

**INSTITUT D'ENSEIGNEMENT TECHNIQUE
DE MECANIQUE ET D'ELECTRICITE
MARGUERITE MASSART**

**BASES
DE
DONNEES**

Professeur : BANZIRA Pascal

1^{ère} PARTIE :

Introduction à MySQL / PhPMyAdmin

Introduction

- MySQL dérive directement de SQL (Structured Query Language)
- L'outil phpMyAdmin est développé en PHP et offre une interface pour l'administration des bases de données
- phpMyAdmin est téléchargeable ici :
<http://phpmyadmin.sourceforge.net>
- cet outil permet de :
 - créer de nouvelles bases
 - créer/modifier/supprimer des tables
 - afficher/ajouter/modifier/supprimer des tuples dans des tables
 - effectuer des sauvegardes de la structure et/ou des données
 - effectuer des requêtes
 - gérer les priviléges des utilisateurs

Liens intéressants (MySQL)

- La référence MySQL (anglais): <http://www.mysql.org>
- Le manuel MySQL :
<http://dev.mysql.com/doc/refman/5.7/en>
- Des cours et articles intéressants :
 - <http://www.developpez.com>
 - <http://cyberzoide.developpez.com/php4/mysql/>

Types des attributs MySQL

- Nombre entier signé ou non
- Nombre à virgule
- Chaîne de caractères
- Date et heure
- Enumeration
- Ensemble

Entiers

nom	borne inférieure	borne supérieure
TINYINT	-128	127
TINYINT UNSIGNED	0	255
SMALLINT	-32768	32767
SMALLINT UNSIGNED	0	65535
MEDIUMINT	-8388608	8388607
MEDIUMINT UNSIGNED	0	16777215
INT*	-2147483648	2147483647
INT* UNSIGNED	0	4294967295
BIGINT	-9223372036854775808	9223372036854775807
BIGINT UNSIGNED	0	18446744073709551615

(*) : INTEGER est un synonyme de INT.

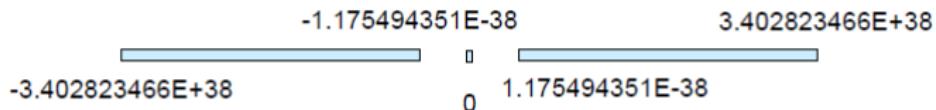
UNSIGNED permet d'avoir un type non signé.

ZEROFILL : remplissage des zéros non significatifs.

Flottants

Les flottants – dits aussi nombres réels – sont des nombres à virgule. Contrairement aux entiers, leur domaine n'est pas continu du fait de l'impossibilité de les représenter avec une précision absolue.

Exemple du type **FLOAT** :



nom	domaine négatif : borne inférieure borne supérieure	Domaine positif : borne inférieure borne supérieure
FLOAT	-3.402823466E+38 -1.175494351E-38	1.175494351E-38 3.402823466E+38
DOUBLE*	-1.7976931348623157E+308 -2.2250738585072014E-308	2.2250738585072014E-308 1.7976931348623157E+308

(*) : **REAL** est un synonyme de **DOUBLE**.

Chaînes

nom	longueur
CHAR(M)	Chaîne de taille fixée à M, où $1 < M < 255$, complétée avec des espaces si nécessaire.
CHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
VARCHAR(M)	Chaîne de taille variable, de taille maximum M, où $1 < M < 255$, complété avec des espaces si nécessaire.
VARCHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
TINYTEXT	Longueur maximale de 255 caractères.
TEXT	Longueur maximale de 65535 caractères.
MEDIUMTEXT	Longueur maximale de 16777215 caractères.
LONGTEXT	Longueur maximale de 4294967295 caractères.
DECIMAL(M,D)*	Simule un nombre flottant de D chiffres après la virgule et de M chiffres au maximum. Chaque chiffre ainsi que la virgule et le signe moins (pas le plus) occupe un caractère.

(*) : **NUMERIC** est un synonyme de **DECIMAL**.

Dates et heures

nom	description
DATE	Date au format anglophone AAAA-MM-JJ.
DATETIME	Date et heure au format anglophone AAAA-MM-JJ HH:MM:SS.
TIMESTAMP	Affiche la date et l'heure sans séparateur : AAAAMMJJHHMMSS.
TIMESTAMP(M)	Idem mais M vaut un entier pair entre 2 et 14. Affiche les M premiers caractères de TIMESTAMP .
TIME	Heure au format HH:MM:SS.
YEAR	Année au format AAAA.

nom	description
TIMESTAMP(2)	AA
TIMESTAMP(4)	AAMM
TIMESTAMP(6)	AAMMJJ
TIMESTAMP(8)	AAAAMMJJ
...	...

Si attribut de type **TIMESTAMP**

= *NULL* :

- prend la date et heure de l'insertion.
- suit le format indiqué

Interface graphique (phpMyAdmin)

- développé en PHP
- téléchargeable :
<http://phpmyadmin.sourceforge.net>
- <http://euterpe.unice.fr/phpmyadmin>
- version installée (5.x)

Interface graphique (phpMyAdmin) - 1

The screenshot shows the phpMyAdmin interface running in Mozilla Firefox. The title bar reads "euterpe.unice.fr / localhost | phpMyAdmin 2.6.2-Dbian-3sarge1 - Mozilla Firefox". The menu bar includes "Fichier", "Edition", "Affichage", "Aide à", "Marque-pages", and "Outils ?". The toolbar has icons for back, forward, search, and other functions. The address bar shows the URL "http://euterpe.unice.fr/phpmyadmin/index.php?lang=fr-utf8&server=1&collation_ci=...". Below the address bar, there are links to "Ma page WEB", "pagesjaunes.fr", "demande...", "Bienvenue sur Free", "Bureau virtuel", and "Annuaire de l'université".

The main content area is titled "Bienvenue à phpMyAdmin 2.6.2-Dbian-3sarge1" and shows "MySQL 4.1.11-Dbian_4sarge4-log sur le serveur localhost - utilisateur : lahire@localhost".

The interface is organized into three main sections:

- Accès** (Access): Located on the left, it contains a sidebar with "Base de données: (Bases de données) ...". A green callout box points to this sidebar.
- Création** (Creation): Located in the center, it contains a "MySQL" section with various options like "Créer une base de données", "Interclassement", and "Ordre". A green callout box points to this section.
- langue** (Language): Located on the right, it contains settings for "Language: French (fr-utf8)", "Jeu de caractères pour MySQL: UTF-8 Unicode (utf8)", "Interclassement pour la connection MySQL: utf8_general_ci", "Thème / Style: Original", and links to "Documentation de phpMyAdmin", "Afficher les informations relatives à PHP", "Site officiel de phpMyAdmin", "ChangeLog", "CVS", and "Lists". A green callout box points to this section.

At the bottom, the URL "http://euterpe.unice.fr/phpmyadmin/changelog.php" is visible, along with the Firefox status bar showing navigation icons.

Interface graphique (phpMyAdmin) - 2

The screenshot shows the phpMyAdmin interface in Mozilla Firefox. The URL is `http://euterpe.unice.fr/phpmyadmin/index.php?lang=fr-utf-8&server=1&collation=utf8_general_ci`. The main menu bar includes 'Fichier', 'Edition', 'Affichage', 'Aller à', 'Marque-pages', 'Outils'. The toolbar includes icons for back, forward, search, and refresh.

Autres bases (Other databases) is highlighted in a green box on the left sidebar, pointing to the 'Iahire06-07' database.

Requête SQL (SQL Query) is highlighted in a green box at the top right, pointing to the SQL query input field which contains `CREATE DATABASE 'Iahire06-07'`.

Création de table (Table Creation) is highlighted in a green box at the bottom right, pointing to the 'Nom:' field containing 'client' and the 'Champs:' field containing '4'.

The main content area shows:

- Base de données Iahire06-07 a été créé(e).
- réquête SQL:
`CREATE DATABASE 'Iahire06-07'`
- Aucune table n'a été trouvée dans cette base.
- Créer une nouvelle table sur la base Iahire06-07:
Nom:
Champs:

Interface graphique (phpMyAdmin) - 3

Ecran de gestion d'une table Client dans phpMyAdmin sur MySQL 5.0.45-0ubuntu0.4.1-0.1.1. L'écran montre les colonnes de la table avec leurs types et propriétés.

Champ	Type	Taille/Valeurs*	Interclassement	Attributs	Null	Defaut**	Extra
	VARCHAR				<input checked="" type="checkbox"/>	not null	
	VARCHAR				<input checked="" type="checkbox"/>	not null	
	VARCHAR				<input checked="" type="checkbox"/>	not null	
	VARCHAR				<input checked="" type="checkbox"/>	not null	

Commentaires sur la table:

Type de la table: MyISAM Interclassement:

InnoDB Signé?

Auto?

Ajouter 1 champ(s) Exécuter Sauvegarder

* Les différences valeurs des champs de type enum/set sont à spécifier sous la forme 's1,s2,s3...' . Pour utiliser un caractère '*' ou '** dans l'une de ces valeurs, faites le précédé du caractère d'échappement '\' (par exemple 'Noays'\ ou 'ib'b').

** Pour les valeurs par défaut, veuillez n'entrer qu'une seule valeur, sans caractère d'échappement ou apostrophes, sous la forme: a

4 champs Type possible (pas de domaine) Saisie obligatoire? Clé primaire? Index? Unique?

Quelques mots sur InnoDB

- Moteur de tables (licence GNU GLP)
- gestionnaire de tables transactionnelles (verrouillage de lignes)
- maximisation de performances (grands volumes de données)
- support clé étrangère
- stockage des tables et index : espace de tables (un ou plusieurs fichiers). MyISAM : un fichier par table

Interface graphique (phpMyAdmin) - 4

The screenshot shows the phpMyAdmin interface for managing a database table. A green callout box points to the primary key icon in the action column of the table structure, with the text "Raccourci Clé primaire?" (Primary key shortcut?). Another green callout box points to the "Modifier" (Modify) button at the bottom of the page, with the text "Modifier supprimer" (Modify delete).

Table Structure:

	Champ	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
<input type="checkbox"/>	numero	smallint(5)		UNSIGNED	Non		auto_increment	
<input type="checkbox"/>	nom	varchar(20)	latin1_swedish_ci		Non			
<input type="checkbox"/>	adresse	varchar(30)	latin1_swedish_ci		Non			
<input type="checkbox"/>	num-telephone	varchar(10)	latin1_swedish_ci		Oui	NULL		

Action Bar:

- Tout cocher / Tout décocher
- Pour la sélection :
-

Tablettes et Statistiques:

Index:		Espace utilisé:		Statistiques:				
Nom de la clé	Type	Cardinalité	Action	Champ	Type	Espace	Information	Valeur
PRIMARY	PRIMARY	0		numero	Données	0 Octets	Format	dynamique
nom	FULLTEXT	aucune		nom	Index	1 024 Octets	Interclassement	latin1_swedish_ci
					Total	1 024 Octets	Enregistrements	0

Buttons:

- Version imprimable
- Suggérer des optimisations quant à la structure de la table
- Ajouter 1 champ(s)
- En fin de Table
- En début de Table
- Après numero
- Exécuter

Bottom Buttons:

- Index:
- Modifier supprimer
- Terminé

Callout Boxes:

- Raccourci Clé primaire?
- Modifier supprimer

Interface graphique (phpMyAdmin) – 4 bis

The screenshot shows the phpMyAdmin interface for the 'vente' table in the 'lahire06-07' database. The 'Structure' tab is selected. A red box highlights the status bar message: "Les ventes de l'entreprise (InnoDB free: 4096 KB; (ref-client) REFER Lahire06-07/client ('numero'); (ref-produit) REFER Lahire06-07/produit ('numero')". Below the table structure, a green box contains the text "Pour clés étrangères". A green arrow points from this text to the 'Index' section of the interface. The 'Index' section shows three indexes: PRIMARY (PRIMARY), ref-produit (INDEX), and ref-client (INDEX). The 'Espace utilisé' and 'Statistiques' sections are also visible.

Pour clés
étrangères

Index pour Clés étrangères

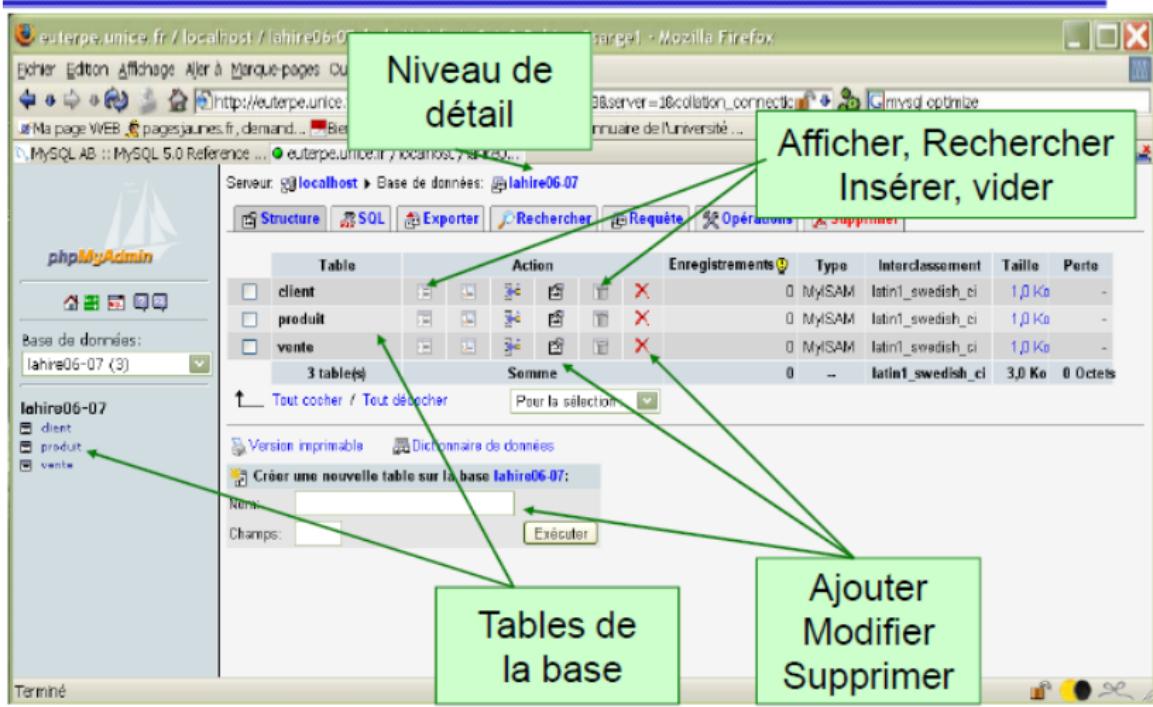
Intégrité référentielle

ON UPDATE et ON DELETE

Si l'utilisateur tente de supprimer une ligne d'une table parente, alors qu'une ou plusieurs lignes existent dans une table enfant correspondante à la première, il y a plusieurs possibilités :

- CASCADE efface la ligne de la table parente et supprime automatiquement les lignes correspondants dans la table enfante
- SET NULL supprime la ligne de la table parente et met la (ou les) valeur(s) de la clé étrangère à NULL
- SET DEFAULT supprime la ligne de la table parente et met tous les composants de la clé étrangère à leur valeur par default dans la table enfante
- NO ACTION rejette l'opération de suppression dans la table parente

Interface graphique (phpMyAdmin) - 5



Interface graphique (phpMyAdmin) - 6

The screenshot shows the phpMyAdmin interface for the database 'lahire06_07'. A green callout box labeled 'Modification Suppression' points to the 'Structure' and 'Afficher' buttons in the top menu bar. Another green callout box labeled 'Contenu' points to the list of client records. A green callout box labeled 'Paramétrage affichage' points to the 'Afficher' and 'en mode' dropdown settings. A green callout box labeled 'Equivalent en SQL' points to the SQL query editor area.

Niveau de détail

Tableau des clients

Affichage des enregistrements 0 - 1 (0 total, traitement 0.0001 sec.)

requête SQL :

```
SELECT * FROM `client`  
LIMIT 0, 30
```

[Modifier] [Expliquer SQL] [Créer source PHP] [Actualiser]

Afficher 30 ligne(s) à partir de l'enregistrement n° 0
en mode horizontal et répéter les en-têtes à chaque groupe de 100

Trier sur l'index: PRIMARY (Croissant) Exécuter

	numero	nom	adresse	num-telephone
1	Durand	Nicolas	0610465412	
2	Fabre	Pascal	Mifflin	

Tout cacher / Tout déplier Pour la sélection :

Afficher 30 ligne(s) à partir de l'enregistrement n° 0
en mode horizontal

Terminé

Interface graphique (phpMyAdmin) - 7

The screenshot shows the phpMyAdmin interface for the database 'lahire06-07'. The 'Structure' tab is selected. A red box highlights the 'Exporter' button in the top menu bar. Three green callout boxes are present: one labeled 'Niveau de détail' pointing to the 'Structure' checkbox in the 'Options SQL' section; one labeled 'Configuration' pointing to the 'Inclure sous forme de commentaires' section; and one labeled 'Exportation' pointing to the 'Type d'exportation' dropdown set to 'INSERT'.

Niveau de détail

Configuration

Exportation

Exporter

Structure

SQL

Rechercher

Opérations

Supprimer

Schéma et données de la base

Exporter

Structure:

- Inclure des énoncés "DROP TABLE"
- Ajouter "IF NOT EXISTS"
- Inclure la valeur courante de l'AUTO_INCREMENT
- Protéger les noms des tables et des champs par des ""

Inclure sous forme de commentaires

- Dates de création/modification/verification

Compatibilité de l'exportation: NONE

Données:

- Insertions complètes
- Insertions étendues
- Insertions avec délais (DELAYED)
- Ignorer les erreurs de doublons (INSERT IGNORE)
- Encoder les champs binaires en hexadécimal

Type d'exportation: INSERT

Interface graphique (phpMyAdmin) - 8

Champs

Niveau de détail

Affichage

Critère

phpMyAdmin

Base de données: Ahire06-07 (3)

Ahire06-07

client

produit

vente

Structure Afficher SQL Rechercher Insérer Exporter Opérations Vider Supprimer

Table des clients

Choisissez les champs à afficher (au moins un)

numéro nom adresse num-telephone

DISTINCT

Nombre d'enregistrements par page : 30

Ordre d'affichage :

Croissant

Décroissant

Critères de recherche (pour l'échancrure "where"):

Exécuter

Où Recherche par valeur (passerpartout "%")

Champ	Type	Interclassement	Opérateur	Valeur
numero	smallint(5)		LIKE	
nom	varchar(20) latin1_swedish_ci		LIKE	
adresse	varchar(30) latin1_swedish_ci		LIKE	
num-telephone	varchar(10) latin1_swedish_ci		LIKE	

Interface graphique (phpMyAdmin) – 8b

The screenshot shows the phpMyAdmin interface for the 'lahire06-07' database. The 'Rechercher' (Search) tab is selected. A green box labeled '1...tous' points to the 'Choisir les champs à afficher (au moins un)' (Select fields to display (at least one)) dropdown, which contains all columns: 'numero', 'nom', 'prenom', 'age', 'nb_heures', and 'ville'. A red box highlights the 'Rechercher' button. A green box labeled 'Recherche mono table' is positioned to the right of the search results table.

Choisir les champs à afficher (au moins un)

- numero
- nom
- prenom
- age
- nb_heures
- ville

DISTINCT

Nombre d'enregistrements par page : 30

Ordre d'affichage :

- Croissant
- Décroissant

Critères de recherche (pour l'énoncé "where"):

Exécuter

On Recherche par valeur (passepapout: "%")

Champ	Type	Interclassement	Opérateur	Valeur
numero	int(10)		>	1001
nom	varchar(20) latin1_swedish_ci		LIKE	Philippe
prenom	varchar(20) latin1_swedish_ci		LIKE	Philippe
age	tinyint(3)		=	
nb_heures	tinyint(3)		=	
ville	varchar(40) latin1_swedish_ci		LIKE	

Exécuter

Terminé

Interface graphique (phpMyAdmin) - 9

The screenshot shows the phpMyAdmin interface for the database 'lahire06-07'. A search query is being constructed in the 'Requête' (Query) tab:

```
SELECT `client`.`nom`,  
       `client`.`adresse`,  
       `client`.`num-telephone`  
FROM client  
WHERE (`client`.`nom` = 'prosper')
```

Annotations explain various parts of the query:

- Critère**: Points to the WHERE clause: `WHERE ('client`.`nom` = 'prosper')`.
- Niveau de détail**: Points to the 'Afficher' (Display) section where columns are selected: `'client`.`nom'`, `'client`.`adresse'`, and `'client`.`num-telephone'`.
- Champs impliqués**: Points to the 'Champ' (Field) section where the columns are listed.
- Tables**: Points to the 'Utiliser les tables:' (Use tables:) section, which lists the tables involved: `client`, `produit`, and `vente`.
- Requête SQL**: Points to the 'Requête SQL sur la base Lahire06-07:' (SQL query on base Lahire06-07) text area containing the full query.

Interface graphique (phpMyAdmin) - 10

The screenshot shows the phpMyAdmin interface for the database 'lahire06-07'. A search query is being constructed in the 'Requête' tab:

```
SELECT 'Enseignant'.nom,  
       'UE'.libelle  
FROM Enseignant, UE  
WHERE ((Enseignant.nom  
= 'Tahire') AND (UE.responsable =  
Enseignant.numero))  
((Enseignant.nom = 'menez') AND  
(UE.responsable =
```

The interface is annotated with four green boxes and red numbers:

- 1 Sélection**: Points to the search term 'Tahire' in the 'Critères' field.
- 2 Tables**: Points to the table selection area at the bottom left, showing tables like 'Enseignant', 'UE', 'Adresse', etc.
- 3 Niveau de détail**: Points to the 'Afficher' dropdown set to 'Tahire'.
- 4 Champs impliqués**: Points to the 'Champ' dropdowns for 'Enseignant.nom' and 'UE.libelle'.
- 5 Requête SQL**: Points to the generated SQL query in the 'Requête SQL sur la base lah06-07:' section.

Interface graphique (phpMyAdmin) - 11

The screenshot shows the phpMyAdmin interface for the database 'lahre06_07'. The left sidebar lists tables: Adresse, Enseignant, Etudiant, EtudiantUE, UE, client, produit, and vente. The main area displays a query result for the 'Enseignant' table. The query is:

```
REQUÊTE SQL:  
SELECT `Enseignant`.`nom`, `UE`.`libelle`  
FROM `Enseignant` LEFT  
JOIN `UE`  
ON `Enseignant`.`nom` = `UE`.`enseignant_nom`  
LIMIT 0 , 30
```

Below the query, there are buttons: [Modifier], [Expliquer SQL], [Créer source PHP], and [Actualiser]. The result table has columns 'nom' and 'libelle'. The 'nom' column is bolded. The 'libelle' column header is highlighted with a red box. A green arrow points from the 'Champs impliqués' box to the 'nom' column header. Another green arrow points from the 'Requête SQL' box to the 'libelle' column header. A green arrow points from the 'Résultat' box to the 'libelle' column header.

Niveau de détail

Champs impliqués

Requête SQL

Résultat

Interface graphique (phpMyAdmin) - 12

The screenshot shows the phpMyAdmin interface with several annotations:

- Niveau de détail**: A green box at the top right of the browser window.
- Affichage**: A green box pointing to the results table area where a row for "Prosper Paris" is displayed.
- Requête SQL**: A green box pointing to the SQL query editor area containing the following code:

```
SELECT `client`.`nom`, `client`.`adresse`, `client`.`num-telephone`
FROM `CLIENT`
WHERE (
    `client`.`nom` = 'prospe'
```
- Résultat**: A green box pointing to the bottom section of the interface which includes buttons for inserting new records, printing versions, exporting, and a summary message "Terminé".

The browser address bar shows the URL: http://euterpe.unice.fr/phpmyadmin/index.php?lang=fr-utf8&server=1&collation_connection=utf8.

The phpMyAdmin interface itself has the following visible elements:

- Server: localhost
- Base de données: Lahire06-07 (3)
- Tables: client, produit, vente
- SQL tab selected
- Structure, Exporter, Rechercher, Opérations, Supprimer tabs available
- SQL query editor with the above SQL code
- Results table showing one row: nom, adresse, num-telephone
- Bottom navigation: Insérer un nouvel enregistrement, Version imprimable, Version imprimable (avec textes complets), Exporter, Terminé

Interface graphique (phpMyAdmin) - 13

Niveau de détail

Champs

insertion

Valeurs pour le tuple

Attribut calculé

phpMyAdmin

Base de données: Lahire05-07 (3)

lahire05-07

- client
- produit
- vente

Champ	Type	Fonction	Null	Valeur
numero	smallint(5) unsigned			
nom	varchar(20)	ASCII CHAR MD5 SHA1 ENCRYPT RAND		
adresse	varchar(30)			
num-telephone	varchar(10)			

Ignorer

Champ	Type	Fonction	Null	Valeur
numero	smallint(5) unsigned	LAST_INSERT_ID UNIX_TIMESTAMP COUNT AVG SUM		
nom	varchar(20)			
adresse	varchar(30)	SOUNDEX LCASE UCASE NOW PASSWORD CURDATE CURTIME		
num-telephone	varchar(10)			

Insérer en tant que nouvel enregistrement

Revenir à la page précédente

Inserer un nouvel enregistrement

Exécuter Rémuniser les val...

Terminé

Résumé sur le modèle relationnel (1)

- Le système de gestion de base de données relationnelle est actuellement le logiciel de traitement de données le plus fréquemment utilisé
- Une **relation** mathématique est un sous-ensemble du produit cartésien de deux ensembles ou plus. En termes de base de données, une relation est n'importe quel sous-ensemble du produit cartésien des domaines des attributs
- Les relations sont représentées de manière physique par des **tables**, dont les lignes correspondent aux tuples individuels et les colonnes aux attributs.

Résumé sur le modèle relationnel (2)

- Les propriétés d'une base de données sont les suivantes :
 - chaque cellule contient exactement une valeur atomique
 - les noms d'attributs sont distincts les uns des autres
 - l'ordre des attributs est immatériel
 - l'ordre des tuples est immatériel
 - il n'existe pas de tuples en double
- Dans un tuple, un **nul** représente une valeur d'un attribut, inconnue à l'heure actuelle ou qui ne s'applique pas à ce tuple

Résumé sur le modèle relationnel (3)

- Une **clé candidate** est un ensemble minimum d'attributs qu'identifie les tuples d'une relation de façon unique
- Une **clé primaire** est la clé candidate choisie pour servir à l'identification de tuples
- Une **clé étrangère** est un ensemble d'attributs au sein d'une relation qui constitue une clé candidate d'une autre relation
- **l'intégrité d'entité** établit que, dans une relation de base, aucun attribut qui fait partie de la clé primaire ne peut être nul
- **l'intégrité référentielle** établit que les valeurs d'une clé étrangère doivent correspondre à une valeur d'une clé candidate d'un tuple dans la relation de référence de la clé candidate ou être complètement nulles

2ème PARTIE :

Le langage SQL

Introduction

Objectifs de SQL

- Créer la structure de la base de données et de ses tables
- Exécuter les tâches de base de la gestion des données, telle que l'insertion, la modification et la suppression de données des tables
- Effectuer des requêtes simples ou complexes

Langage orienté transformation

Origine

Structured Query Language

- caractéristiques des langages déclaratifs
- origine : IBM, System R, milieu des années 70
- implémenté dans de nombreux SGBD

Plusieurs versions :

- SQL1 initial : ANSI* —1986
- SQL1 avec intégrité référentielle, ANSI —1989
- SQL2 ANSI —1992
- SQL3 ANSI — 1999 incorpore la notion d'objet

* ANSI = *American National Standard Institute*

Caractéristiques de SQL

- Fonctionnalités :
 - Définition des objets de la base de données (LDD)
 - Manipulation de données (LMD)
 - Contrôle des accès aux données
 - Gestion de transactions
- Utilisé par : DBA, développeurs, quelques utilisateurs



Parallèle avec mySQL

Principales Instructions

- Définitions (LDD)

CREATE, DROP, ALTER

- Mises à jour (LMD)

INSERT, UPDATE, DELETE

- Interrogations (LMD)

SELECT



- Contrôle d'accès aux données

GRANT, REVOKE

- Gestion de transactions

COMMIT, ROLLBACK

Consultation des données

- Hypothèse:
 - un schéma de base de données est créé
 - Une base de données a été remplie
- La création a été faite par une interface QBE (mySQL)
- On expérimente la consultation avec SQL
- On expérimentera la création du schéma et le remplissage de la base avec SQL plus tard

- ➡ Langage algébrique en SQL
- ➡ Requêtes mono et multi table(s)

Exemple pour les requêtes

CLIENT

numéro	nom	adresse	numéro_téléphone
101	Durand	NICE	0493456743
108	Fabre	PARIS	NULL
110	Prosper	PARIS	NULL
125	Antonin	MARSEILLE	NULL

PRODUIT

référence	marque	Prix HT
153	BMW	1000
589	PEUGEOT	1800
158	TOYOTA	1500

VENTE

numéro	référence_produit	numéro_client	date
00102	153	101	12/10/04
00809	589	108	20/01/05
11005	158	108	15/03/05
12005	589	125	30/03/05

Format des requêtes

- SELECT
- FROM
- WHERE
- GROUP BY
- HAVING
- ORDER BY
- FROM spécifie la table ou les tables à utiliser
- WHERE filtre les lignes selon une condition donnée
- GROUP BY forme des groupes de lignes de même valeur de colonne
- HAVING filtre les groupes sujets à une certaine condition
- SELECT spécifie les colonnes qui doivent apparaître dans les résultats
- ORDER BY spécifie l'ordre d'apparition des données dans le résultat

Requêtes simples (SELECT-FROM)

- Afficher le nom et l'adresse des clients

```
SELECT nom , adresse
```

```
FROM Client ;
```

- Afficher toutes les informations des clients

```
SELECT *
```

```
FROM Client ;
```

Elimination des doublons

- Afficher l'adresse de tous les clients

```
SELECT adresse  
FROM Client ;
```

```
SELECT ALL adresse  
FROM Client;
```

- Afficher toutes les adresses existantes (sans doublons)

```
SELECT DISTINCT adresse  
FROM Client ;
```

Par défaut

Projection

→ { Relationnel : On doit pouvoir distinguer chaque tuple !
SQL : le résultat peut ne pas être une relation

Sélection de colonne (clause WHERE)

Les conditions fondamentales de recherche

- *comparaison*
(salaire > 10000, ville = 'Paris')
- *étendue ou intervalle*
(salaire BETWEEN 20000 and 30000)
- *appartenance à un ensemble*
(couleur IN ('red', 'vert'))
- *correspondance à un masque*
(adresse LIKE '%Montréal%')
- *nul*
(adresse IS NULL)

Opérateur Sélection

- Quels sont les clients dont l'adresse est *Paris*

```
SELECT *
FROM Client
WHERE adresse = 'Paris';
```

- Quels sont les produits dont le prix TTC est supérieur à 1000

```
SELECT *
FROM Produit
WHERE prix_HT + prix_HT * 0.195 > 1000;
```

Opérations possibles (mySQL)

- Booléennes

`and, or, xor, =, !=, <, >, <=, >=`

- Arithmétiques

`+, -, *, /`

`+, -` *opérateurs unaires*

- Fonctions numériques

`abs, log, cos, sin, mod, power ...`

- Arithmétiques sur date

`+, -`

Un grand nombre

- Fonctions sur chaînes

`length, concat, ...`



{ Dans **Select**: attribut calculé (résultat)
Dans **Where**: participer à la sélection

Précérence des opérateurs

- := + faible
- ||, OR, XOR
- &&, AND
- BETWEEN, CASE, WHEN, THEN, ELSE
- =, <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN
- |
- &
- <<, >>
- -, +
- *, /, DIV, %, MOD
- ^
- - (unary minus), ~ (unary bit inversion)
- !, NOT
- BINARY, COLLATE + forte

SELECT 1+2*3;

7

+ forte

Utilisation des opérateurs

- **SELECT ABS** (-32); 32 *Bien sur on peut utiliser des attributs*
- **SELECT FLOOR** (1.23); 1
- **SELECT MOD** (234, 10); 4 *Peut se trouver dans Where*
- **SELECT 253 % 7;** 1 **WHERE** 3 / 5 < 1;
- **SELECT ROUND** (1.298, 1); 1,3
- **SELECT ROUND** (1.298, 0); 1
- **SELECT SIGN** (234) **SIGN** (-32) **SIGN** (0); 1 / -1 / 0
- **SELECT 3 / 5;** 0,60

Utilisation des opérateurs (Chaînes)

- **SELECT CONCAT ('My', 'S', 'QL');** 'MySQL'
- **SELECT CHAR_LENGTH ('MySQL');** 5
- **SELECT LOCATE ('bar', 'foobarbar');** 4
- **SELECT LOCATE ('bar', 'foobarbar', 5);** 7
- **SELECT INSERT ('Quadratic', 3, 4, 'What');** 'QuWhattic'
- **SELECT LOWER ('MySQL');** 'mysql'
- **SELECT SUBSTRING ('Quadratically',5,6);** 'ratica'
- **SELECT 'David!' LIKE 'David_';** 1
- **SELECT 'David!' LIKE '%D%D%v%';** 1
- **SELECT STRCMP (S1, S2);** -1,0,1

_ et %
0 (false), 1 (true)

Combinaison Sélection + projection

SELECT liste d'attributs ou *

Attributs recherchés / calculés

FROM liste des relations

WHERE condition ;

Si plusieurs: il faut une jointure

Test sur chaque tuple: résultat vrai ou faux

Utilise des fonctions, opérateurs...

Autres clauses possibles

Requêtes simples (1)

- Y a-t-il des produits dont le nom est « XBOX »

```
SELECT *
FROM Produit
WHERE nom = 'XBOX' ;
```

- Quels sont les ventes réalisés il y a plus de 30 jours?

```
SELECT *
FROM Vente
WHERE CURRENT_DATE () > 30 + date ;
```

- Quels sont les ventes faites après le 1er janvier 2007?

```
SELECT *
FROM Vente
WHERE date > DATE ('2007-01-01') ;
```

Requêtes simples (2)

- Quels sont les ventes dont le montant HT est entre 1000 et 3000 euros et dont le client n'est pas le numéro 101?

SELECT *

FROM Vente

WHERE (*prix_ht between 1000 and 3000*) and
(*numero != 101*);

- Quels sont les clients dont le nom est soit Prosper, soit Durand, soit Anthonin ?

SELECT *

FROM Client

WHERE *nom in ('Prosper', 'Durand', 'Anthonin')* ;

Requêtes simples (3)

- Quels sont les clients dont le nom commence par 'P'

SELECT *

FROM Employe

WHERE nom LIKE 'P%' ;

- Quels sont les clients dont le nom commence par 'P' et a un 'S' comme 4^{ème} lettre

SELECT *

FROM Client

WHERE nom LIKE 'P__S%' ;

Requêtes et valeurs nulles

- Quels sont les ventes dont la date de réalisation est inconnue?

SELECT *

FROM Vente

Ou : IS NOT NULL

WHERE date is null ;

Attention :

WHERE date = NULL



Whatever comparé avec NULL
→ Ni vrai ni faux

COUNT, MIN, SUM



Ignore les valeurs NULL
→ sauf COUNT (*)

Toute opération appliquée à NULL donne pour résultat NULL

Les autres clauses (tri)

```
SELECT attribut1 ...  
FROM ...  
WHERE expression  
ORDER BY attribut1 [ASC] [DESC] ...
```

Affichage!

- Donner le numero, le prix HT et la marque des produits selon l'ordre décroissant des marques et l'ordre croissant des prix HT

```
SELECT marque, prix_ht, numero
```

```
FROM Produit
```

```
ORDER BY marque DESC, prix_ht ASC;
```

Ordre d'affichage!

Mais aussi:

```
ORDER BY 1 DESC, 2 ASC;
```

Requêtes multi-tables (Opérateur Jointure)

Deux points de vue:

- exécuter des **boucles imbriquées** (une table par boucle)
appliquer la clause WHERE dans les boucles
- calculer le **produit cartésien** (une nouvelle table)
Appliquer la clause WHERE sur chaque ligne

Donner le nom d'un produit et le montant de la vente

```
SELECT Produit.nom, Vente.prix_HT  
FROM Produit , Vente  
WHERE Vente.reference_produit = Produit.numero ;  
          Critères de jointure
```

Il faut joindre les tables (jointure)

Définition de la jointure

```
SELECT Client.nom, Vente.prix_ht  
FROM Client, Vente ;
```

→ tous les tuples (*nom, prix_ht*)

→ nom est un nom de client et prix_ht est un prix de vente

Pas de critère de jointure → Produit cartésien

Intérêt?



Mots-clés pour exprimer le critère de sélection

Jointure (compléments)

- Donner la marque des produits dont le prix HT est supérieur à celui d'une BMW

SELECT Produit.marque Renommage
FROM Produit Produit_reference, Produit
WHERE Produit_reference.marque = 'BMW' **AND**
Produit.prix_HT > Produit_reference.prix_HT ;

Ou de même prix HT:

SELECT Produit.marque
FROM Produit Produit_reference, Produit
WHERE Produit_reference.marque = 'BMW' **AND**
Produit.prix_ht = Produit_reference.prix_ht **AND**
Produit.marque != Produit_reference.marque ;

Quelques jointures (compléments)

Que deviennent les tuples non sélectionnés de R1 ou R2?

- dans la jointure (interne **INNER JOIN**) les tuples qui ne peuvent pas être joints sont éliminés du résultat
- dans la jointure externe (**OUTER JOIN**) les tuples qui ne peuvent pas être joints sont conservés dans le résultat

Pour les tuples de la relation de gauche (R1) et / ou de droite (R2)

Défaut

R1 **FULL OUTER JOIN** R2 : Remplit $R1.*$ et $R2.*$

R1 **LEFT OUTER JOIN** R2 : Remplit $R1.*$

R1 **RIGHT OUTER JOIN** R2 : Remplit $R2.*$

avec NULL si nécessaire.

Exercice (1)

Relations :

- Journal (code-j, titre, prix, type, périodicité)
- Dépôt (no-dépôt, nom-dépôt, adresse)
- Livraison (no-dépôt, code-j, date-liv, quantité-livrée)

Requêtes : donner...

- le prix des journaux livrés le 15/01/07 ?
- tous le nom des hebdomadaires reçus par le dépôt de Paris..
- les titre des journaux livrés à Nice.
- le nom des dépôts qui reçoivent des hebdomadaires dont la quantité livrée excède 100.

Exercice (2)

SELECT *

FROM LIVRAISON RIGHT JOIN (DEPOT, JOURNAL)
ON (LIVRAISON.no-depot = DEPOT.no-depot AND
LIVRAISON.code-j = JOURNAL.code-j)

*Toutes les lignes de DEPOT et JOURNAL seront présentes
Avec éventuellement rien pour la partie livraison*

SELECT *

FROM LIVRAISON, DEPOT, JOURNAL
WHERE LIVRAISON.no-depot = DEPOT.no-depot AND
LIVRAISON.code-j = JOURNAL.code-j)

*Seulement les lignes de DEPOT et JOURNAL qui
correspondent à une livraison*

Opérateur Union

Afficher la liste des numéros d'employé des responsables de département et des directeurs

Département

numéro-département	...	responsable

Employé

numéro-employé	...	fonction

```
SELECT responsable  
FROM Département  
UNION  
SELECT numéro-employé  
FROM Employé  
WHERE fonction = 'Directeur';
```

Opérateurs *Intersection*, *Différence*

Afficher les numéros d'employé des responsables de département
qui sont aussi des directeurs


SELECT responsable
FROM Département
INTERSECT
SELECT numéro-employé
FROM Employé
WHERE fonction = 'Directeur';

SELECT responsable
FROM Département
EXCEPT
SELECT numéro-employé
FROM Employé
WHERE fonction = 'Directeur';


Afficher les numéros d'employé des responsables de département
Sauf ceux qui sont aussi des directeurs

Fonctions d'agrégat

- SUM (*nom d'attribut*)
- COUNT(*)
- COUNT(DISTINCT *nom d'attribut*)
- MAX (*nom d'attribut*)
- MIN (*nom d'attribut*)
- AVG (*nom d'attribut*)
- AVG (DISTINCT *nom d'attribut*)



Dans les clauses :

- SELECT
- HAVING TO

Clauses Group by

SELECT attributs recherchés

FROM liste des relations

WHERE condition

GROUP BY attributs de regroupement

[**HAVING** condition sur le groupe];

SELECT COUNT (*)

FROM Produit

GROUP BY marque

SELECT AVG (prix_ht), nom

FROM Produit

GROUP BY marque

~~nom~~

SELECT COUNT (*)

FROM Produit

WHERE marque <> 'BMW'

GROUP BY marque

HAVING AVG (prix_ht) > 10.5

Utilisation des fonctions statistiques

- *Donner la moyenne des prix HT et les prix min. et max.*

SELECT AVG (*prix_ht*), MIN (*prix_ht*), MAX (*prix_ht*)

FROM *Produit*

- *Même chose mais par marque*

SELECT AVG (*prix_ht*), MIN (*prix_ht*), MAX (*prix_ht*)

FROM *Produit*

GROUP BY *marque*

- *Même chose mais par marque si la moyenne > 10,5*

SELECT AVG (*prix_ht*), MIN (*prix_ht*), MAX (*prix_ht*)

FROM *Produit*

GROUP BY *marque* **HAVING** AVG (*prix_ht*) > 10.5

Requêtes imbriquées (1)

SELECT *liste d'attributs recherchés*

FROM *liste_des_relations*

WHERE *bloc SFW dans la condition*

Intérêt: indiquer qu'un attribut doit prendre ses valeurs dans une liste de valeurs définies par un autre bloc SWF

- Itérations imbriquées
- autre façon de faire certaines formes de jointure
- Sous-requêtes indépendantes ou pas

Requêtes imbriquées (2)

Principaux connecteurs: = != > < IN EXISTS ANY ALL

EXISTS R : retourne TRUE si R n'est pas vide, FALSE sinon

$t \text{ IN } R$: retourne TRUE si t appartient à R , FALSE sinon

valeur comp ANY R : retourne TRUE si la comparaison avec **au moins un** des tuples de R renvoie TRUE

valeur comp ALL R : retourne TRUE si la comparaison avec **tous** les tuples de R renvoie TRUE

Sous-requêtes indépendantes

Quelles sont les références produit dont le prix HT est supérieur au prix HT moyen des produits ?

```
SELECT référence  
FROM Produit  
WHERE Produit.prix_ht > (SELECT AVG(P.prix_ht)  
                 FROM Produit P);
```



Le bloc SFW imbriqué peut être évalué séparément du bloc principal

Sous-requêtes indépendantes (1)

Quels sont les marques de produits vendus le 1/1/2007 ?

```
SELECT marque  
FROM Produit, Vente  
WHERE Vente.ref-produit = Produit.référence  
      AND date = 1/1/2007;
```

ou

```
SELECT Produit.marque      Jointure ou requête imbriquée  
FROM Produit  
WHERE Produit.référence IN  
      (SELECT Vente.ref-produit  
        FROM Vente  
        WHERE date = 1/1/2007);
```

Sous-requêtes indépendantes (2)

Quelles sont les marques de produits qui n'ont pas le prix HT le plus élevé ?

SELECT marque
FROM Produit < **SELECT max (P.prix_ht) ...**
WHERE Produit.prix-ht < ANY
 (**SELECT P.prix-ht**
 FROM Produit P;)

Quelles sont les marques de produits qui ont le prix HT le plus élevé ?

SELECT marque > **SELECT max (P.prix_ht) ...**
FROM Produit
WHERE Produit.prix_ht >= ALL
 (**SELECT P.prix_ht FROM Produit P;**)

Sous-requêtes corrélées (1)

Quels produits dont le prix HT est supérieur au prix HT moyen des produits de la même marque ?

```
SELECT Produit.référence  
FROM Produit  
WHERE Produit.prix-ht > (SELECT AVG(P.prix-ht)  
                 FROM Produit P  
                 WHERE Produit.marque = P.marque);)
```

Le bloc *SFW* principal et le bloc *SFW* imbriqué doivent être évalués simultanément

Sous-requêtes corrélées (2)

Quels sont les marques dont aucun produit n'a été vendu le 1/01/2007 ?

SELECT Produit.marque

FROM Produit

WHERE NOT EXISTS

(**SELECT** Vente.ref-produit

FROM Vente, Produit P

WHERE Vente.ref-produit = P.référence and

P.marque = Produit.marque and

date = 1/01/2007;)

Exercices

Donner le nom des employés qui ont le même salaire que Dupont

Employé

numéro-employé	nom	...	salaire
----------------	-----	-----	---------

```
SELECT Y.nom  
FROM Employé X , Employé Y  
WHERE  
    X.nom='Dupont'  
    and Y.salaire = X.salaire  
    and X.nom != Y.nom;
```

Remplacer la jointure par une requête imbriquée

Manipulation des structures de données

- Schéma logique / Base de données
 - CREATE/DROP SCHEMA/DATABASE ...
- Schéma de **tables** (*relations*) et de leur contenu
 - CREATE/ALTER/DROP TABLE ...
- Définition des **contraintes** qui assurent des contrôles sur l'intégrité des données
- Mais encore :
 - les index
 - les utilisateurs et les priviléges

Création/Suppression d'un schéma

CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] *db_name*

DROP DATABASE [IF EXISTS] *db_name*

- Dans la norme SQL: création de schéma
- MySQL: *schema* = *database*
- Il faut avoir les droits de le faire (administrateur?)
- Création d'un compte s'il n'existe pas
- Association des droits

*Pas de message
d'erreur si la BD
existe!*

CREATE DATABASE *lahire06-07*

GRANT ALL PRIVILEGES

ON *lahire06-07.**
TO '*lahire06-07*'@localhost
IDENTIFIED BY '*lahire06-07*';

DROP DATABASE *lahire06-07*

Attention !

Types de la norme ANSI

INTEGER	Entiers relatifs	4 octets
SMALLINT	Entiers relatifs	2 octets
BIGINT	Entiers relatifs	8 octets
FLOAT	Flottants	4 octets
DOUBLE	Flottants	8 octets
REAL	Flottants	4 ou 8 octets
NUMERIC(N,D)/ DECIMAL(N,D)	Décimaux à précision fixe	N octets

CHAR(M)	Chaînes de longueur fixe	M octets
VARCHAR(M)	Chaînes de longueur variable	au plus M octets
BIT VARYING	Chaînes d'octets	longueur de la chaîne
DATE	date(hour, mois, an)	4 octets
TIME	heure(h, mn, sh)	4 octets
DATETIME	date et heure	8 octets
YEAR	année	2 octets

Création et gestion des relations

- Structures de données (relations, attributs, tuples)

relation	→	table
attribut	→	column
tuple	→	row

- Instructions sur les relations

CREATE TABLE

créer une relation

DROP TABLE

supprimer une relation

ALTER TABLE

modifier la structure d'une relation

Création de table/relation (1)

```
CREATE TABLE Produit (
référence CHAR(4),
nom VARCHAR(20),
marque VARCHAR(15),
constructeur CHAR(20)
);
```

```
CREATE TABLE Client (
numéro INT(10),
nom VARCHAR(20),
adresse VARCHAR(30),
téléphone CHAR(10)
);
```

```
CREATE TABLE Vente (
numéro INT(10),
ref-produit INT(10),
ref-client INT(10),
date DATE
);
```

- Compléter la description
- Voir avec les types MySQL

Création de table (valeurs nulles, défaut)

- NOT NULL : l'attribut correspondant doit toujours avoir une valeur
nom **VARCHAR(20) NOT NULL**
- DEFAULT : définir la valeur d'un attribut par défaut
adresse **VARCHAR (30) DEFAULT 'Inconnue'**

```
CREATE TABLE Produit (
    référence CHAR(4) NOT NULL,
    nom VARCHAR(20) NOT NULL,
    marque VARCHAR(15),
    constructeur CHAR(20) DEFAULT 'Renault'
);
```

Création de table (clé primaire)

Identification des tuples

PRIMARY KEY (att)

PRIMARY KEY (att1, ..., attn)

→ mono attribut ou multi attributs

→ Attributs figurant dans une clé : déclaration NOT NULL

```
CREATE TABLE Produit (
    référence CHAR(4) NOT NULL,
    nom VARCHAR(20) NOT NULL,
    marque VARCHAR(15),
    constructeur CHAR(20) DEFAULT 'Renault',
    PRIMARY KEY (référence)
);
```

Création de table (unicité)

Unicité des valeurs dans une colonne

UNIQUE (nom, prenom)

→ Principalement pour les clés secondaires

```
CREATE TABLE Produit (
    référence CHAR(4) NOT NULL,
    nom VARCHAR(20) NOT NULL,
    marque VARCHAR(15),
    constructeur CHAR(20),
    PRIMARY KEY (référence),
    UNIQUE (nom)
);
```

NOT NULL non spécifié
UNIQUE ne s'applique pas aux valeurs nulles

Création de table (clé étrangère)

FOREIGN KEY (att) REFERENCES ...

FOREIGN KEY (att1, ..., attn) REFERENCES ...

FOREIGN KEY(ref-produit) REFERENCES *Produit*

→ ref-produit référence la clé primaire de la table **Produit**

```
CREATE TABLE Vente (
    numéro INT(10) NOT NULL,
    ref-produit INT(10) NOT NULL,
    ref-client INT(10) NOT NULL,
    date DATE,
    PRIMARY KEY (numéro),
    FOREIGN KEY (ref-produit) REFERENCES Produit,
    FOREIGN KEY (ref-client) REFERENCES Client
);
```

Gestion de l'intégrité référentielle

Vérification, si insertion, suppression, mise à jour ...

Les valeurs prises par la clé étrangère correspondent-elles à la clé?

Action par défaut : rejet de l'opération

Autres actions si celle par défaut ne convient pas

- **ON UPDATE** (*en cas de mise à jour*)
- **ON DELETE** (*en cas de suppression*)

Objectif: faire respecter la contrainte

- **SET NULL** clé étrangère mise à NULL
- **CASCADE** application de la même opération
- **SET DEFAULT** valeur par défaut pour la clé étrangère

Intégrité référentielle (exemple)

```
CREATE TABLE Vente (
    numéro INT(10) NOT NULL,
    ref-produit INT(10) NOT NULL,
    ref-client INT(10) NOT NULL,
    date DATE,
    PRIMARY KEY (numéro),
    FOREIGN KEY (ref-produit) REFERENCES Produit
    ON DELETE SET NULL ON UPDATE CASCADE,
    FOREIGN KEY (ref-client) REFERENCES Client
    ON DELETE SET NULL ON UPDATE CASCADE
);
```

- Si suppression dans Produit : ref-produit = NULL
- Si mise à jour dans Produit : ref-produit = mise à jour

Mise à jour d'un schéma de BD

- Suppression d'une relation

`DROP TABLE nom de relation ;`

- Modification d'une table

`ALTER TABLE nom de relation ACTION description`

`ADD, MODIFY, ALTER, DROP, RENAME`

`description commande`

Mais aussi :

`ALTER TABLE Vente TYPE = innodb`

Modification d'une table (structure)

```
ALTER TABLE Produit MODIFY nom VARCHAR(25);
```

```
ALTER TABLE Produit ADD quantité-stock VARCHAR(20);
```

```
ALTER TABLE Client ALTER adresse SET DEFAULT 'NICE';
```

```
ALTER TABLE Client DROP adresse;
```

```
ALTER TABLE Client
    ADD CONSTRAINT CT_UN UNIQUE (nom) ;
```

```
ALTER TABLE Client
    ADD CONSTRAINT CT_PR PRIMARY KEY (numéro) ;
```

```
ALTER TABLE Vente ADD CONSTRAINT CT_ET
    FOREIGN KEY (ref-client) REFERENCES Client ;
```

Mises à jour des données

- **INSERT INTO**

pour insérer des tuple

- **DELETE FROM**

pour supprimer des tuples

- **UPDATE**

pour mettre à jour des tuples

Insertion de tuples (1)

Insertion avec désignation explicite des colonnes

```
INSERT INTO Produit (nom, marque, constructeur)  
VALUES ('mercedes 300 SL', 'mercedes benz',  
         'daimler');
```

Insertion dans l'ordre des colonnes

```
INSERT INTO Produit  
VALUES (DEFAULT, 'mercedes 300 SL', 'mercedes  
benz', 'daimler');
```

Insertion de tuples (2)

Insertion à partir d'une autre table

INSERT INTO *Produit* (nom, marque, constructeur)

SELECT P.nom, P.marque, p.constructeur

FROM *OldProduit P*

WHERE P.marque = 'Peugeot'



Evolution du schéma relationnel

Suppression de tuples

Suppression de tous les tuples

DELETE FROM *Produit*

ou

DELETE FROM *Produit* **WHERE** 1 > 0

Suppression conditionnelle

DELETE FROM *Produit*

WHERE marque = 'peugeot'

DELETE FROM *VENTE* **WHERE** ref-produit **IN**
(SELECT référence
FROM *Produit*
WHERE *Produit.marque* = 'peugeot') ;

Modification des tuples

```
UPDATE Produit  
SET nom = 'mercedes 300 SLK'  
WHERE numero =3;
```

Modification d'un tuple

```
UPDATE Produit  
SET prix-ht = prix-ht * 0,9  
WHERE référence NOT IN  
(SELECT ref-produit  
FROM VENTE  
WHERE quantité > 10);
```

*Modification d'une
collection de tuple*

Compléments: Définition de vues

Une vue est une table virtuelle dérivée de tables de base :

- On stocke seulement la définition de vue
- Son contenu est généré dynamiquement

```
CREATE VIEW ProduitPhare
AS
SELECT *
FROM Vente V, Produit P
WHERE
    V.ref-produit = P.référence
    AND V.quantité > 100 ;
```

```
CREATE VIEW VenteInfo
(référence, date, marque)
AS
SELECT P.référence, V.date,
       P.marque
FROM Vente V, Produit P
WHERE P.référence = V.ref-produit;
```

→ Evolution du schéma relationnel
Requêtes fréquentes

DROP VIEW VenteInfo

Compléments: Création de contraintes

CREATE ASSERTION

CHECK pour décrire des contraintes générales

CREATE ASSERTION QuantiteVendue

CHECK NOT EXISTS (

SELECT V.quantité

FROM Vente V, Produit P

WHERE

V.ref-produit = P.référence AND

P.quantité-vendue < V.quantité);

*On suppose que la table Produit
contient la colonne quantité-vendue*

Compléments: Création d'index

CREATE INDEX

Un index offre un chemin d'accès aux lignes d'une table

CREATE INDEX index1 ON Client(nom,prenom);

un index est systématiquement défini sur la clé primaire de chaque table

pour chaque clause UNIQUE utilisée dans la création de la table, un index permet de vérifier rapidement, au moment d'une insertion, que la clé n'existe pas déjà

Compléments: Gestion des privilèges

SHOW DATABASES ;

SHOW PRIVILEGES ;

GRANT *privileges* ON ... TO ...

REVOKE *privileges* ON ... FROM ...

ALL [PRIVILEGES]	Tous les droits sauf WITH GRANT OPTION
ALTER	Autorise l'utilisation de ALTER TABLE
CREATE	Autorise l'utilisation de CREATE TABLE
DELETE	Autorise l'utilisation de DELETE
DROP	Autorise l'utilisation de DROP TABLE
...	...