

BASE DE DONNÉES

JOURNÉE 2 - MATIN

1

Karine BRIFAUT

BASE DE DONNÉES

REQUÊTES

- Après la conception/modélisation de sa base il faut la valider en réalisant des requêtes.
- Tout comme le scripte de création de la base, les requêtes sont réalisées en SQL. Plus précisément se sera un DML (**Data Manipulation Language**)
- On se limitera à certaines requêtes SQL
 - INSERT, DELETE, UPDATE, SELECT

BASE DE DONNÉES

OPERATEURS

- Vous pourrez faire usage des opérateurs suivants dans vos ordres SQL
- Arithmétiques :
 - +, -, *, /, %
- Comparaisons :
 - =, (!=, <>), (<, !>), (>, !<), <=, >=
- Logiques:
 - ALL, AND, OR, ANY, BETWEEN, EXISTS, IN, LIKE, NOT, IS NULL, UNIQUE

! = not

comment

BASE DE DONNÉES

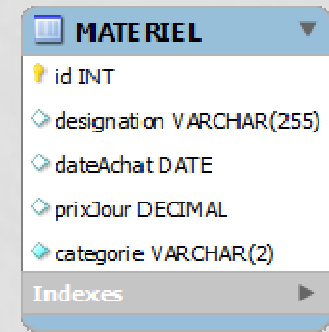
INSERT

- Permet d'ajouter un tuple dans une table.
- Attention : aux formats
 - Les dates : dans quel sens ?, jj-mm-aaaa ?
UTC
formaliser la manière de écrire la date
 - Les chaînes de caractères : avec des ' ou des "
 - Les nombres : avec des . ou des , ?

BASE DE DONNÉES

INSERT

- Exemple pour MySQL
 - On prendra comme référence la table suivante :



A screenshot of a database management tool showing the structure of a table named 'MATERIEL'. The table has five columns: 'id' (INT), 'designation' (VARCHAR(255)), 'dateAchat' (DATE), 'prixJour' (DECIMAL), and 'categorie' (VARCHAR(2)). There is an 'Indexes' section at the bottom with a right-pointing arrow.

MATERIEL	
id	INT
designation	VARCHAR(255)
dateAchat	DATE
prixJour	DECIMAL
categorie	VARCHAR(2)
Indexes ▶	

id, designation) values (1, tracteur)

- **INSERT INTO** *MATERIEL* **SET** designation='tracteur',
dateAchat='2015-01-22', prixJour=103.5,
categorie='IM';

BASE DE DONNÉES

INSERT

- Chaines de caractères : on utilise ' (et pas ")
- Date : MySQL est intelligent, il peut comprendre beaucoup de formats de date, mais le format 'standard' est yyyy-MM-dd La date se met à la méthode américaine
- Chiffre : les chiffres sont des chiffres (donc pas de "). Les chiffres décimaux ont des . et pas des ,

BASE DE DONNÉES

UPDATE

modification

- Permet de mettre à jour un ou plusieurs tuples déjà présents.

- Update est un mixte entre *insert* et *select*

je modifie

- **UPDATE** MATERIEL **SET** designation='tracteur',
dateAchat='2015-01-22', prixJour=103.5,
categorie='IM' **WHERE** id=1;

clé unique

l'unique chose que peut etre modifie
dans c'est cas c'est le prix ou la
catégorie

BASE DE DONNÉES

DELETE

supprime

- Permet de supprimer un ou plusieurs tuples dans une table.
- **DELETE FROM** *MATERIEL* **WHERE** id=1; on cascada
- **ATTENTION** : aux contraintes de clefs étrangères

BASE DE DONNÉES

SELECT

le select s'utilise poser de questions à la table

- Permet de sélectionner / filtrer un ensemble de tuples.
- Récupérer toutes les colonnes et les éléments d'une table
 - **SELECT** ^{tout} * **FROM** MATERIEL;
- Récupérer uniquement certaines colonnes d'une table
 - **SELECT** *id* **FROM** MATERIEL;

BASE DE DONNÉES

SELECT

- Récupérer toutes les colonnes et filtrer les éléments d'une table
 - **SELECT * FROM MATERIEL WHERE** id=1;
- Récupérer uniquement certaines colonnes et filtrer les éléments d'une table sur l'id = 1
 - **SELECT** id, prixJour **FROM** MATERIEL **WHERE** id=1;

il faut mettre à partir de l'ordre de colonnes

BASE DE DONNÉES

SELECT

- Récupérer toutes les colonnes et filtrer les éléments d'une table dont la désignation commence par 't'
 - **SELECT * FROM MATERIEL WHERE** designation **LIKE 't%';**
't%' mon nom va commencer par un T et après
- Récupérer uniquement certaines colonnes et filtrer les éléments d'une table qui ont un prixJour < 10
 - **SELECT id, prixJour FROM MATERIEL WHERE** prixJour<10; *AND prix > -0.5*

BASE DE DONNÉES

SELECT

- Récupérer toutes les colonnes et filtrer les éléments d'une table entre deux dates
 - **SELECT * FROM MATERIEL WHERE** dateAchat **BETWEEN** '2015-01-20 ' **AND** '2015-01-25';
- Récupérer uniquement certaines colonnes et filtrer les éléments d'une table qui ont une désignation null
 - **SELECT** *id, prixJour* **FROM MATERIEL WHERE** designation **IS NULL;** pas valeur

BASE DE DONNÉES

SELECT

- Prendre les éléments de catégorie IM et AM manière développer
 - **SELECT * FROM MATERIEL WHERE** categorie='IM' ou **OR** categorie='AM';
- Donne le même résultat : manière fonctionnelle
 - **SELECT * FROM MATERIEL WHERE** categorie **IN** ('IM', 'AM');

BASE DE DONNÉES SELECT

interroger deux tables à la fois

- Il est possible d'imbriquer les select :

requête imbriquée

- SELECT * FROM MaTable1 m1 WHERE (SELECT * FROM MaTable2 m2 WHERE m1.champ1 = m2.champ2)

jointure

- L'objectif étant de créer des ensembles de plus en plus petits.

patient
Id.
Id. Inf

Infirmiere
Id. Inf

$\text{infirmière.id Inf} = \text{patient.id Inf}$

```
select i.nom  
FROM ; infirmiere i, Patient p  
WHERE i.Id Inf = p.Id Inf
```

BASE DE DONNÉES

ORDER BY

- Permet d'ordonner le résultat d'une requête.
- S'accompagne
 - ASC : pour indiquer un trie ascendant (0,1,2,3 ...)
 - DESC : pour indiquer un trie descendant (9,8,7,6, ...)
- Concerne une ou plusieurs colonnes.
- Récupérer toutes les colonnes triées par date d'achat (plus récent en premier)
 - **SELECT * FROM MATERIEL ORDER BY** dateAchat **ASC;**

BASE DE DONNÉES

ORDER BY

- Récupérer la colonne id triée par date d'achat (plus récent en premier)
 - **SELECT** *id* **FROM** *MATERIEL* **ORDER BY** *dateAchat* **ASC**;
- Récupérer la colonne id triée par date d'achat (plus récent en premier) et si égalité de date prixJour (le plus cher en premier)
 - **SELECT** *id* **FROM** *MATERIEL* **ORDER BY** *dateAchat* **ASC**, *prixJour* **DESC**;

toujours de gauche à droite

BASE DE DONNÉES

COUNT

- Fonction qui permet de compter le nombre de tuples.
- Compte le nombre de tuples dans la table MATERIEL
 - **SELECT** ^{combient} **count(*)** **FROM** MATERIEL;
- Fait de même, mais on ne ramène que la colonne id
 - **SELECT** **count(id)** **FROM** MATERIEL;

BASE DE DONNÉES

SUM

- Fonction qui permet de faire une somme numérique sur une valeur de colonne.
- Fait la somme de tous les prix jour pour la date du 22 Janvier 2015
 - **SELECT SUM(prixJour) FROM MATERIEL WHERE dateAchat='2015-01-22' ;**

BASE DE DONNÉES

MAX - MIN

- Fonction qui permet de calculer le maximum ou le minimum sur une valeur de colonne.
- Trouve le prix le moins cher pour la date du 22 Janvier 2015
 - **SELECT MIN**(prixJour) **FROM** MATERIEL **WHERE** dateAchat='2015-01-22' ;

BASE DE DONNÉES

GROUP BY

- Ordre SQL qui permet de regrouper les résultats d'une requête.
- Concerne une ou plusieurs colonnes.
- Récupérer le prix le moins cher par designation
 - **SELECT MIN**(prixJour) **FROM** MATERIEL **GROUP BY** designation;


permet de faire catégories

BASE DE DONNÉES DISTINCT

- Ordre SQL qui indique que seuls les éléments différents nous intéressent dans le résultat de la requête.
- Indique tous les éléments matériels achetés le 22 janvier 2015 mais en ne prenant qu'un seul élément par désignation plusieurs éléments au'il ont la même désignation
 - **SELECT DISTINCT**(designation) **FROM** MATERIEL **WHERE** dateAchat='2015-01-22';

BASE DE DONNÉES LIMIT

- Ordre SQL qui permet de limiter le nombre de tuple retourné.
- Récupérer le premier client dont le nom commence par a
 - **SELECT * FROM CLIENT WHERE** nom LIKE 'a%' LIMIT 1;

BASE DE DONNÉES

ALIAS

- Un alias permet de nommer explicitement une colonne ou une table.
- Il peut
 - Simplifier l'affichage, le rendre plus lisible
 - Lever les ambiguïtés (quand une colonne a le même nom dans deux tables)
- On a placé un alias sur la table MATERIEL, dans cette requête elle s'appelle m
 - **SELECT MIN**(m.prixJour) **FROM** MATERIEL **AS** m **GROUP BY** m.designation;

BASE DE DONNÉES

ALIAS

As me permet de faire un alias

- On a placé un alias sur la table MATERIEL, pour cette requête elle s'appelle m. Idem sur le résultat
 - **SELECT MIN(m.prixJour) AS prixJourMin FROM MATERIEL AS m GROUP BY m.designation;**

dans c'est exemple est m = alias lien symbolique materiel

Alias = comment je vais exprimer quelque chose avec le AS

BASE DE DONNÉES

FAIRE UNE JOINTURE

- Une jointure permet de récupérer des informations qui relient plusieurs tables. pas faire de jointures des éléments qui sont différents d'un et l'autre
- Par exemple, tous les comptes qui appartiennent à un client.
- Dès qu'un ordre SELECT est réalisé sur deux tables, vous faites une jointure.

BASE DE DONNÉES

FAIRE UNE JOINTURE

- Récupérer le nom de tous les clients qui habitent dans le 78000

- **SELECT** nom **FROM** client, adresse toutes les tables
WHERE client.ADRESSE_id=adresse.id Jointure
AND adresse.codePostal=78000;

CLIENT	
id	INT
nom	VARCHAR(255)
type	TINYINT(1)
telephone	VARCHAR(20)
ADRESSE_id	INT
Indexes	

ADRESSE	
id	INT
numero	VARCHAR(10)
nomRue	VARCHAR(255)
nomVille	VARCHAR(255)
codePostal	INT
Indexes	



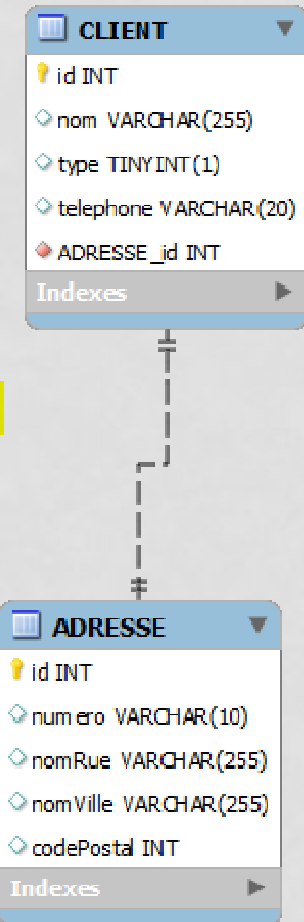
BASE DE DONNÉES

FAIRE UNE JOINTURE

- On peut aussi écrire :

table

- SELECT** nom **FROM** client **JOIN** adresse
ON client.ADRESSE_id=adresse.id
AND adresse.codePostal=78000;



BASE DE DONNÉES

JOINTURES

- Il existe quatre types de jointure comparaison avec le même élément
 - INNER JOIN (présenté avant)
 - Le résultat respecte scrupuleusement les critères
 - LEFT JOIN (ou LEFT OUTER JOIN)
 - En plus du résultat classique de l'INNER JOIN, seront ajoutés tous les éléments rejetés de la table 'de gauche'
 - RIGHT JOIN (ou RIGHT OUTER JOIN)
 - En plus du résultat classique de l'INNER JOIN, seront ajoutés tous les éléments rejetés de la table 'de droite'
 - FULL JOIN (ou FULL OUTER JOIN)
 - En plus du résultat classique de l'INNER JOIN, seront ajoutés tous les éléments rejetés de deux tables