Module : Analyse et Conception de Base de Données

Documentation sur le MCD,MLD Et SQL

I. méthode d'analyse et de conception

une **méthode d'analyse et de conception** est un procédé qui a pour objectif de permettre de formaliser les étapes préliminaires du développement d'un système afin de rendre ce développement plus fidèle aux besoins du client. Pour ce faire, on part d'un énoncé informel (le besoin tel qu'il est exprimé par le client, complété par des recherches d'informations auprès des experts du domaine fonctionnel, comme les futurs utilisateurs d'un logiciel), ainsi que de l'analyse de l'existant éventuel (c'est-à-dire la manière dont les processus à traiter par le système se déroulent actuellement chez le client).

II. Merise

MERISE est une méthode de conception, de développement et de réalisation de projets informatiques.
Le but de cette méthode est d'arriver à concevoir un système d'information. La méthode MERISE est basée sur la séparation des données et des traitements à effectuer en plusieurs modèles conceptuels et physiques.

La séparation des données et des traitements assure une longévité au modèle. En effet, l'agencement des données n'a pas à être souvent remanié, tandis que les traitements le sont plus fréquemment.

III. UML

Le langage UML (Unified Modeling Language, ou langage de modélisation unifié) a été pensé pour être un langage de modélisation visuelle commun, et riche sémantiquement et syntaxiquement. Il est destiné à l'architecture, la conception et la mise en œuvre de systèmes logiciels complexes par leur structure aussi bien que leur comportement. L'UML a des applications qui vont au-delà du développement logiciel, notamment pour les flux de processus dans l'industrie.

Il ressemble aux plans utilisés dans d'autres domaines et se compose de différents types de diagrammes. Dans l'ensemble, les diagrammes UML décrivent la limite, la structure et le comportement du système et des objets qui s'y trouvent.

L'UML n'est pas un langage de programmation, mais il existe des outils qui peuvent être utilisés pour générer du code en plusieurs langages à partir de diagrammes UML. L'UML a une relation directe avec l'analyse et la conception orientées objet.

1. UML et son rôle dans la modélisation et la conception orientées objet

Il existe de nombreux modèles et paradigmes de résolution de problèmes en science informatique, qui est l'étude des algorithmes et des données. On dénombre quatre catégories de modèles de résolution de problèmes : les langages impératifs, fonctionnels, déclaratifs et orientés objet (LOO). Dans le cas des langages orientés objet, les algorithmes sont exprimés en définissant des objets et en les faisant interagir les uns avec les autres. Ces objets sont des éléments que l'on peut manipuler et qui existent dans le monde réel. Il peut s'agir d'immeubles, de widgets sur un ordinateur ou encore d'êtres humains.

Les langages orientés objet dominent le monde de la programmation parce qu'ils modélisent des objets du monde réel. L'UML combine plusieurs notations orientées objet : Object-Oriented Design (conception orientée objet), Object Modeling Technique (technique de modélisation objet) et Object-Oriented Software Engineering (génie logiciel orienté objet).

L'UML utilise les points forts de ces trois approches pour présenter une méthodologie plus cohérente et plus facile à utiliser. Il représente les meilleures pratiques de création et de documentation des différents aspects de la modélisation des systèmes logiciels et d'entreprise.

IV. Comparaison entre Merise et UML

MERISE (Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise) est une méthode d'analyse et de réalisaton des systèmes d'information qui est élaborée en plusieurs étapes: schéma directeur, étude préalable, étude détaillée et la réalisation.
Alors que UML (Unified Modeling Langage), est un langage de modélisation des systèmes standard, qui utilise des diagrammes pour représenter chaque aspect d'un systèmes ie: statique, dynamique,....en s'appuyant sur la notion d'orienté objet qui est un véritable atout pour ce langage.

1.Analyse et Conception

Après l'élection des besoins exprimés sous la forme de fonctionnalités modélisées comme des cas d'utilisation et sous la forme de scénarios modélisés dans des diagrammes d'activité, nous pouvons modéliser la structure logique du système, c'est-à-dire les aspects statiques du système. Cette modélisation est en grande partie effectuée dans des diagrammes de classes, avec éventuellement des diagrammes d'objets montrant des configurations spécifiques du système dans des conditions particulières. Le contenu principal de cette section est donc la présentation des éléments de modélisation du diagramme de classes. Avant d'introduire les diagrammes de classes et d'objets, et puisque ces deux diagrammes sont aussi utilisés pendant la phase suivant l'analyse, c'est-à-dire pendant la conception, nous expliquons en quelques mots les différences entre ces deux phases.

La présentation du diagramme de classes est effectuée en deux parties : les concepts de base et les concepts avancés. Pendant les bureaux d'étude et pendant le bureau d'étude noté, les solutions que vous concevez n'ont pas besoin d'utiliser les concepts avancés qui demandent plus de recul pour être mis en œuvre. Souvent, ces concepts sont introduits dans les diagrammes de classes après la première itération. Or, vous n'avez pas le temps de faire plus d'une itération dans tous les cas. Cependant, à la fin du premier bureau d'étude, nous présentons une seconde version du corrigé type utilisant une bonne partie de la panoplie des concepts avancés du diagramme de classes. C'est cette version qui ensuite sert de base aux autres bureaux d'étude et aux travaux pratiques. Ainsi, vous aurez l'occasion de manipuler les concepts avancés dès la modélisation.

V. MCD

Dans la méthodologie Merise destinée à créer des bases de données, il y a des outils dédiés aux traitements et aux données. Le **MCD** (**Modèle Conceptuel des Données**) est un des outils majeurs concernant les données.

● Les entités

Dans la méthode Merise au niveau du MCD, une entité correspond à un **ensemble homogène d'informations** qui correspondent au même « objet » à informatiser. Cette **entité** a un nom unique afin de la manipuler facilement. Plus tard dans l'analyse, l'entité se transforme en table et devient concrètement une table lors de la réalisation effective de la base de données.

Cet ensemble d'informations, **l'entité**, partage les mêmes caractéristiques et peut être manipulé au sein du système d'information mais aussi en discutant entre informaticiens et personnes du métier. L'entité « *contrat* » et l'entité « *client* » sont typiques des bases de données de gestion. Isoler, comprendre le rôle et les informations qui se trouvent dans ces coquilles est une des premières étapes de l'analyse.

Une **entité** reste cependant peu utile si elle n'est pas remplie avec des informations calibrées et si elle ne discute pas avec les autres entités de la base de données : on parle alors de propriétés et d'associations ce qui a donné le nom de modèle entités-associations que l'on croise parfois.

Dans un MCD, les entités sont représentés par des rectangles et dans le cartouche supérieur de chaque rectangle est écrit au singulier le nom de l'entité.

● Les attributs

Les attributs d'entité sont les éléments d'information de base attachés à une entité. Lorsque vous générez un Modèle Physique de Données (MPD) à partir d'un Modèle Conceptuel de Données (MCD), les attributs d'entité deviennent des colonnes de tables dans le MPD.

● Les Occurrences

Il ne faut pas confondre
- ENTITE et OCCURRENCE d'entité,
- ni PROPRIETE et VALEUR de propriété
- ni ASSOCIATION et OCCURRENCE d'association

Reprenons ces concepts
Une ENTITE est une FAMILLE d'objets ayant les mêmes caractéristiques, appelées propriétés.
Un MEMBRE de la famille est appelé OCCURRENCE d'entité.
Une entité représente un ENSEMBLE d'OCCURRENCES.
Une PROPRIETE est une information élémentaire qui permet de décrire une entité ou une association. Une propriété peut prendre une VALEUR (c'est l'équivalent d'une variable).
(On peut dire qu'une VALEUR est une OCCURRENCE de PROPRIETE)
De même, une ASSOCIATION est un LIEN entre 2 entités ou plus, et une OCCURRENCE d'association est un lien entre 2 OCCURRENCES d'entités.

● Les Cardinalités

3 cardinalités (2 autour de l'association « se_situe » et une dernière tronquée)

Les cardinalités caractérisent le lien entre une entité et une association. La cardinalité d'uneassociation est le nombre de fois minimal et maximal qu'une occurrence d'une des entités associée peut intervenir dans l'association :

- *minimale*: nombre minimum de fois qu'une occurrence d'une entité participe aux occurrences de l'association, généralement **0** ou **1**.

- *maximale* : nombre maximum de fois qu'une occurrence d'une entité participe aux occurrencesde l'association, généralement **1** ou **n**.

Les cardinalités maximales sont nécessaires pour la création de la base de données. Les cardinalités minimales sont nécessaires pour exprimer les contraintes d'intégrités[22(*)].

● **Formalisme de Représentation**

FORMALISATION DU MCD

I) Concepts de base

Le formalisme utilisé dans MERISE est désigné par individu-relation o entité/association.
Ce formalisme comporte trois concepts de base :

- l'individu ou entité,
- la relation ou association,
- la propriété ou champ.

I.1) Représentation graphique du formalisme

Entité — association — entité

Champ : n°personne, prénom, âge — date début — N°logement, adresse, type, surface

VI. MLD

Le **MLD** ou **Modèle Logique des Données** est simplement la représentation textuelle du MPD. Il s'agit juste de la représentation en ligne du schéma représentant la structure de la base de données. Il n'y a pas de travail poussé à réaliser à cette étape, il s'agit juste d'appliquer quelques règles toutes simples. Cette étape est parfois omise.

● MLDR

● Modele Relationnel

Le **modèle relationnel** est une manière de modéliser les relations existantes entre plusieurs informations, et de les ordonner entre elles. Cette modélisation qui repose sur des principes mathématiques mis en avant par E.F. Codd est souvent retranscrite physiquement (« implémentée ») dans une base de données.

● Règles de passage du MCD au MLDR

Règle numéro 1 :
a) Une entité du MCD devient une relation, c'est à dire une table.
b) Son identifiant devient la clé primaire de la relation.
c) Les autres propriétés deviennent les attributs de la relation.
Règle numéro 2 :

Une association de type 1:N (c’est à dire qui a les cardinalités maximales positionnées à « 1 » d’une côté de l’association et à « n » de l’autre côté) se traduit par la création d’une clé étrangère dans la relation correspondante à l’entité côté « 1 ». Cette clé étrangère référence la clé primaire de la relation correspondant à l’autre entité.
Règle numéro 3 : Une association de type N :N (c’est à dire qui a les cardinalités maximales positionnées à « N » des 2 côtés de l’association) se traduit par la création d’une table dont la clé primaire est composée des clés étrangères référençant les relations correspondant aux entités liées par l’association. Les éventuelles propriétés de l’association deviennent des attributs de la relation.

● Formalisme de Représentation

Table : conserve le formalisme de l'objet ;

Table de lien : pointe les autres tables ;

Tous les arcs sont orientés.

I. IV) SQL

Le **SQL** (Structured Query Language) est un langage permettant de communiquer avec une base de données. Ce langage informatique est notamment très utilisé par les développeurs web pour communiquer avec les données d’un site web

Langage SQL

- Contenu
  - LDD
  - LMD
  - LCD

Introduction

- SQL (Structured Query Language) est un langage informatique normalisé servant à exploiter des bases de données.
- SQL est basée sur l'algèbre relationnelle (opérations ensemblistes et relationnelles).
- SQL conçu par IBM dans les années 70, a été normalisé dès 1986 mais les premières normes, trop incomplètes, ont été ignorées par les éditeurs de SGBD.
- La norme actuelle SQL-2011 est acceptée par tous les SGBD relationnels.
- Ce langage permet l'accès aux données et se compose de quatre sous-ensembles :

Borchani Anis

Introduction

LDD : « Langage de Définition de Données » : permet la définition et la mise à jour de la structure de la base de données (tables, attributs, vues, index, ...).

LID : « Langage d'Interrogation de Données » : permet de rechercher des informations utiles en interrogeant la base de données.

LMD : « Langage de Manipulation de Données » : permet de manipuler les données de la base et de les mettre à jour. confirmer et d'annuler les transactions.

Introduction

Exemple de base de données

Borchani Anis

Produit (NP, LibP, Coul, Poids, PU, Qtes) - Désigne l'ensemble des produits.

Client (NCl, NomCl, AdrCl) - Désigne l'ensemble des clients.

Commande (NCmd, DateCmd, #NCl) - Désigne l'ensemble des commandes.

Ligne_Cmd (#NCmd, #NP, Qte) - Désigne l'ensemble des lignes de commandes.

Exemple de base de données

Borchani Anis

Client

NCl	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUDURE	Tunis
CL10	MELEC	Monastir
CL11	MBATIM	
CL12	BATFER	Tunis

Commande

NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

Ligne_Cmd

NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

Produit

NP	LibP	Coul	Poids	PU	Qtes
P001	Robinet	Gris	5	18.000	1200
P002	Prise	Blanc	1.2	1.500	1000
P003	Câble	Blanc	2	25.000	1500
P004	Peinture	Blanc	25	33.000	900
P005	Poignée	Gris	3	12.000	1300
P006	Serrure	Jaune	2	47.000	1250
P007	Verrou	Gris	1.7	5.500	2000
P008	Fer	Noir	50	90.000	800

Types de données

Borchani Anis

qlqs types valides pour Oracle

Contraintes d'intégrité

Borchani Anis

Une contrainte d'intégrité est une règle qui permet de contrôler la validité et la cohérence des valeurs entrées dans les différentes tables de la base.

Elle peut être définie sous deux formes :

- Dans les commandes de création des tables.
- Au moment de la modification de la structure de la table.

TYPE	DESCRIPTION	Minimum	Maximum	Exemples
VARCHAR2 (taille) NVARCHAR2 (taille)	Chaîne de caractères de longueur variable	1 car.	2000 car.	'A' 'Bonjour DD'
NUMBER [(taille[,precision])]	Numérique. (prec<=38, exposant max -84 +127)	10 exp -84	10 exp 127	10.9999
LONG (taille)	Chaîne de caractères de longueur variable.	1 car.	2 giga car.	'AAAHHHHHH...HHH'
DATE	Date (du siècle à la seconde)	01/01/-4712 (avant J.C)	31/12/9999	'10-FEB-04' '10/02/04'
RAW (taille)	Données binaires devant être entrées en notation hexadécimale. Taille : 1 à 255 caractères	1 octet	2000 octets	
LONG RAW (taille)	Données binaires devant être entrées en notation hexadécimale.	1 octet	2 GO	
ROWID	Type réservé à la pseudo-colonne ROWID. Ne peut être utilisé.			
CHAR (taille) NCHAR (taille)	Chaîne de caractères de longueur fixe	1 car.	255 car.	'AZERTY' 'W'
BLOB	BLOB ! gros objet binaire	1 octet	4 giga car.	
TIMESTAMP	Date et heure			

Contraintes d'intégrité sur un attribut

La contrainte porte sur un seul attribut.

Ces contraintes sont :

NOT NULL : Spécifie que pour toute occurrence, l'attribut doit avoir une valeur (la saisie de ce champ est obligatoire).

UNIQUE : Toutes les valeurs de l'attribut sont distinctes.

PRIMARY KEY : L'attribut est une clé primaire pour la table.

REFERENCES table (attribut) : Il s'agit d'une contrainte d'intégrité fonctionnelle par rapport à une clé ; chaque valeur de l'attribut doit exister par toutes les valeurs de l'attribut (domaine des valeurs de l'attribut).

Contraintes d'intégrité sur une table

La contrainte porte sur un ensemble d'attributs d'une même table.

Ces contraintes sont :

UNIQUE (attri, attrj,…) : L'unicité porte sur le n-uplet des valeurs.

PRIMARY KEY (attri, attrj,…) : Clé primaire de la table (clé composée).

FOREIGN KEY (attri, attrj, …) REFERENCES table (attrm, attrn, …) [ON DELETE { CASCADE | SET NULL }] : Désigne une clé étrangère sur plusieurs attributs.

L'option ON DELETE CASCADE indique que la suppression d'une ligne de la table de référence va entraîner automatiquement la suppression des lignes référencées.

Création d'une table (1/3)

Syntaxe

CREATE TABLE nom_table

(Attribut1 Type [Contrainte d'attribut],

Attribut2 Type [Contrainte d'attribut],

…

Attributn Type [Contrainte d'attribut],

[Contrainte de relation], …);

Création d'une table (2/3)

Application

Création de la table Client en donnant à sa clé NCl les propriétés NOT NULL et UNIQUE.

CREATE TABLE Client

(NCl VARCHAR2(4) NOT NULL UNIQUE,
NomCl VARCHAR2(15),

AdrCl VARCHAR2(10));

Création de la table Produit en définissant la contrainte pk_NP de clé primaire équivalente à l'attribution des propriétés NOT NULL et UNIQUE. Ne pas définir l'attribut Qtes ceci sera défini ultérieurement par l'instruction ALTER TABLE.

Création d'une table (3/3)

Application

Création de la table Commande sans définir de contrainte ni de propriétés sur la clé, ceci sera défini ultérieurement par l'instruction ALTER TABLE.

CREATE TABLE Commande
(NCmd VARCHAR2(4),

DateCmd DATE,
NCl VARCHAR2(4));

Création de la table Ligne_Cmd en spécifiant la contrainte de clé primaire et l'une des contraintes de clé étrangère.

Modification de la structure d'une table (1/6)

Ajout d'un attribut : Permet d'ajouter un attribut à la structure initiale de la table.

Syntaxe :

ALTER TABLE Nom_Table

ADD Attribut Type**;**

Application :

Ajout de l'attribut Qtes à la table Produit.

ALTER TABLE Produit

Modification de la structure d'une table (3/6)

Suppression d'un attribut : Permet de supprimer un attribut d'une table.

Syntaxe :

ALTER TABLE Nom_Table

DROP COLUMN Attribut;

Il faut noter que la suppression d'attributs (colonnes) n'est possible que dans le cas où :

- L'attribut ne fait pas partie d'une vue,
- L'attribut ne fait pas partie d'un index,
- L'attribut n'est pas l'objet d'une contrainte d'intégrité.

Modification de la structure d'une table (2/6)

Bouchenfl/Arifa

Modification d'un attribut : Associée avec la clause **MODIFY**, la clause ALTER permet la modification du type de données d'un attribut. On ne peut qu'agrandir la taille d'un attribut.

Syntaxe :

ALTER TABLE Nom_Table

MODIFY Attribut Nouveau_Type;

Application :

Modification de la taille de l'attribut LibP à VARCHAR(20).

ALTER TABLE Produit

Modification de la structure d'une table (4/6)

Bordeau/Anita

Ajout de contrainte: Permet d'ajouter une contrainte au niveau d'une table.

Syntaxe :

ALTER TABLE Nom_Table

ADD CONSTRAINT Nom_Contrainte Définition_Contrainte;

Applications :

Ajout de la contrainte de PRIMARY KEY sur l'attribut NCmd de la table Commande.

ALTER TABLE Commande

ADD CONSTRAINT pk_NCMD PRIMARY KEY (NCmd);

Modification de la structure d'une table (5/6)

Bordeau/Anita

Ajout de contrainte:

Applications :

Ajout des contraintes de clés étrangères sur la table Ligne_Cmd.

ALTER TABLE Ligne_Cmd

ADD CONSTRAINT fk_NCMD FOREIGN KEY (NCmd) REFERENCES Commande(NCmd)**;**

Modification de la structure d'une table (6/6)

Bordeau/Anita

Suppression de contrainte : Permet de supprimer une contrainte.

Syntaxe :

ALTER TABLE Nom_Table

DROP CONSTRAINT Nom_Contrainte;

Désactivation d'une contrainte : Permet de désactiver une contrainte, elle est par défaut active (au moment de sa création).

Syntaxe :

ALTER TABLE Nom_Table

Création d'index

Bordeau/Anita

- La création d'un index permet d'accélérer les recherches d'informations dans la base.
- La ligne est retrouvée instantanément si la recherche peut utiliser un index.
- Sinon la recherche se fait séquentiellement.
- Une autre utilité de la création d'index est de garantir l'unicité de la clé en utilisant l'option UNIQUE.

Syntaxe :

CREATE [UNIQUE] **INDEX** nom_index

ON nom_table (Attr1[ASC/DESC], Attr2[ASC/DESC], …);

- L'option **UNIQUE** permet de définir la présence ou non de doublons pour les valeurs de l'attribut.
- Les options **ASC/DESC** permettent de définir un ordre de classement des valeurs présentes dans l'attribut.

Suppression d'une table / un index

La clause **DROP** permet de supprimer des vues, des index et même des tables. Cette clause est toutefois à utiliser avec précaution dans la mesure où elle est irréversible.

Syntaxe :

Pour supprimer un index : **DROP** INDEX Nom_Index [ON Nom_Table];

Pour supprimer une table : **DROP** TABLE Nom_table ;

NB :

- Le nom de la table est obligatoire si on veut supprimer un index d'une table d'un autre utilisateur.

Langage de Manipulation de Données

Le LMD est un ensemble de commandes permettant de:

- l'insertion de nouvelles données. **(Insert)**
- la modification. **(Update)**
- la suppression de données existantes. **(Delete)**

Insertion de données

Syntaxe :

INSERT INTO Nom_Table [(Attr1, Attr2,…, Attrn)]

Permet d'insérer un tuple à la fois.

INSERT INTO Nom_Table (Attr1, Attr2, …, Attrn)

Permet d'insérer plusieurs tuple à partir d'une ou plusieurs autres tables.

L'ordre INSERT attend la clause INTO, suivie du nom de la table, ainsi que du nom de chacun des attributs entre parenthèses (les attributs omis prendront la valeur NULL par défaut).

Les données sont affectées aux attributs (colonnes) dans l'ordre dans lequel les

Insertion de données

Application :

Remplissage de la table Client.

INSERT INTO Client VALUES('CL01','BATAM','Tunis');

INSERT INTO Client VALUES('CL02','BATIMENT','Tunis');

INSERT INTO Client VALUES('CL03','AMS','Sousse');

INSERT INTO Client VALUES('CL04','GLOULOU','Sousse');

INSERT INTO Client VALUES('CL05','PRODELEC','Tunis');

INSERT INTO Client VALUES('CL06','ELECTRON','Sousse');

INSERT INTO Client VALUES('CL07','SBATIM','Sousse');

INSERT INTO Client VALUES('CL08','SANITAIRE','Tunis');

INSERT INTO Client VALUES('CL09','SOUDURE','Tunis');

INSERT INTO Client VALUES('CL10','MELEC','Monastir');

INSERT INTO Client VALUES('CL11','MBATIM','');

Insertion de données

Application :

Remplissage de la table Produit.

INSERT INTO Produit VALUES('P001','Robinet','Gris',5,18,1200);

INSERT INTO Produit VALUES('P002','Prise','Blanc',1.2,1.5,1000);

INSERT INTO Produit VALUES('P003','Câble','Blanc',2,25,1500);

INSERT INTO Produit VALUES('P004','Peinture','Blanc',25,33,900);

INSERT INTO Produit VALUES('P005','Poignée','Gris',3,12,1300);

INSERT INTO Produit VALUES('P006','Serrure','Jaune',2,47,1250);

INSERT INTO Produit VALUES('P007','Verrou','Gris',1.7,5.5,2000);

INSERT INTO Produit VALUES('P008','Fer','Noir',50,90,800);

Bordeaux Aquitaine

Insertion de données

Application :

Remplissage de la table Commande.

INSERT INTO Commande VALUES('C001', '10/12/2003', 'CL02');

INSERT INTO Commande VALUES('C002', '13/02/2004', 'CL05');

INSERT INTO Commande VALUES('C003', '15/01/2004', 'CL03');

INSERT INTO Commande VALUES('C004', '03/09/2003', 'CL10');

INSERT INTO Commande VALUES('C005', '11/03/2004', 'CL03');

Bordeaux Aquitaine

Insertion de données

Application :

Remplissage de la table Ligne_Cmd.

```
INSERT INTO Ligne_Cmd VALUES('C001','P001',250);
INSERT INTO Ligne_Cmd VALUES('C001','P004',300);
INSERT INTO Ligne_Cmd VALUES('C001','P006',100);
INSERT INTO Ligne_Cmd VALUES('C002','P002',200);
INSERT INTO Ligne_Cmd VALUES('C002','P007',550);
INSERT INTO Ligne_Cmd VALUES('C003','P001',50);
INSERT INTO Ligne_Cmd VALUES('C004','P002',100);
INSERT INTO Ligne_Cmd VALUES('C004','P004',150);
INSERT INTO Ligne_Cmd VALUES('C004','P005',70);
INSERT INTO Ligne_Cmd VALUES('C004','P008',90);
INSERT INTO Ligne_Cmd VALUES('C005','P001',650);
```


Borchani Anis

Modification de données

Syntaxe :

UPDATE Nom_Table

SET Attr1 = Expr1, Attr2 = Expr2, …

WHERE Condition;

ou

UPDATE Nom_Table

SET (Attr1, Attr2, …) = (**SELECT** …)

WHERE Condition;

La modification à effectuer est précisée après la clause SET. Il s'agit d'une affectation d'une valeur à un attribut grâce à l'opérateur = suivi d'une expression algébrique, d'une constante ou du résultat provenant d'une clause SELECT.

La clause WHERE permet de préciser les tuples sur lesquels la mise à jour aura lieu.

Borchani Anis

Modification de données

Application :

Modifier le Poids du produit numéro P002 à 1.

UPDATE Produit

SET Poids = 1

WHERE NP = 'P002';

Augmenter la quantité en stock des différents produits de 10%.

UPDATE Produit

Suppression de données

L'ordre DELETE permet de supprimer des lignes d'une table.

Syntaxe :

DELETE FROM Nom_Table

WHERE Condition;

Application :

Suppression de données

Application :

Supprimer les produits blanc avec un poids <5.

DELETE FROM produit

WHERE coul='blanc' AND poids<5;

Supprimer tous les commandes des clients de sousse.

DELETE FROM commande

Langage d'interrogation de Données

La principale commande du langage d'interrogation de données est la commande SELECT.

La commande SELECT est basée sur l'algèbre relationnelle, en effectuant des opérations de sélection de données sur plusieurs tables relationnelles par projection. Sa syntaxe générale est la suivante :

La commande Select

Syntaxe :

SELECT [ALL] | [DISTINCT] <liste des attributs> | *

FROM <liste des tables>

[**WHERE** <condition>]

[**GROUP** BY <liste des attributs>]

[.....] : optionnel

Borchani Anis

La commande Select : Options

- L'option **ALL** est, par opposition à l'option **DISTINCT**, l'option par défaut. Elle permet de sélectionner l'ensemble des lignes satisfaisant à la condition logique.
- L'option **DISTINCT** permet de ne conserver que des lignes distinctes, en éliminant les doublons.
- La liste des attributs indique la liste des attributs choisis, séparés par des virgules.
- L'option * permet de sélectionner l'ensemble des colonnes d'une table.
- La liste des tables indique l'ensemble des tables (séparées par des virgules) sur lesquelles on opère.
- La **condition** permet d'exprimer des qualifications complexes à l'aide d'opérateurs logiques et de comparateurs arithmétiques.

Borchani Anis

Projection

Une projection est une instruction permettant de sélectionner un ensemble d'attributs dans une table.

Syntaxe :

SELECT [ALL] | [DISTINCT] <liste des attributs> | *

FROM <la table> ;

Application 1 :

SELECT *
FROM Client ;

Client

NCl	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUDURE	Tunis
CL10	MELEC	Monastir
CL11	MBATIM	
CL12	BATFER	Tunis

NCl	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUDURE	Tunis
CL10	MELEC	Monastir
CL11	MBATIM	
CL12	BATFER	Tunis

Borchani Anis

Projection

Application 2 :

✓ Donner l'ensemble des produits qui ont été commandés (NP seulement).

SELECT NP
FROM Ligne_Cmd;

SELECT DISTINCT NP
FROM Ligne_Cmd;

Ligne_Cmd

NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

NP
P001
P004
P006
P002
P007
P001
P002
P004
P005
P008
P001
P002

NP
P001
P004
P006
P002
P007
P005
P008

DISTINCT est utilisé pour éliminer les duplications, par défaut on obtient tous les tuples (**ALL**).

Borchani Anis

Restrictions

Une restriction consiste à sélectionner les lignes satisfaisant à une condition logique effectuée sur leurs attributs.

les restrictions s'expriment à l'aide de la clause WHERE suivie d'une condition logique exprimée à l'aide:

- Opérateurs logiques (AND, OR, NOT)
- Opérateurs arithmétiques (+, -, *, /, %)
- Comparateurs arithmétiques (=, !=, >, <, >=, <=)
- Prédicats (NULL, IN, BETWEEN, LIKE, ALL, SOME, ANY, EXISTS).

Ces opérateurs s'appliquent aux valeurs numériques, aux chaînes de caractères et aux dates.

Syntaxe :

La clause WHERE (1/2)

WHERE exp1 = exp2	Condition est vraie si les deux expressions exp1 et exp2 sont égales.
WHERE exp1 != exp2	Condition est vraie si les deux expressions exp1 et exp2 sont différentes.
WHERE exp1 < exp2	Condition est vraie si exp1 est inférieure à exp2.
WHERE exp1 > exp2	Condition est vraie si exp1 est supérieure à exp2.
WHERE exp1 <= exp2	Condition est vraie si exp1 est inférieure ou égale à exp2.
WHERE exp1 >= exp2	Condition est vraie si exp1 est supérieure ou égale à exp2.
WHERE exp1 **BETWEEN** exp2 **AND** exp3	Condition est vraie si exp1 est comprise entre exp2 et exp3, bornes incluses.
WHERE exp1 **LIKE** exp2	Condition est vraie si la sous–chaîne exp2 est présente dans exp1.
WHERE exp1 **NOT LIKE** exp2	Condition est vraie si la sous–chaîne exp2 n'est pas présente dans exp1.

La clause WHERE (2/2)

WHERE exp1 **IN** (exp2, exp3,…)	Condition est vraie si exp1 appartient à l'ensemble (exp2, exp3, …).
WHERE exp1 **NOT IN** (exp2, exp3,…)	Condition est vraie si exp1 n'appartient pas à l'ensemble (exp2, exp3, …).
WHERE exp1 **IS NULL**	Condition est vraie si exp1 est nulle.
WHERE exp1 **IS NOT NULL**	Condition est vraie si exp1 n'est pas nulle.
WHERE exp1 Op **ALL** (exp2, exp3,…)	Op représente un opérateur de comparaison (<, >, =, !=, <=, >=) Condition est vraie si la comparaison de l'exp1 est vraie avec toutes les valeurs de la liste (exp2, exp3, …). Si la liste est vide, le résultat est vrai.
WHERE exp1 Op **ANY** (exp2, exp3,…) **WHERE** exp1 Op **SOME** (exp2, exp3,…)	Op représente un opérateur de comparaison (<, >, =, !=, <=, >=) Condition est vraie si la comparaison de l'exp1 est vraie avec au moins une valeur de la liste (exp2, exp3, …). Si la liste est vide, le résultat est faux.
WHERE EXISTS sous-requête	Condition est vraie si le résultat de la sous-requête n'est pas vide.

Restrictions : Applications

Application 1:

✓ Donner le numéro et le nom des clients de la ville de Sousse.

SELECT NCl, NomCl
FROM Client
WHERE AdrCl = 'Sousse';

Client

NCl	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUDURE	Tunis
CL10	MELEC	Monastir
CL11	MBATIM	
CL12	BATFER	Tunis

NCl	NomCl
CL03	AMS
CL04	GLOULOU
CL06	ELECTRON
CL07	SBATIM

Restrictions : Applications

Application 2:

✓ Donner la liste des commandes dont la date est supérieure à '01/01/2004'.

SELECT *
FROM Commandes
WHERE DateCmd > '01/01/2004';

Commande

NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

NCmd	DateCmd	NCl
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C005	11/03/2004	CL03

Restrictions : Applications

Application 3:

✓ Donner la liste des produits dont le prix est compris entre 20 et 50.

SELECT *
FROM Produit
WHERE PU **BETWEEN** 20 **AND** 50;

Ou bien

SELECT *
FROM Produit
WHERE (PU >= 20) **AND** (PU <= 50);

Produit

NP	LibP	Coul	Poids	PU	Qtes
P001	Robinet	Gris	5	18.000	1200
P002	Prise	Blanc	1.2	1.500	1000
P003	Câble	Blanc	2	25.000	1500
P004	Peinture	Blanc	25	33.000	900
P005	Poignée	Gris	3	12.000	1300
P006	Serrure	Jaune	2	47.000	1250
P007	Verrou	Gris	1.7	5.500	2000
P008	Fer	Noir	50	90.000	800

NP	LibP	Coul	Poids	PU	Qtes
P003	Câble	Blanc	2	25.000	1500
P004	Peinture	Blanc	25	33.000	900
P006	Serrure	Jaune	2	47.000	1250

Restrictions : Applications

Application 4:

✓ Donner la liste des clients dont les noms commencent par 'B'.

SELECT *
FROM Client
WHERE NomCl **LIKE** 'B%';

Client

NCl	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUDURE	Tunis
CL10	MELEC	Monastir
CL11	MBATIM	
CL12	BATFER	Tunis

NCl	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL12	BATFER	Tunis

Le prédicat Like

- Le prédicat **LIKE** permet de faire des comparaisons sur des chaînes grâce à des caractères, appelés caractères jokers :
- Le caractère **%** permet de remplacer une séquence de caractères (éventuellement nulle).
- Le caractère **_** permet de remplacer un caractère.
- Les caractères [-] permettent de définir un intervalle de caractères (par exemple [J-M]).
- La sélection des clients dont les noms ont un E en deuxième position se fait par l'instruction : *WHERE NomCl LIKE "_E%"*

Restrictions : Applications

Application 5:

✓ Donner les numéros des clients dont les dates de leurs commandes se trouvent parmi les dates suivantes : ('10-12-03', '10-12-04','13-02-04','11-03-04').

SELECT DateCmd, NCl
FROM Commande
WHERE DateCmd **IN** ('10-12-2003','10-12-2004','13-02-2004','11-03-2004');

Commande

NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

DateCmd	NCl
10/12/2003	CL02
13/02/2004	CL05
11/03/2004	CL03

Restrictions : Applications

Application 6:

✓ Donner les noms des clients qui n'ont pas d'adresse.

SELECT NomCl
FROM Client
WHERE AdrCl **IS NULL;**

Client

NCl	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUDURE	Tunis
CL10	MELEC	Monastir
CL11	MBATIM	
CL12	BATFER	Tunis

NomCl
MBATIM

*Lorsqu'un champ n'est pas renseigné, le SGBD lui attribue une valeur spéciale que l'on note **NULL**. La recherche de cette valeur ne peut pas se faire à l'aide des opérateurs standards, il faut utiliser les prédicats **IS NULL** ou bien **IS NOT NULL**.*

Les Alias

Les alias permettent de renommer des attributs ou des tables.

*SELECT attr1 **AS** aliasa1, attr2 **AS** aliasa2, …*

FROM table1 aliast1, table2 aliast2… ;

Pour les attributs, l'alias correspond aux titres des colonnes affichées dans le résultat de la requête. Il est souvent utilisé lorsqu'il s'agit d'attributs calculés (expression).

NB : Le mot clé AS est optionnel

Les Alias

Application :

✓ Donner la valeur des produits en stock.

SELECT NP, (Qtes * PU) **AS** "Valeur Totale"
FROM Produit ;

Produit

NP	LibP	Coul	Poids	PU	Qtes
P001	Robinet	Gris	5	18.000	1200
P002	Prise	Blanc	1.2	1.500	1000
P003	Câble	Blanc	2	25.000	1500
P004	Peinture	Blanc	25	33.000	900
P005	Poignée	Gris	3	12.000	1300
P006	Serrure	Jaune	2	47.000	1250
P007	Verrou	Gris	1.7	5.500	2000
P008	Fer	Noir	50	90.000	800

NP	Valeur Totale
P001	21600
P002	1500
P003	37500
P004	29700
P005	15600
P006	58750
P007	11000
P008	72000

Les fonctions

Le tableau suivant donne le nom des principales fonctions prédéfinies :

Nom de la fonction	Rôle de la fonction
AVG	Moyenne
SUM	Somme
MIN	Minimum
MAX	Maximum
COUNT (*)	Nombre de lignes
COUNT (Attr)	Nombre de valeurs non nulles de l'attribut
COUNT([DISTINCT] Attr)	Nombre de valeurs non nulles différentes de l'attribut

✓ Donner la moyenne des prix unitaires des produits.

SELECT AVG(PU)
FROM Produit;

Produit

NP	LibP	Coul	Poids	PU	Qtes
P001	Robinet	Gris	5	18.000	1200
P002	Prise	Blanc	1.2	1.500	1000
P003	Câble	Blanc	2	25.000	1500
P004	Peinture	Blanc	25	33.000	900
P005	Poignée	Gris	3	12.000	1300
P006	Serrure	Jaune	2	47.000	1250
P007	Verrou	Gris	1.7	5.500	2000
P008	Fer	Noir	50	90.000	800

AVG(PU)
29

Application :

Borchani Anis

Sélection avec **jointure**

Il s'agit ici de sélectionner les données provenant de plusieurs tables ayant un ou plusieurs attributs communs. Cette jointure sera assurée grâce aux conditions spécifiées dans la clause WHERE.

Syntaxe :

SELECT [ALL] | [DISTINCT] <liste des attributs> | *

FROM <liste des tables>

WHERE *Nom_Table1.Attrj = Nom_Table2.Attrj* AND …

AND <condition> ;

Borchani Anis

Sélection avec **jointure**

Application 1:

✓ Donner les libellés des produits de la commande numéro 'C002'.

```
SELECT DISTINCT LibP
FROM Produit, Ligne_Cmd
WHERE Produit.NP = Ligne_Cmd.NP
      AND NCmd='C002';
```

Ou bien

```
SELECT DISTINCT LibP
FROM Produit P, Ligne_Cmd L
WHERE P.NP = L.NP
      AND NCmd='C002';
```

Produit

NP	LibP	Coul	Poids	PU	Qtes
P001	Robinet	Gris	5	18.000	1200
P002	Prise	Blanc	1.2	1.500	1000
P003	Câble	Blanc	2	25.000	1500
P004	Peinture	Blanc	25	33.000	900
P005	Poignée	Gris	3	12.000	1300
P006	Serrure	Jaune	2	47.000	1250
P007	Verrou	Gris	1.7	5.500	2000
P008	Fer	Noir	50	90.000	800

Ligne_Cmd

NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

LibP
Prise
Verrou

On peut exprimer cette jointure autrement ?

Borchani Anis

Sélection avec **jointure**

Application 2:

✓ Donner les produits (toutes les informations) commandés au cours de l'année 2003 et qui sont vendus aux clients de Tunis.

```
SELECT DISTINCT P.NP, LibP, Coul, Poids, PU, Qtes
FROM Produit P, Commande C, Client Cl, Ligne_Cmd L
WHERE P.NP = L.NP
      AND C.NCmd = L.NCmd
      AND Cl.NCl = C.NCl
      AND Cl.AdrCl ='Tunis'
      AND DateCmd BETWEEN '01/01/2003' AND '31/12/2003';
```

NP	LibP	Coul	Poids	PU	Qtes
P001	Robinet	Gris	5	18	1200
P004	Peinture	Blanc	25	33	900
P006	Serrure	Jaune	2	47	1250

Borchani Anis

Groupement

Il est possible de grouper des lignes de données ayant une valeur commune à l'aide de la clause GROUP BY et des fonctions de groupe.

Syntaxe :

SELECT Attr1, Attr2,…, Fonction_Groupe
FROM Nom_Table1, Nom_Table2,…
WHERE Liste_Condition

GROUP BY Liste_Groupe

HAVING Condition;

- La clause **GROUP BY**, suivie du nom de chaque attribut sur laquelle on veut effectuer des regroupements.
- La clause **HAVING** va de pair avec la clause GROUP BY, elle permet d'appliquer une restriction sur les groupes créés grâce à la clause GROUP BY.

Bouchenf Anis

Groupement

Application 1 :

✓ Donner le nombre de commandes par client.

SELECT NCl, **COUNT**((NCmd) NbCmd
FROM Commande
GROUP BY NCl ;

…

Commande

NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

NCl	NbCmd
CL02	1
CL05	1
CL03	2
CL10	1

Bouchenf Anis

Groupement

Application 2 :

✓ Donner la quantité totale commandée par produit.

SELECT NP, **SUM**(Qte) Som
FROM Ligne_Cmd
GROUP BY NP;

… **Ligne_Cmd**

NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

NP	Som
P001	950
P002	400
P004	450
P005	70
P006	100
P007	550
P008	90

Groupement

Application 3 :

✓ Donner le nombre de produits par commande.

SELECT NCmd, **COUNT**(NP) NbProd
FROM Ligne_Cmd
GROUP BY NCmd;

… **Ligne_Cmd**

NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

NCmd	NbProd
C001	3
C002	2
C003	1
C004	4
C005	2

Groupement

Application 4 :

✓ Donner les commandes dont le nombre de produits dépasse 2.

SELECT NCmd, **COUNT**(NP) NbProd
FROM Ligne_Cmd
GROUP BY NCmd
HAVING COUNT(NP) > 2;

... **Ligne_Cmd**

NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

NCmd	NbProd
C001	3
C004	4

Groupement

Application 5 :

Produit

NP	LibP	Coul	Poids	PU	Qtes
P001	Robinet	Gris	5	18.000	1200
P002	Prise	Blanc	1.2	1.500	1000
P003	Câble	Blanc	2	25.000	1500
P004	Peinture	Blanc	25	33.000	900
P005	Poignée	Gris	3	12.000	1300
P006	Serrure	Jaune	2	47.000	1250
P007	Verrou	Gris	1.7	5.500	2000
P008	Fer	Noir	50	90.000	800

Ligne_Cmd

NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

✓ Donner le total des montants par commande.

SELECT NCmd, **SUM**(PU*Qte) TotMontant
FROM Produit P, Ligne_Cmd L
WHERE L.NP = P.NP
GROUP BY NCmd;

NCmd	TotMontant
C001	19100
C002	3325
C003	900
C004	14040
C005	11850

Tri

Les lignes constituant le résultat d'un SELECT sont obtenues dans un ordre quelconque. La clause **ORDER BY** précise l'ordre dans lequel la liste des lignes sélectionnées sera donnée.

Syntaxe :

SELECT Attr1, Attr2,…, Attrn

FROM Nom_Table1, Nom_Table2, …

WHERE Liste_Condition

ORDER BY Attr1 [**ASC**| **DESC**], Attr2 [**ASC**| **DESC**], …;

➢ **NB** : L'ordre de tri par défaut est croissant (ASC).

Tri

Application 1 :

✓ Donner les noms des clients suivant l'ordre décroissant.

SELECT NomCl
FROM Client
ORDER BY NomCl **DESC** ;

Client

NCl	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUDURE	Tunis
CL10	MELEC	Monastir
CL11	MBATIM	
CL12	BATFER	Tunis

NomCl
SOUDURE
SBATIM
SANITAIRE
PRODELEC
MELEC
MBATIM
GLOULOU
ELECTRON
BATIMENT
BATFER
BATAM
AMS

Tri

Application 2 :

✓ Donner le nombre de produits et la quantité totale par commande suivant l'ordre décroissant du nombre de produits et l'ordre croissant de la quantité totale.

```
SELECT NCmd,
COUNT DISTINCT(NP) NbProd,
SUM(Qte) SomQte
FROM Ligne_Cmd
GROUP BY NCmd
ORDER BY NbProd DESC,
         SomQte ASC ;
```

... Ligne_Cmd

NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

NCmd	NbProd	SomQte
C004	4	410
C001	3	650
C005	2	750
C002	2	750
C003	1	50

Bordeaux Arts

Sous requêtes

Consiste à effectuer une requête à l'intérieur d'une autre, ou en d'autres termes d'utiliser une requête afin d'en réaliser une autre (requêtes en cascade).

Une sous-requête doit être placée à la suite d'une clause WHERE ou HAVING, et doit remplacer une constante ou un groupe de constantes qui permettraient d'exprimer la qualification.

lorsque la sous-requête remplace une constante utilisée avec des opérateurs classiques, elle doit obligatoirement renvoyer une seule réponse (une table d'une ligne et une colonne). Par exemple :

SELECT … FROM …

Bordeaux Arts

Bordeaux Arts

Sous requêtes

lorsque la sous-requête remplace une constante utilisée dans une expression mettant en jeu les opérateurs IN, ALL ou ANY, elle doit obligatoirement renvoyer une seule colonne.

SELECT … FROM …

WHERE … IN (SELECT … FROM …) ;

lorsque la sous-requête remplace une constante utilisée dans une expression mettant en jeu l'opérateur EXISTS, elle peut renvoyer une table de n colonnes et m lignes.

Sous requêtes

Application 1 :

✓ Donner les produits dont les prix unitaires dépassent la moyenne des prix.

SELECT *
FROM Produit
WHERE PU > (**SELECT AVG**(PU) **FROM** Produit);

Le résultat de la sous-requête :

AVG(PU)
29

… **Produit**

NP	LibP	Coul	Poids	PU	Qtes
P001	Robinet	Gris	5	18.000	1200
P002	Prise	Blanc	1.2	1.500	1000
P003	Câble	Blanc	2	25.000	1500
P004	Peinture	Blanc	25	33.000	900
P005	Poignée	Gris	3	12.000	1300
P006	Serrure	Jaune	2	47.000	1250
P007	Verrou	Gris	1.7	5.500	2000
P008	Fer	Noir	50	90.000	800

NP	LibP	Coul	Poids	PU	Qtes
P004	Peinture	Blanc	25	33.000	900
P006	Serrure	Jaune	2	47.000	1250
P008	Fer	Noir	50	90.000	800

Sous requêtes

Application 2 :

✓ Donner les commandes qui ont une date inférieure à chacune des commandes du client 'CL03'.

```
SELECT *
FROM Commande
WHERE DateCmd < ALL
        (SELECT DateCmd
         FROM Commande
         WHERE NCl = 'CL03');
```

Commande

NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

Le résultat de la sous-requête :

DateCmd
15/01/2004
11/03/2004

NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C004	03/09/2003	CL10

Boudent/Adic

Sous requêtes

Application 3 :

✓ Donner les commandes qui ont une date inférieure à au moins une des commandes du client 'CL03'.

```
SELECT *
FROM Commande
WHERE DateCmd < ANY
        (SELECT DateCmd
         FROM Commande
         WHERE NCl = 'CL03');
```

Commande

NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

Le résultat de la sous-requête :

DateCmd
15/01/2004
11/03/2004

NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10

Boudent/Adic

Sous requêtes

Application 4 :

✓ Donner les clients qui ont passé au moins une commande.

```
SELECT NomCl
FROM Client Cl
WHERE EXIST (SELECT * FROM Commande C WHERE Cl.NCl =C.NCl);
```

Client

NCl	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUDURE	Tunis
CL10	MELEC	Monastir
CL11	MBATIM	
CL12	BATFER	Tunis

Commande

NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

NomCl
BATIMENT
AMS
PRODELEC
MELEC

Boudent/Adic

Sous requêtes

Application 5 :

✓ Donner des clients qui n'ont passé aucune commande.

```
SELECT NomCl
FROM Client Cl
WHERE NOT EXIST (SELECT * FROM Commande C WHERE Cl.NCl =C.NCl);
```

Client

NCl	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUDURE	Tunis
CL10	MELEC	Monastir
CL11	MBATIM	
CL12	BATFER	Tunis

Commande

NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

NomCl
BATAM
GLOULOU
ELECTRON
SBATIM
SANITAIRE
SOUDURE
MBATIM
BATFER

Boudent/Adic

Opérateurs ensemblistes : **UNION**

L'opérateur **UNION** permet de fusionner deux sélections de tables pour obtenir un ensemble de lignes égal à la réunion des lignes des deux sélections. Les lignes communes n'apparaîtront qu'une seule fois.

Syntaxe :

Requête1

UNION

Requête2 ;

- Requête1 et Requête2 doivent avoir la même structure.
- Par défaut les doublons sont automatiquement éliminés.
- Pour conserver les doublons, il est possible d'utiliser une clause UNION ALL.

Bouchaud Anis

Opérateurs ensemblistes : **UNION**

Application :

✓ Donner l'ensemble des clients de Tunis et de Sousse qui ont des commandes.

```
SELECT Cl.NCl, NomCl, AdrCl
FROM Client Cl, Commande C
WHERE Cl.NCl=C.NCl
      AND J.AdrCl='Tunis'
UNION
SELECT Cl.NCl, NomCl, AdrCl
FROM Client Cl, Commande C
WHERE Cl.NCl=C.NCl
      AND J.AdrCl='Sousse';
```

Client

NCl	NomCl	AdrCl
CL01	BATAM	Tunis
CL02	BATIMENT	Tunis
CL03	AMS	Sousse
CL04	GLOULOU	Sousse
CL05	PRODELEC	Tunis
CL06	ELECTRON	Sousse
CL07	SBATIM	Sousse
CL08	SANITAIRE	Tunis
CL09	SOUDURE	Tunis
CL10	MELEC	Monastir
CL11	MBATIM	
CL12	BATFER	Tunis

Commande

NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

R1

NCl	NomCl	AdrCl
CL02	BATIMENT	Tunis
CL05	PRODELEC	Tunis

R2

NCl	NomCl	AdrCl
CL03	AMS	Sousse

NCl	NomCl	AdrCl
CL02	BATIMENT	Tunis
CL05	PRODELEC	Tunis
CL03	AMS	Sousse

Bouchaud Anis

Bouchaud Anis

Opérateurs ensemblistes : **INTERSECTION**

L'opérateur **INTERSECT** permet d'obtenir l'ensemble des lignes communes à deux requêtes.

Syntaxe :

Requête1

INTERSECT

Requête2 ;

- Requête1 et Requête2 doivent avoir la même structure.
- Il est possible de remplacer l'opérateur INTERSECT par des commandes usuelles :

```
SELECT a, b

FROM table1

WHERE EXISTS ( SELECT c, d FROM table2 WHERE a=c AND b=d) ;
```


Opérateurs ensemblistes : **INTERSECTION**

Application :

✓ Donner l'ensemble des produits communs aux commandes C001 et C005.

```
SELECT P.NP, LibP
FROM Produit P, Ligne_Cmd L
WHERE P.NP=L.NP
      AND NCmd='C001'
INTERSECT
SELECT P.NP, LibP
FROM Produit P, Ligne_Cmd L
WHERE P.NP=L.NP
      AND NCmd='C005' ;
```

Produit

NP	LibP	Coul	Poids	PU	Qtes
P001	Robinet	Gris	5	18.000	1200
P002	Prise	Blanc	1.2	1.500	1000
P003	Câble	Blanc	2	25.000	1500
P004	Peinture	Blanc	25	33.000	900
P005	Poignée	Gris	3	12.000	1300
P006	Serrure	Jaune	2	47.000	1250
P007	Verrou	Gris	1.7	5.500	2000
P008	Fer	Noir	50	90.000	800

Ligne_Cmd

NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

R1

NP	LibP
P001	Robinet
P004	Peinture
P006	Serrure

R2

NP	LibP
P001	Robinet
P002	Prise

NP	LibP
P001	Robinet

Opérateurs ensemblistes : **Différence**

L'opérateur **MINUS** permet d'obtenir les lignes de la première requête et qui ne figurent pas dans la deuxième.

Syntaxe :

Requête1

MINUS (ou EXCEPT)

Requête2 ;

- Requête1 et Requête2 doivent avoir la même structure.

Opérateurs ensemblistes : **DIFFERENCE**

Application :

✓ Donner l'ensemble des produits qui n'ont pas été commandés.

```
SELECT NP, LibP
FROM Produit
MINUS
SELECT P.NP, LibP
FROM Produit P, Ligne_Cmd L
WHERE P.NP=L.NP;
```

Produit

NP	LibP	Coul	Poids	PU	Qtes
P001	Robinet	Gris	5	18.000	1200
P002	Prise	Blanc	1.2	1.500	1000
P003	Câble	Blanc	2	25.000	1500
P004	Peinture	Blanc	25	33.000	900
P005	Poignée	Gris	3	12.000	1300
P006	Serrure	Jaune	2	47.000	1250
P007	Verrou	Gris	1.7	5.500	2000
P008	Fer	Noir	50	90.000	800

Ligne_Cmd

NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

Tous les produits

Les produits commandés

R1

NP	LibP
P001	Robinet
P002	Prise
P003	Câble
P004	Peinture
P005	Poignée
P006	Serrure
P007	Verrou
P008	Fer

R2

NP	LibP
P001	Robinet
P002	Prise
P004	Peinture
P005	Poignée
P006	Serrure
P007	Verrou
P008	Fer

NP	LibP
P003	Câble

Langage de Contrôle de Données

Plusieurs personnes peuvent travailler simultanément sur une base de données, toutefois ces personnes n'ont pas forcément les mêmes besoins : certaines peuvent par exemple nécessiter de modifier des données dans la table, tandis que les autres ne l'utiliseront que pour la consulter. Ainsi, il est possible de définir des permissions pour chaque personne en leur octroyant un mot de passe.

Cette tâche incombe à l'administrateur de la base de données (en anglais DBA, DataBase Administrator). Il doit dans un premier temps définir les besoins de chacun, puis les appliquer à la base de donnée sous forme de permissions.

156

Langage de Contrôle de Données

Le langage SQL permet d'effectuer ces opérations grâce à deux clauses :

GRANT permet d'accorder des droits à un (parfois plusieurs sur certains SGBD) utilisateur ·

REVOKE permet de retirer des droits à un (ou plusieurs sur certains SGBD) utilisateur Les permissions (appelées aussi droits ou privilèges) peuvent être définies pour chaque (un grand nombre) clause.

D'autre part il est aussi possible de définir des rôles c'est-à-dire de permettre à d'autres utilisateurs d'accorder des permissions.

Langage de Contrôle de Données

GRANT : Permet au propriétaire d'une table ou vue de donner à d'autres utilisateurs des droits d'accès à celles ci.

Syntaxe :

GRANT Liste_Privilège **ON** Table/ Vue **TO** Utilisateur *[WITH GRANT OPTION];*

Les privilèges sont :

SELECT	Droit de lecture
INSERT	Droit d'insertion de lignes
UPDATE	Droit de modification de lignes
UPDATE (Attr1, Attr2, …)	Droit de modification de lignes à certains attributs
DELETE	Droit de suppression de lignes

GRANT SELECT ON Produit TO User1;

Langage de Contrôle de Données

REVOKE : Un utilisateur ayant accordé un privilège peut l'annuler à l'aide de la commande REVOKE.

Syntaxe :

REVOKE Liste_Privilège **ON** Table/Vue **FROM** Utilisateur;

Application :

REVOKE SELECT ON Produit FROM User1;

NB : Si on enlève un privilège à un utilisateur, ce même privilège est