, Grade, No_Equ*)

Il va falloir décrire les relations Equipe, Domaine, Projet, Contrat, Financement, Développe, Participe, Publication, et Auteurs.

Les opérations propres à l'Algèbre Relationnelle

Opération PROJECTION

Formalisme :
R = PROJECTION (R1, liste des attributs) ou Relation * Liste des attributs → Relation

Exemples :
CHAMPIGNONS

Espèce	**Catégorie**	**Conditionnement**
Rosé des prés	Conserve	Bocal
Rosé des prés	Sec	Verrine
Coulemelle	Frais	Boîte
Rosé des prés	Sec	Sachet plastique

R1 = PROJECTION (CHAMPIGNONS, Espèce)

Espèce
Rosé des prés
Coulemelle

R2 = PROJECTION (CHAMPIGNONS, Espèce, Catégorie)

Espèce	**Catégorie**
Rosés des prés	Conserve
Rosé des prés	Sec
Coulemelle	Frais

- Cet opérateur ne porte que sur 1 relation.
- Il permet de ne retenir que certains attributs spécifiés d'une relation.
- On obtient tous les n-uplets de la relation ***à l'exception des doublons***.

Opération SELECTION

Formalisme :
R = SELECTION (R1, condition) ou Relation * Condition → Relation

On peut veut voir la ou les conditions comme une expression logique, La sélection parfois aussi appelée restriction, permet de rechercher dans une relation (ou table) les élément vérifiant une ou plusieurs condition.

Exemple :
CHAMPIGNONS

Espèce	Catégorie	Conditionnement
Rosé des prés	Conserve	Bocal
Rosé des prés	Sec	Verrine
Coulemelle	Frais	Boîte
Rosé des prés	Sec	Sachet plastique

R3 = SELECTION (CHAMPIGNONS, Catégorie = "Sec")

Espèce	Catégorie	Conditionnement
Rosé des prés	Sec	Verrine
Rosé des prés	Sec	Sachet plastique

- Cet opérateur porte sur 1 relation.

- Il permet de ne retenir que les n-uplets répondant à une condition exprimée à l'aide des opérateurs arithmétiques (=, >, <, >=, <=, <>) ou logiques de base (ET, OU, NON).
- Tous les attributs de la relation sont conservés.

Opération DIVISION

Formalisme :
R = DIVISION (R1, R2) ou Relation * Relation → Relation

Exemple :
PARTICIPER EPREUVE DIVISION (PARTICIPER, EPREUVE)

Athlète	Epreuve		Epreuve		Athlète
Dupont	200 m		200 m		Dupont
Durand	400 m		400 m		
Dupont	400 m		110 m H		
Martin	110 m H				
Dupont	110 m H				
Martin	200 m				

"L'athlète Dupont participe à toutes les épreuves"

- Cet opérateur porte sur 2 relations qui doivent avoir au moins un attribut défini dans le même domaine.
- Tous les attributs du diviseur (ici EPREUVE) doivent être des attributs du dividende (ici PARTICIPER).
- La relation dividende doit avoir au moins une colonne de plus que la relation diviseur.
- La relation résultat, le quotient, possède les attributs non communs aux deux relations initiales et est formée de tous les n-uplets qui, concaténés à chacun des n-uplets du diviseur (ici EPREUVE) donne toujours un n-uplet du dividende (ici PARTICIPER).

Opération JOINTURE (équijointure)

Formalisme :

- R = JOINTURE (R1, R2, condition d'égalité entre attributs) ou
- Relation * Relation * Condition → Relation

Exemple :

PRODUIT DETAIL_COMMANDE

CodePrd	Libellé	Prix unitaire		N°cde	CodePrd	quantité
590A	HD 1,6 Go	1615		97001	590A	2
588J	Scanner HP	1700		97002	515J	1
515J	LBP 660	1820		97003	515J	3

R = JOINTURE (PRODUIT, DETAIL_COMMANDE,Produit.CodePrd=Détail_Commande.CodePrd)

CodePrd	Libellé	Prix unitaire	N°cde	quantité
590A	HD 1,6 Go	1615	97001	2
515J	LBP 660	1820	97002	1
515J	LBP 660	1820	97003	3

- Cet opérateur porte sur 2 relations qui doivent avoir au moins un attribut défini dans le même domaine (ensemble des valeurs permises pour un attribut).
- La condition de jointure peut porter sur l'égalité d'un ou de plusieurs attributs définis dans le même domaine (mais n'ayant pas forcément le même nom).
- Les n-uplets de la relation résultat sont formés par la concaténation des n-uplets des relations d'origine qui vérifient la condition de jointure.

Remarque : Des jointures plus complexes que l'équijointure peuvent être réalisées en généralisant l'usage de la condition de jointure à d'autres critères de comparaison que l'égalité (<,>, <=,>=, <>).

Les opérations ensemblistes

Opération UNION

Formalisme :
R = UNION (R1, R2) ou Relation * Relation → Relation
Exemple :
E1 : Enseignants élus au CA E2 : Enseignants représentants syndicaux

n° enseignant	nom_enseignant
1	DUPONT
3	DURAND
4	MARTIN
5	BERTRAND

n°enseignant	nom_enseignant
1	DUPONT
4	MARTIN
6	MICHEL

On désire obtenir l'ensemble des enseignants élus au CA ou représentants syndicaux.
R1 = UNION (E1, E2)

n°enseignant	nom_enseignant
1	DUPONT
3	DURAND
4	MARTIN
5	BERTRAND
6	MICHEL

- Cet opérateur porte sur deux relations qui doivent avoir le même nombre d'attributs définis dans le même domaine (ensemble des valeurs permises pour un attribut). On parle de relations ayant le même schéma.
- La relation résultat possède les attributs des relations d'origine et les n-uplets de chacune, avec élimination des doublons éventuels.

Opération INTERSECTION

Formalisme :
R = INTERSECTION (R1, R2) ou Relation * Relation → Relation

Exemple :
E1 : Enseignants élus au CA E2 : Enseignants représentants syndicaux

n° enseignant	nom_enseignant
1	DUPONT
3	DURAND
4	MARTIN
5	BERTRAND

n°enseignant	nom_enseignant
1	DUPONT
4	MARTIN
6	MICHEL

On désire connaître les enseignants du CA qui sont des représentants syndicaux.
R2 = INTERSECTION (E1, E2)

n°enseignant	nom_enseignant
1	DUPONT
4	MARTIN

- Cet opérateur porte sur deux relations de même schéma.
- La relation résultat possède les attributs des relations d'origine et les n-uplets communs à chacune.

Opération DIFFERENCE

Formalisme :
R = DIFFERENCE (R1, R2) ou Relation * Relation → Relation

Exemple :
E1 : Enseignants élus au CA E2 : Enseignants représentants syndicaux

n° enseignant	nom_enseignant
1	DUPONT
3	DURAND
4	MARTIN
5	BERTRAND

n°enseignant	nom_enseignant
1	DUPONT
4	MARTIN
6	MICHEL

On désire obtenir la liste des enseignants du CA qui ne sont pas des représentants syndicaux.
R3 = DIFFERENCE (E1, E2)

n°enseignant	nom_enseignant
3	DURAND
5	BERTRAND

- Cet opérateur porte sur deux relations de même schéma
- La relation résultat possède les attributs des relations d'origine et les n-uplets de la première relation qui n'appartiennent pas à la deuxième.
- Attention ! DIFFERENCE (R1, R2) ne donne pas le même résultat que DIFFERENCE (R2, R1)

Opération PRODUIT CARTESIEN

Formalisme :
R = PRODUIT (R1, R2) ou Relation * Relation → Relation

Exemple :
Etudiants Epreuves

n°étudiant	nom		libellé épreuve	coefficient
101	DUPONT		Informatique	2
102	MARTIN		Mathématiques	3
			Gestion financière	5

Examen = PRODUIT (Etudiants, Epreuves)

n°étudiant	nom	libellé épreuve	coefficient
101	DUPONT	Informatique	2
101	DUPONT	Mathématiques	3
101	DUPONT	Gestion financière	5
102	MARTIN	Informatique	2
102	MARTIN	Mathématiques	3
102	MARTIN	Gestion financière	5

- Cet opérateur porte sur deux relations.
- La relation résultat possède les attributs de chacune des relations d'origine et ses n-uplets sont formés par la concaténation de chaque n-uplet de la première relation avec l'ensemble des n-uplets de la deuxième.

Exercice

- Exprimer les requêtes ci-dessous en algèbre relationnelle (elles se basent sur le modèle relationnel établit lors de l'exercice précédent)
- Pour chaque requête donner le schéma de la relation résultat

1. Quels sont les intitulés des domaines de recherche du laboratoire ?
2. c
3. Trouver les noms et prénoms des personnes qui ont été auteurs de publications durant les 2 dernières années.
4. Donner les noms et prénoms des personnes qui n'ont pas été auteur de publications durant les 2 années précédentes
5. Donner les noms et prénoms des personnes qui ont le même grade que le responsable d'équipe

Les langages des SGBD relationnels

Introduction

Le développement des langages pour les SGBD relationnels s 'est inspiré des résultats des travaux qui ont conduit à l'implémentation de SYSTEM-R. En effet, celui-ci proposait un langage déclaratif appelé SQUARE puis renommé en SEQUEL puis en SQL (Structured Query Language).
Ce langage a donné lieu à diverses implémentations et est toujours l'objet de travaux de normalisation.
SQL peut être logiquement vu comme étant composé de 3 sous-langages :

- Un langage de manipulation de données
- Un langage de description de données
- Un langage de contrôle de données

Interrogation en SQL

L'interrogation de relation s'exprime à l'aide de la commande SELECT, à ne pas confondre avec l'expression algébrique.
Dans ce chapitre, nous montrerons comment exprimer en SQL les opérateurs algébriques et les formules prédicatives.

Expression des projections

SELECT <liste d'attributs> FROM <nom de la relation>

Exemple :
Pour reprendre l'exemple sur les projections du chapitre précédent, on écrira :

SELECT Espèce FROM CHAMPIGNONS
Ou
SELECT Espèce, Catégorie FROM CHAMPIGNONS

Exercice :

Ecrire en SQL les projections

1. R1=Projection(COURREUR, Nom, Prénom, Poids)
2. R2=Projection(PERSONNE, Nom, Statut, Grade)
3. R3=Projection(PROJET, Responsable, Financement)
4. Donner le nom et la performance des coureurs

Expression des sélections

SELECT * FROM <nom de relation> WHERE <liste de condition>

Décrit toutes les colonnes de la relation. La qualification est une expression conditionnelle qui peut comporter les opérateurs de comparaison =, !=, >, <, >=, <= ou [NOT] LIKE (qui permet de rechercher un « modèle » de valeur). Les expressions conditionnelles peuvent être reliées par les connecteurs AND et OR. On pourra également utiliser des caractères génériques comme « % », auquel peut se substituer toute chaîne de caractères, ou « _ », auquel peut se substituer tout caractère.

Exemple

Sélectionner les champignons dont l'espèce est Coulemelle
SELECT * FROM CHAMPIGNONS WHERE Espèce = 'Coulemelle'

Sélectionner les champignons de catégorie sec et dont l'espèce est Coulemelle
SELECT * FROM CHAMPIGNONS WHERE Catégorie = "Sec" AND Espèce = 'Coulemelle'

Sélectionner les coureurs dont le nom commence par un « L »
SELECT * FROM COUREURS WHERE nom LIKE « L% »

Exercice

Ecrire en SQL les sélections :

1. R1=SELECTION(ETAPE,TempsRéalisé>06 :00 :00)
2. R2=SELECTION(EQUIPE,DirecteurSportif= « Marc MADIOT »)
3. Sélectionner les coureurs appartenant à l 'équipe GAN
4. Sélectionner les personnes appartenant à l'équipe ayant le numéro 1

Composition de la sélection et de la projection

SELECT <liste d'attributs> FROM <nom de la relation> WHERE <liste de condition>

Exemple :

Sélectionner le nom des coureurs dont le nom commence par un « L »
SELECT nom FROM COUREURS WHERE nom LIKE « L% »

Exercice:

1. Quels sont les noms et prénoms des membres de l'équipe ayant le numéro 1
2. Trouver le nom et le prénom du responsable de l'équipe numéro 1

Expression du produit cartésien

Le produit cartésien s'exprime comme une jointure sans qualification.
SELECT * FROM <liste de relations>

Exemple

Exercice

Expression de la jointure :Première formulation

SELECT * FROM <liste de relations> WHERE <expression de jointure>

Exemple

SELECT * FROM EQUIPE,COURREUR WHERE EQUIPE.CodeEquipe = COURREUR.CodeEquipe

Expression de la jointure : Deuxième formulation

Cette formulation utilise une imbrication de SELECT, appelée parfois sous-requête. On notera que l'utilisation (le l'opérateur d'appartenance à un ensemble (IN) requiert que l'élément dont on veut tester l'appartenance soit du rnême type que les éléments de l'ensemble : ceci justifie, dans l'exemple ci-dessous, l'utilisation de la projection de Produit sur prod, dans la sous-requête. SELECT* FROM Stock WHERE prod= IN (SELECT prod- FROM Produit)

Exemple

SELECT * FROM EQUIPE WHERE EQUIPE.CodeEquipe IN (SELECT COURREUR.CodeEquipe FROM COURREUR)

Expression de l'union, de l'intersection et de la différence

(< Clause SELECT>)< Opérateur ensembliste > (< Clause SELECT>)

Certaines implantations de SQL offrent des opérateurs ensemblistes qui permettent d'exprimer l'union (UNION), l'intersection (INTERSECT) et la différence (MINUS) à l'aide de sous-requête. Il faut veiller à ce que les opérandes de gauche et de droite de l'opérateur soient des relations de même schéma ou de schéma compatibles.
Pour cela, il suffit de formuler les projections adéquates dans chaque clause SELECT.

Exemples

SELECT PERSONNE .nom FROM PERSONNE

UNION

SELECT EQUIPE.responsable FROM EQUIPE

SELECT PERSONNE .nom FROM PERSONNE

INTERSECT

SELECT EQUIPE.responsable FROM EQUIPE

Nommage des objets et alias de relations

Des formules ci-dessus utilisent des alias de relation. En effet, la désignation des attributs d'une relation peut se faire sous l'une des formes

1. NomAttribut, lorsqu'il n'y a pas d'ambiguïté,
2. NomDeRelation.NomAttribut,
3. ALiasDeRelation.NomAttribut.

Cette dénotation peut parfois être étendue à gauche par le nom du propriétaire de la relation voire par le nom de la base contenant la relation.

Exemple

Il y a conflit entre le no_équipe dans la table personne et équipe on est donc obligé de faire préceder l'attribut no_équipe du nom de la table qui le concerne.

De façon similaire, il est possible de renommer les attributs de la relation résultat ; en effet implicitement, dans la table résultat, les noms des attributs sont ceux indiqués dans la liste de projection .
Les alias des attributs servent à les renommer mais ils ne sont pas utilisables dans le corps de la requête.

Exemple

SELECT nom AS « Nom de la personne » FROM PERSONNE

Notion de valeur inconnue (ou NULL)

Des colonnes d'une relation peuvent ne pas comporter de valeur pour certaines lignes de la relation.
L'interprétation de cette absence de valeur peut être

- que des valeurs n'ont pas été fournies lors de l'insertion des lignes concernées
- que les colonnes en question n'ont pas sens pour les lignes concernées.

Cette absence de valeur est appelée **valeur inconnue** et elle est dénotée **NULL**.

Il est important de remarquer qu'une valeur NULL n'est ni zéro ni une chaîne vide, par exemple.

La seule opération permise sur ce type de valeur est une opération de test (IS NULL ou IS NOT NULL) permettant de savoir si un attribut a une valeur inconnue ou pas. Par ailleurs, compte tenu de cette notion de valeur inconnue, les tables de vérité traditionnelles du ET et du OU logiques sont étendues comme indiqué dans les tableaux suivants :

ET	Vrai	Faux	Null
Vrai	Vrai	Faux	Null
Faux	Faux	Faux	Faux
Null	Null	Faux	Null

OU	Vrai	Faux	Null
Vrai	Vrai	Vrai	Vrai
Faux	Vrai	Faux	Null
Null	Vrai	Null	Null

Extension de la jointure

La jointure externe (OUTER JOIN) étend le résultat de la jointure ordinaire par les n-uplets d'une relation auxquels il ne correspond pas de n-uplets dans l'autre relation.

Par exemple, s'il existe des produits dans la relation Produit qui ne figurent pas dans la relation Stock et que l'on désire obtenir les libellés, les prix unitaires et les quantités en stock de tous les produits, une équi-jointure n'inclurait pas ces produits dans la relation résultat. Par contre, la jointure externe le ferait en valorisant à NULL l'attribut quantité en stock des produits qui n'apparaissent pas dans la relation Stock :

PRODUIT

n°produit	nom
101	Feuilles
102	Crayons
103	Feutres
104	Transparents

STOCK

n°produit	quantité
101	2
102	3
103	5

SELECT nom, quantité FROM PRODUIT , STOCK WHERE PRODUITn°produit *= STOCK n°produit
Ou
SELECT nom, quantité FROM PRODUIT , STOCK WHERE PRODUITn°produit (+)= STOCK n°produit

Donne le résultat :

nom	quantité
Feuilles	2
Crayons	3
Feutres	5
Transparents	NULL

La jointure externe est dénotée sous Sybase :

- Attribut1 *= Attrïbut2 donc PRODUIT.n°produit *= STOCK.n°produit
- Attribut1 =* Attribut2 donc PRODUIT.n°produit =* STOCK.n°produit

La jointure externe est dénotée sous Oracle :

- Attribut1 = Attribut2 (+) donc PRODUIT.n°produit = STOCK.n°produit (+)
- Attribut1 (+) = Attribut2 donc PRODUIT.n°produit (+)= STOCK.n°produit

Cette dernière notation étant plus proche de la notation préconisée par SQL-92.
La relation à laquelle appartient l'attribut du côté duquel se trouve le symbole de jointure externe (`(+)' ou `*') est parfois appelée table externe, la seconde étant appelée table interne.
Les valeurs nulles sont engendrées pour les attributs de la table externe.

Extension de l'union

L'union de relations peut parfois être étendue pour admettre comme opérandes des relations n'ayant pas des schémas identiques ou compatibles.
Par exemple, l'union externe (OUTER UNION) des relations

- Employe(pers#, nom, salaire) et
- Etudiant(pers#, nom, niveau)

a comme résultat une relation de schéma (pers#, nom, salaire, niveau) où l'attribut salaire a une valeur nulle pour les instances de la relation Etudiant dont <pers#, nom> ne figurent pas dans la relation Employe et l'attribut niveau a une valeur nulle pour les instances de la relation Employe dont <pers#, nom> ne figurent pas dans la relation Etudiant :

(SELECT * FROM ETIDIANT) OUTER UNION (SELECT * FROM EMPLOYE)

donnera le résultat suivant

pers	nom	salaire	niveau
01	Etud1	Null	Bac+3
02	Emplois jeune	SMIC	BAC+4
03	Salarié1	120 KF	Null
04	Salarié2	130 KF	Null

Ordonner les résultats en SQL

La clause ORDER BY permet d'ordonner la relation résultat sur un ou plusieurs attributs, l'ordre pouvant être croissant (ASC) ou décroissant (DESC), ASC étant l'ordre par défaut.
Si elle est présente, cette clause suit la clause WHERE ou la clause GROUP BY (paragraphe suivant).

Exemple :
Sortir la relation Stock triée selon l'ordre croissant des numéros de produits et l'ordre décroissant des quantités en stock des produits qui ne sont pas en rupture de stock.

SELECT * FROM Stock WHERE qte > 0 ORDER BY prod ASC, qte DESC

Les fonctions d'agrégation dans SQL

- COUNT ((ALL | DISTINCT) attribut ou expression) : compte le nombre d'attribut ou les résultats de l'expression

ALL permet de les compter tous, DISTINCT uniquement ceux qui sont différents

- SUM([ALL 1 DISTINCT] expression) : Somme d'une colonne.

- AVG([ALL DISTINCT] expression) : Moyenne d'une colonne.

- MAX([ALL DISTINCT] expression): Maximum d'une colonne.

- MIN([ALL DISTINCT] expression) : Minimum d'une colonne.

- Ces fonctions peuvent être utilisées dans une clause SELECT, si celle-ci ne rend pas des résultats individuels, c'est-à-dire si elle ne rend pas un ensemble de valeurs en plus de la valeur rendue par la fonction.
- Hormis COUNT(*), toutes les fonctions ci-dessus ignorent les valeurs nulles.
- Enfin, la variété des fonctions disponibles dépend du SGBD.

Exemples :

1. Nombre de produits en rupture de stock.

SELECT COUNT(*) FROM Stock WHERE qte <=0

2. Moyenne des prix unitaires des produits stockés dans le dépôt d4.

SELECT AVG(prix_unitaires) FROM Depot , Stock , Produit WHERE Depot . dep = `d4'
AND Depot .dep = Stock .dep AND Stock .prod = Produit .prod

3. Dépôt de plus grande capacité de stockage.

SELECT dep, adr FROM Depot WHERE volume IN (SELECT MAX(volume) FROM Depot))

La clause GROUP BY … HAVING

GROUP BY liste d'attributs [HAVING qualification avec fonction d'agrégation]

Cette clause permet de partitionner la relation résultat selon les valeurs d'un ou de plusieurs attributs.
La clause HAVING, quand elle est présente, joue un rôle similaire à celui d'une expression de qualification, mais appliquée au résultat du partitionnement

(i) la clause WHERE est évaluée pour ne retenir que les seuls t-uples la satisfaisant ;
(ii) il y a parlitionnement logique du résultat en des sous-relations comportant les t-uples ayant les mêmes valeurs pour les attributs de groupement spécifiés après le GROUP BY ;
(iii) si la clause HAVING est présente, elle est appliquée à chaque sous-relation et ne sont conservées dans la relation résultat que les sous-relations qui vérifient la condition qui suit le HAVING. En outre, si le résultat doit être trié, la clause ORDER BY doit apparaître après le GROUP BY.

Exemples

1. Somme des quantités en stock de chaque produit ?

SELECT prod, SUM(qte) FROM Stock GROUP BY prod

2. Quels sont les produits stockés dans au moins trois dépôts ?

SELECT prod FROM Stock GROUP BY prod HAVING COUNT(*) > 2

3. Somme des quantités en stock de chaque produit dans tous les dépôts de Nancy; trier le résultat sur prod#.

SELECT Stock.prod, SUM(qte) FROM Stock , Depot WHERE adr LIKE "Nancy" AND Depot.dep - Stock.dep GROUP BY Stock.prod ORDER BY Stock.prod

Le mot clé EXIST

SQL offre le mot-clé EXIST pour exprimer le quantificateur existentiel.
La disponibilité de la négation (NOT) permet de formuler des requêtes comportant le quantificateur universel ou l'implication.

Exemple :
La requête « Adresse des dépôts où est stocké le produit de numéro P35 » peut être exprimée de multiples façons, mais le mot-clé EXIST permet de simplifier énormément les choses :

SELECT Depot.dep, Depot.adr FROM Depot
WHERE EXIST(SELECT * FROM Stock WHERE .prod = 'P35' AND Stock.dep = Depot.dep)

L'Interprétation de la requête imbriquée étant faux si son résultat est vide, vrai sinon.

Exercices

Exercice 1

Soit le modèle relationnel suivant relatif à une base de données sur des représentations musicales :

REPRESENTATION (n°représentation, titre_représentation, lieu)
MUSICIEN (nom, n°représentation*)
PROGRAMMER (date, n°représentation*, tarif)

Remarque : les clés primaires sont soulignées et les clés étrangères sont marquées par *

Questions :
1 - Donner la liste des titres des représentations.
2 - Donner la liste des titres des représentations ayant lieu à l'opéra Bastille.
3 - Donner la liste des noms des musiciens et des titres des représentations auxquelles ils participent.
4 - Donner la liste des titres des représentations, les lieux et les tarifs pour la journée du 14/09/96.

Exercice 2

Soit le modèle relationnel suivant relatif à la gestion des notes annuelles d'une promotion d'étudiants :

ETUDIANT(N°Etudiant, Nom, Prénom)
MATIERE(CodeMat, LibelléMat, CoeffMat)
EVALUER(N°Etudiant*, CodeMat*, Date, Note)

Remarque : les clés primaires sont soulignées et les clés étrangères sont marquées par *

Questions :
1 - Quel est le nombre total d'étudiants ?
2 - Quelles sont, parmi l'ensemble des notes, la note la plus haute et la note la plus basse ?
3 - Quelles sont les moyennes de chaque étudiant dans chacune des matières ?
4 - Quelles sont les moyennes par matière ?
5 - Quelle est la moyenne générale de chaque étudiant ?
6 - Quelle est la moyenne générale de la promotion ?
7 - Quels sont les étudiants qui ont une moyenne générale supérieure ou égale à la moyenne générale de la promotion ?

Exercice 3

Soit le modèle relationnel suivant relatif à la gestion simplifiée des étapes du Tour de France 97, dont une des étapes de type "contre la montre individuel" se déroula à Saint-Etienne :

EQUIPE(CodeEquipe, NomEquipe, DirecteurSportif)
COUREUR(NuméroCoureur, NomCoureur, CodeEquipe*, CodePays*)
PAYS(CodePays, NomPays)
TYPE_ETAPE(CodeType, LibelléType)
ETAPE(NuméroEtape, DateEtape, VilleDép, VilleArr, NbKm, CodeType*)
PARTICIPER(NuméroCoureur*, NuméroEtape*, TempsRéalisé)
ATTRIBUER_BONIFICATION(NuméroEtape*, km, Rang, NbSecondes, NuméroCoureur*)

Remarque : les clés primaires sont soulignées et les clés étrangères sont marquées par *

Questions :
1 - Quelle est la composition de l'équipe Festina (Numéro, nom et pays des coureurs) ?
2 - Quel est le nombre de kilomètres total du Tour de France 97 ?
3 - Quel est le nombre de kilomètres total des étapes de type "Haute Montagne" ?
4 - Quels sont les noms des coureurs qui n'ont pas obtenu de bonifications ?
5 - Quels sont les noms des coureurs qui ont participé à toutes les étapes ?
6 - Quel est le classement général des coureurs (nom, code équipe, code pays et temps des coureurs) à l'issue des 13 premières étapes sachant que les bonifications ont été intégrées dans les temps réalisés à chaque étape ?
7 - Quel est le classement par équipe à l'issue des 13 premières étapes (nom et temps des équipes) ?

Instruction SQL de mise à jour

INSERT : ajout de n-uplets

L'opération d'insertion permet d'ajouter un ou plusieurs n-uplets à une relation. Dans le second cas, les n-uplets à insérer sont le résultat d'un SELECT.

INSERT INTO nom-relation [(liste de colonnes)]
{VALUES (expression-constante[, expression-constante[, ...]]) | clause select}

Exemples :

INSERT INTO Stock VALUES (4, 1234, 500)
INSERT INTO Produit(prod, libelle) VALUES (345, "Ecroul2")
INSERT INTO produit (SELECT * FROM AutreRelationProduit WHERE . . .)

Lorsque des attributs ne sont pas spécifiés dans la commande d'insertion, comme dans le deuxième exemple ci-dessus, ils sont généralement instanciés à la valeur inconnue (NULL).

UPDATE : Modification de n-uplets

UPDATE modifie des valeurs de colonnes dans une ou plusieurs lignes.

UPDATE nom de relation SET colonne = {expression | NULL | (clause select)}[, colonne2 = ...] [FROM ...] [WHERE qualification]

Exemples :

1. Mettre à 0 la quantité en stock du produit 1234 dans tous les dépôts.

UPDATE Stock SET qte = 0 WHERE 1234 IN (SELECT prod FROM Stock)

2. Même question pour tous les dépôts de Nancy.

UPDATE Stock SET qte = 0 FROM Depot , Stock
WHERE
Stock.dep = Depot.dep
AND
Stock.prod = 1234
AND
Depot.adr LIKE « Nancy »

DELETE :Suppression de n-uplets

Permet de supprimer des enregistrement d'une table
La suppression peut concerner tous les n-uplets d'une relation ou un sous-ensemble de n-uplets qui vérifie une condition de sélection.
Il faut noter que DELETE opère sur le contenu d'une relation, c'est-à-dire que le schéma d'une relation persiste après suppression de tous les n-uplets de la relation.

DELETE [FROM] nom de relation[, nom de relation[, . . .]] [WHERE qualification]

Exemples :

Dans la relation Stock, supprimer les n-uplets du dépôt d4.

DELETE Stock WHERE d4 IN (SELECT dep FROM Stock)

Manipulation de schéma de relation

Les instructions que nous avons vues jusqu'à présent concernent le contenu de relations. Celles qui suivent concernent les schémas de relations.

Instruction CREATE TABLE et DROP TABLE

La création d'un schéma de relation (table) consiste à nommer la relation, à en définir les attributs et à spécifier certaines contraintes d'intégrité.
Pour l'instant nous ne considérerons que la contrainte de valeur obligatoire (NOT NULL) qui, lorsqu'elle est apposée sur., un attribut, interdit à celui-ci d'avoir une valeur inconnue (NULL).

La forme syntaxique simplifiée de CREATE TABLE est :

```
CREATE TABLE nom-de-relation
(nom-attribut type attribut [NOT NULL | NULL] [,
nom attribut type attribut [NOT NULL | NULL] . . . )
```

Application à SYBASE

Sous le système Sybase, type-attribut peut être, entre autres

- int : entier signé représenté sur quatre octets ,
- float : virgule flottante ,
- char(longueur) : chaîne de longueur fixe ,
- varchar(longueur) : chaîne de longueur variable
- text : chaîne de longueur variable, d'au plus 231 - 1 caractères ;
- binary(longueur) : chaîne binaire de longueur fixe ;
- varbinary(longueur) : chaîne binaire de longueur variable ;
- image: chaîne binaire de longueur variable ;
- money : valeur monétaire positive ou négative ;
- datetime : date en représentation entière sur deux fois quatre octets, les premiers quatre octets contenant le nombre de jours avant (nombre négatif) ou après (nombre positif) la date de base du ler janvier 1900.

En outre, Sybase offre des possibilités (très limitées) de définir de nouveaux types de données qui pourront alors servir comme domaine des attributs des relations.

Exemple :

```
/*- Définition d'un type TypeNoProd pour typer prod */
/*- sp-addtype est une procédure système */

sp-addtype TypeNoProd, INT, "NOT NULL"

CREATE TABLE Produit (
 Prod#    TypeNoProdt,
 libelle  varchar(30),
 pu       float              NOT NULL)

CREATE TABLE Stock (
 prod#    TypeNoProd,
 dep#     int                NOT NULL,
 qte      int                NOT NULL)

CREATE TABLE Depot (
 dep#     int                NOT NULL,
 adr      varchar(20),
 volume   int)
```

Application à ORACLE

Sous le système Oracle, les types de données suivants font partie des types utilisables lors de la définition des attributs d'une relation

- CHAR(longueur) : chaîne de longueur fixe (longueur _- 255 caractères) ;
- {VARCHAR2 | VARCHAR}(longueur) : chaîne de longueur variable;
- NUMBER[(longueurl[, longueur2])] : virgule fixe ou flottante ;
- DAM: format standard jour-mois-an ;
- LONG: chaîne d'au plus deux giga-octets ;
- RAW (longueur) et LONG RA W: similaires à CHAR et à LONG respectivement, mais pour des chaînes héxadécimales.

Il est à noter que CREATE TABLE ne permet que la définition des schémas de relations, l'instanciation d'une relation pouvant se faire

1. par une suite d'insertions de n-uplets,
2. en utilisant un utilitaire de chargement « en masse » , c'est-à-dire un outil du SGBD qui permet d'instancier une relation à partir de données fournies dans un fichier, généralement de type texte (Oracle et Sybase offrent de tels utilitaires : SQLLoader et bcp, respectivement) ou
3. dynamiquement, c'est-à-dire par évaluation d'une requête, celle qui suit la clause AS dans une commande CREATE TABLE sous Oracle ou par le biais d'un SELECT... INTO relation-resultat FROM . . ., sous Sybase.

Pour supprimer un schéma de relation (ou table) on utilisera DROP TABLE <nom_relation>

Vues et relations temporaires

Une vue est une relation virtuelle (table virtuelle) au sens où ses instances n'existent pas physiquement mais elles sont calculées à chaque interrogation de la vue.
Une vue est définie par un requête qui utilise des relations ou des vues existantes.
Le schéma de la vue est déterminé par la liste de projection de sa requête de définition :

CREATE VIEW <nom de la vue> AS <clause SELECT>

En interrogation, une vue est utilisée comme toute autre relation (table), la seule différence réside dans le fait que ses n-uplets ne sont pas stockés mais calculés.
En mise à jour, toute modification des relations ayant servi à la définition de la vue est répercutée sur la vue.

Pour supprimer la vue on utilisera le mot clé DROP VIEW

Remarque : Dans Access, il n'y a pas de notion de vue, mais elle peut être associée à l'onglet requêtes dans lequel on peut enregistrer des requêtes et les utiliser ensuite comme des tables à part entière.

Index et clusters

Notion d'index

Un index est une structure associée à un ensemble de données dans le but d'accélérer l'accès à ces données. Il associe généralement une « adresse » à une valeur de l'ensemble, cette adresse permettant d'accéder directement à l'enregistrement qui contient la valeur. Dans les SGBD relationnels, les index sont le plus souvent représentés par des arbres équilibrés (Balanced Tree ou B-arbre), c'est-à-dire des arbres dont les feuilles sont toutes à égale distance de la racine (voir l'annexe A pour plus de détails).

Index de groupement physique (clustering)

Un index de clustering (figure 3.2) est un index construit sur un attribut dont les valeurs sont ordonnées et qui n'est pas forcément clé de la relation. De ce fait, les nuplets ayant la même valeur pour l'attribut de clustering se retrouveront rangés de façon contiguë. Ce type d'index est approprié pour les attributs de jointure afin de rapprocher physiquement les n-uplets à joindre et ce, en vue de minimiser les transferts entre la base de données et la mémoire centrale.