BASES DE DONNEES RELATIONNELLE
Algèbre relationnelle et SQL
1 : Select mono-table

Manuels de référence du SQL :

Voir après le sommaire

Bertrand Liaudet

SOMMAIRE

Edition : octobre 2017

Manuels de référence du SQL

Chaque SGBD propose son manuel de référence en ligne sur internet.

Mémo SQL : http://www.volubis.fr/bonus/SQL_memo.htm

MySQL

Site officiel MySQL : https://www.mysql.com

Documentations MySQL : http://dev.mysql.com/doc/index.html

Documentations MySQL 5.0 en français :

http://bliaudet.free.fr/IMG/pdf/MySQL-refman-5.0-fr.pdf

C'est la bible ! On fait les recherches dans le « Search manuel ».

Quand on passe plusieurs mots (logique de « et »), il faut mettre des « + » devant chaque mot.

Pour chercher une phrase, il faut la mettre entre guillemets.

MariaDB

MariaDB est la suite opensource de MySQL

Site officiel MariaDB : https://mariadb.org/

ORACLE

http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/toc.htm

PostgreSQL

http://www.postgresql.org/docs/9.0/static/index.html

SQL Server

http://technet.microsoft.com/fr-fr/library/ms173372%28SQL.90%29.aspx

1 - MODELISATION : BASES DU MODELE RELATIONNEL

1. La modélisation

La modélisation est l'activité qui consiste à produire un modèle.

Un modèle est ce qui sert ou doit servir d'objet d'imitation pour faire ou reproduire quelque chose.

On s'intéresse ici à la modélisation des données.

Un modèle des données est une représentation de l'ensemble des données. Cette représentation prend en compte un outil de représentation (un langage) et un niveau de précision (des contraintes méthodologiques).

Il existe plusieurs modèles de représentation des données : hiérarchique, relationnel, entité-association, objet, ensembliste, etc.

Les deux modèles dominants actuellement sont : le **modèle relationnel : MR** (qui correspond aux SGBD-R) et le **modèle entité-association : MEA** (qui est indépendant du type de SGBD utilisé). Ces deux modèles correspondent à 2 langages différents.

Les schémas entité-relation et les diagrammes de classes UML peuvent être utilisés comme autres langages à peu près équivalents au MEA. On peut aussi les utiliser pour le MR.

La méthode MERISE, utilisée quasi-exclusivement en France, distingue entre 3 types de modèles selon des critères méthodologiques : le **modèle conceptuel des données : MCD**, le **modèle logique des données : MLD** et le **modèle physique des données : MPD**. L'usage tend à rendre équivalents **MCD et MEA**, **MLD et MR**, **MPD et SQL**.

Le MCD est du niveau de l'analyse fonctionnelle : il est adapté à la maîtrise d'ouvrage (MOA) et à la partie fonctionnelle de la maîtrise d'œuvre (MOE).

Le MLD est du niveau de l'analyse organique et est adapté à la maîtrise d'œuvre (MOE).

La « jungle » des modèles !			
Méthode	MCD	MLD	MPD
Langage	MEA, schema E-R, UML	MR	SQL
Type de langage	Algo client	Algo informaticien	Code
Niveau	Analyse fonctionnelle (externe). MOA ou MOE	Analyse organique (interne, technique) MOE	Réalisation MOE

2. Le modèle relationnel

Présentation

Le modèle relationnel a été inventé par Codd à IBM-San Jose en 1970.

C'est un modèle mathématique rigoureux basé sur un concept simple : celui de relation (ou table, ou tableau).

Ce modèle, c'est celui qui est implanté dans les SGBR-R.

Il permet à la fois de fabriquer la BD et de l'interroger.

Table, tuple, attribut, clé primaire

Exemple traité

Un service de ressource humaine dans une entreprise veut gérer le personnel. Dans un premier temps, on veut pouvoir connaître le nom, la fonction, la date d'entrée, le salaire, la commission (part de salaire variable) de chaque employé et le numéro du département dans lequel travaille chaque employé.

Chaque employé a donc les caractéristiques suivantes :

Nom, fonction, date d'entrée, salaire, commission, numéro de département

Table, tuples et attributs

Pour ranger ces données, on peut faire un tableau à 6 colonnes :

RELATION — 6 attributs :

Employés	Nom	Fonction	Date d'entrée	Salaire	Comm.	Num. Dept
	TURNER	SALESMAN	8-SEP-81	3000	0	10
	JAMES	CLERK	3-DEC-81	1800	NULL	30
	WARD	SALESMAN	22-FEB-81	2500	500	20
4 tuples :	TURNER	ANALYST	3-DEC-81	5000	NULL	10

Vocabulaire

Relation = tableau = table = classe = ensemble = collection

Tuple = ligne du tableau = élément = enregistrement = individu = objet = donnée

Attribut = colonne du tableau = caractéristique = propriété = champ

BD = toutes les lignes de toutes les tables

NULL

NULL est la seule information codée qu'on rentre dans une table : elle signifie « non renseignée ». La valeur « 0 », par contre, ne signifie pas du tout « non renseignée », mais bien « valeur = 0 », comme on dirait « valeur = 500 ».

Clé primaire

On souhaite pouvoir distinguer facilement chaque ligne d'une autre ligne. Or, certains employés ont le même nom.

Pour distinguer chaque ligne, on introduit la notion de clé primaire.

La clé primaire est un attribut qui identifie un tuple et un seul. Cela veut dire que quand on connait la clé primaire, on sait de quel tuple on parle, et on peut donc accéder sans ambiguïté à toutes les autres informations du tuple.

Une clé primaire est toujours renseignée.

Exemple type de clé primaire : le numéro de sécurité sociale dans un tableau de personne. Quand on connaît le numéro de sécurité sociale, on sait de qui on parle, donc tous les attributs sont déterminés (même si on ne connaît pas leur valeur à un instant donné).

Dans le tableau des employés, la clé primaire pourrait être un numéro de référence choisi par l'entreprise. On le nomme NE (pour Numéro d'Employe).

RELATION 7 attributs :

Employés	**NE**	Nom	Fonction	Date d'entrée	Salaire	Comm.	Num. Dept
	1	TURNER	SALESMAN	8-SEP-81	3000	0	10
	2	JAMES	CLERK	3-DEC-81	1800	NULL	30
	3	WARD	SALESMAN	22-FEB-81	2500	500	20
4 tuples :	4	TURNER	ANALYST	3-DEC-81	5000	NULL	10

Clé secondaire

Une clé secondaire est un attribut qui pourrait être clé primaire, mais qui ne l'est pas.

Par exemple, dans le tableau des employés, on pourrait avoir le numéro de sécurité social. Cet attribut détermine tous les autres. Si on garde le numéro d'employé comme clé primaire, le numéro de sécurité sociale est alors clé secondaire.

En l'occurrence, on a tout intérêt à ne pas faire du numéro de sécurité sociale la clé primaire car on peut imaginer que l'employé existe sans que cette information soit renseignée.

Une clé secondaire peut ne pas être renseignée.

Clé significative

La clé significative, c'est l'attribut qui sert de clé dans le langage ordinaire. Dans le cas des employés, c'est leur nom. Toutefois, il peut y avoir des homonymes : la clé significative est utile dans le langage ordinaire pour savoir de quoi on parle, mais elle est insuffisante dans le langage mathématique pour garantir l'identification de l'individu.

Schéma de la BD

Schéma des tables et schéma de la BD

Le schéma d'une table consiste à écrire la table sur une ligne avec les noms de code des attributs:

EMP(NE, nom, fonction, dateEmb, sal, comm)

L'ensemble des schémas des tables constitue le schéma de la BD.

Formalisme

- **La clé primaire** est notée en premier et est soulignée.
- **Les clés secondaires** sont notées après la clé primaire et mises entre parenthèses.
- La table des employés représente une réalité physique. On l'appelle « **table-nom** ».
- **Le nom donné à une « table-nom »** est un nom commun, au pluriel puisqu'une table contient plusieurs tuples. Toutefois, on peut aussi les mettre au singulier si ces tables sont destinées à produire des classes dans un mapping relationnel objet car le nom d'une classe, en tant que type, est par contre au singulier.
- La clé primaire d'une « table-nom » est N (pour numéro) suivi des premières lettres du nom de la table (une seule éventuellement). Cette technique n'est d'usage que pédagogique ! On peut aussi les nommer « idNomDeLaTable » ou encore « id », quelle que soit la table.

Expression ultra-schématique

De façon ultra-schématique, la table EMP peut s'écrire ainsi :

E (E, Ex)

- E, c'est le nom de la table (initial de Emp).
- E : c'est la clé primaire de la table E
- Ex : ce sont les attributs de la table E

L'expression ultra schématique consiste à ne distinguer qu'entre les attributs clés et les attributs non clés. On abordera l'intérêt de cette représentation dans le cas de modélisation complexe.

Définition d'une BD

Une BD c'est un ensemble de tables avec leurs tuples.

Un SGBD gère une ou plusieurs BD.

2 – INTRO. AU SQL ET A LA « CALCULETTE » SQL

PRINCIPALES NOTIONS	
Algèbre relationnelle	SQL
Manuel de référence	Calculette SQL

1. Algèbre relationnelle et SQL

Présentation de l'algèbre relationnelle

L'algèbre relationnelle c'est l'algèbre des relations, c'est-à-dire l'algèbre des tables.

De même que l'algèbre des nombres (c'est-à-dire l'algèbre commune) est la théorie des opérations portant sur les nombres, l'algèbre relationnelle est la théorie des opérations portant sur les tables.

L'intérêt du modèle relationnel réside dans ses capacités de représentation des données, mais aussi dans le fait qu'il permet de développer une algèbre constituée d'un petit nombre d'opérations qui permettent tous les traitements possibles sur les relations.

De même que les opérations de l'algèbre des nombres portent sur des nombres et produisent des nombres comme résultats, **les opérations de l'algèbre relationnelle portent sur des tables et produisent des tables comme résultats**.

Les opérations de l'algèbre relationnelle

L'algèbre relationnelle permet de :

- Créer, modifier, détruire des tables (et donc des attributs) : DDL
- Créer, modifier, détruire des tuples : DML
- Consulter les tuples : DSL.

La consultation des tuples d'une table consiste à :

- Filtrer les attributs
- Filtrer les tuples
- Trier les données
- Faire des opérations statistiques

  Ces opérations peuvent s'appliquer à une ou plusieurs tables.

Exemple d'opérations :

- Créer la table des employés (création d'une table et de ses attributs : DDL).
- Créer les employés dans la table des employés (création de tuples dans une table : DML).
- Augmenter le salaire de tous les employés (modification de tuples dans une table : DML).
- Quel est le nom de tous les employés qui sont vendeurs ? (filtre des lignes et des colonnes : DSL).

- Quel est le nom de tous les employés travaillant dans tel département ? (filtre des lignes et des colonnes à partir de deux tables : DSL).
- Donner la liste des employés par ordre alphabétique (tri : DSL).
- Quel est le salaire moyen de tel métier de l'entreprise ? (filtre et calcul statistique : le résultat est une table avec un tuple et un attribut : DSL).

Le SQL ou algèbre relationnelle ?

Le **SQL** (Structured Query Langage) est un langage de programmation **fondé sur l'algèbre relationnelle** qui est la théorie des opérations qu'on peut appliquer à un objet mathématique particulier : les tableaux de données (appelées « relations » en algèbre relationnelle).

Toutefois, le SQL peut être considéré comme un **ensemble d'opérateurs** plutôt que comme un langage de programmation.

En ce sens, l'application cliente du SGBD qui permet d'utiliser directement le SQL peut être considérée comme une **« calculette SQL »** qui, au lieu de travailler sur des nombres, travaille sur des tableaux de données.

Intérêt de l'algèbre relationnel par rapport au SQL :

- Indépendance par rapport à tout SGBD
- Formalisme mathématique plus concis
- Un opérateur en plus : celui de division

Intérêt du SQL par rapport à l'algèbre relationnel

- Le SQL permet de créer les BD et de tester les calculs.
- Le SQL est un langage normalisé (ANSI - ISO) et implanté dans tous les SGBD-R
- Le formalisme du SQL est très proche de celui de l'algèbre relationnel.
- L'opérateur de division est un opérateur complexe et rarement utilisé et équivalent à la combinaison d'autres opérateurs du SQL.

2. Les commandes du SQL

4 types de commandes

Les commandes du SQL se divisent en 4 sous ensembles :

- Le DSL : data select language (formulation non standard)
- Le DML : data manipulation language
- Le DDL : data definition language
- Le DCL : data control language

Le DSL : SELECT : l'algèbre relationnnelle

Le SELECT est la commande qui permet de consulter les données de la BD.

C'est le SELECT qui va mettre en oeuvre l'algèbre relationnelle.

Le DML : commandes des tuples : INSERT, UPADTE, DELETE

Ce sont les commandes qui vont permettre de créer, modifier, détruire les tuples.

Le DDL : commandes des tables : CREATE, ALTER, DROP

Ce sont les commandes qui vont permettre de créer, modifier, détruire les tables : CREATE TABLE, ALTER TABLE, DROP TABLE.

Ces commandes permettront aussi de créer, modifier et supprimer d'autres objets de la BD : les BD, les séquences (auto-incréments ORACLE), les index, les vues, etc. CREATE DATABASE, CREATE VIEW, CREATE INDEX, etc

Le DCL : commandes de contrôle

La commande **CREATE USER** permet de créer des utilisateurs qui pourront se connecter à la base de données. **DROP USER** et **ALTER USER** permettront la suppression et la modification.

La commande **GRANT** permet de donner des droits aux utilisateurs : droits de création, modification, suppression de tables, et droits de création, modification, suppression et consultation de tuples. Cette commande permet aussi de créer des utilisateurs.

La commande **REVOKE** permet de supprimer les droits créés par la commande GRANT.

La commande **COMMIT** permet de valider les modifications dans la BD (les commandes du DML). Selon les environnements, les modifications peuvent être validée automatiquement par défaut ou pas (variable d'environnement AUTOCOMMIT à vrai ou à faux).

La commande **ROLLBACK** permet au contraire de revenir en arrière, s'il n'y a pas eu de validation manuelle ou automatique.

3. Manuels de référence du SQL

3 - SQL : CONSULTATION DE LA BASE DE DONNEES

PRINCIPALES NOTIONS	
Projection - Restriction	**SELECT / FROM / Where**
Distinct	**year / month / day**
Tri	**length / substr / concat**
Attribut calculé	**in, between, like**
Projection primaire	**Order by / asc / desc**

1. La commande de recherche : le select

La commande SELECT permet de faire des opérations de recherche et de calcul à partir des tables de la base de données.

On peut diviser toutes les opérations réalisables par un select en deux grandes catégories :

- Les opérations s'appliquant à une seule table
- Les opérations s'appliquant à plusieurs tables

Les opérations s'appliquant à une seule table

Il y a 4 grandes catégories d'opérations s'appliquant à une seule table :

- Les filtres sur les colonnes : projection
- Les filtres sur les lignes : restriction
- Les tris
- Les opérations statistiques

Exemples de questions traitées par le SELECT :

- Quel est le nom de tous les employés qui sont vendeurs ? (filtre des lignes et des colonnes).
- Quel est le nom de tous les employés travaillant dans tel département ? (filtre des lignes et des colonnes à partir de deux tables).
- Donner la liste des employés par ordre alphabétique (tri).
- Quel est le salaire moyen de tel métier de l'entreprise ? (filtre et calcul statistique : le résultat est une table avec un tuple et un attribut).
- Quels sont les employés qui gagnent plus que la moyenne des salaires de l'entreprise ? (calcul statistique avec regroupements).
- Combien y a-t-il d'employés par tranche de 1000 de salaire ?

Les opérations s'appliquant à plusieurs tables

Il y a 3 grandes catégories d'opérations s'appliquant à plusieurs tables :

- Les opérations ensemblistes classiques : union, intersection, différence.
- Le produit cartésien et la jointure associée
- Les imbrications d'opérations.

Manuels de référence du Select

ORACLE

http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/queries001.htm#i2053893

MySQL

http://dev.mysql.com/doc/refman/5.6/en/sql-syntax.html

PostgreSQL

http://www.postgresql.org/docs

Dans un moteur de recherche : chercher postgresql docs et le ou les mots clés

SQL Server

http://technet.microsoft.com/fr-fr/library/ms174113%28SQL.90%29.aspx

2. Projection = filtre des colonnes

Projection = filtre des colonnes : tous les tuples, certains attributs

Présentation

La projection d'une table est une nouvelle table constituée de tous les tuples et de certains attributs de la table de départ.

TABLE	Attribut 1	attribut 2	attribut 3	• • •	attribut n
tuple 1					
tuple 2					
tuple n					

En gris : les colonnes sélectionnées.

Syntaxe AR

Tres = Proj(*table* ; *liste d'attributs)*

Syntaxe SQL

La syntaxe d'une projection est la suivante[1] :

```
SELECT liste d'attributs [2] FROM table ;
```

Exemples

Donner les noms de tous les employés :

```
SELECT NE, nom FROM emp;
```

Remarque 1 :

En toute rigueur, il est nécessaire de projeter NE même s'il n'est pas explicitement demandé dans la question, car la table produite ne doit pas contenir de tuples en double. Pour distinguer les homonymes, on projette la clé primaire de la relation.

Donner la liste de tous les employés avec tous leurs attributs :

```
SELECT * FROM emp;
```

1 Remarques sur le métalangage utilisé : il ne prétend pas être parfaitement formel! Les mots clés du langage SQL sont en gras (**Select**) ou pas... Les expressions générales sont en italiques (*liste d'attributs*). Les explications concernant ces expressions générales sont données soit en note, soit dans le texte, soit à travers des exemples. Les cas particuliers des exemples sont au format standard (NE, nom).

2 Les attributs de la liste d'attributs sont séparés par une virgule.

Deux types de projection

Quand on crée une nouvelle table, en toute rigueur, il faut éviter qu'il y ait des tuples en double.

Il y a deux manières d'éviter les doublons :

- Projeter la clé primaire (projection primaire)
- Eliminer les doublons (projection avec distinct)

a) Projection primaire

La syntaxe est la suivante :

```
SELECT clé primaire, liste d'attributs  FROM table ;
```

La clé primaire de la table résultat est la clé primaire de la table de départ.

Exemple : tous les employés avec leurs fonctions.

```
SELECT NE, nom, fonction
FROM emp;
```

Le résultat est une table d'employés avec moins d'attributs.

Remarque :

Puisqu'on veut les employés, il faut aussi projeter l'attribut donnant le nom (nom), c'est-à-dire la **clé significative**.

b) Projection avec élimination des doublons : la clause distinct

La clause **distinct** permet, à partir d'une projection, d'éliminer les tuples en doubles

```
SELECT distinct liste d'attributs  FROM table  ;
```

Exemples : pour obtenir les différents métiers de la société, on écrira :

```
SELECT distinct fonction FROM emp ;
```

Pour obtenir les différents métiers de la société par numéro de département, on écrira :

```
SELECT distinct ND, fonction FROM emp ;
```

Remarque 1 :

La clé primaire de la table résultat est constituée par la liste des attributs projetés. Les tuples produits sont donc conceptuellement différents des tuples présents dans la table d'origine. Dans l'exemple, on produits des métiers (« fonction ») en partant d'employés.

Remarque 2 :

On ne doit jamais projeter la clé primaire auquel cas on retomberais sur la table d'origine.

Inversement, si la liste d'attributs projetés contient la clé primaire, alors le distinct ne sert à rien :

```
SELECT distinct clé primaire, liste d'attributs
FROM table  ;
```

équivaut à :

```
SELECT clé primaire, liste d'attributs  FROM  table ;
```

Remarque 3:

La table est classée dans un ordre croissant (cf. clause **order by** à venir). Ceci vient du fait que l'ordre initial n'a plus aucun sens dans le résultat d'un distinct.

Création d'attributs calculés

Présentation

Il est possible de créer des attributs qui soient le résultat d'une opération arithmétique ou autre faite à partir d'autres attributs.

```
SELECT liste d'attributs avec des opérations sur attributs
FROM table ;
```

Exemples

tous les salaires, commissions et salaires totaux des salariés :

```
SELECT NE, nom, sal, comm, sal + comm
FROM emp;
```

nombre de lettres du nom de chaque employé :

```
SELECT NE, nom, length(nom)
FROM emp;
```

année d'embauche de chaque employé

```
SELECT NE, nom, datemb, year(datemb)
FROM emp;
```

Salaire par tranche : petit (<1000), moyen (<2000), gros : autres.

```
SELECT NE, nom,
CASE
   WHEN sal < 1000 THEN "petit"
   WHEN sal < 2000 THEN "moyen"
   ELSE "gros"
END
FROM emp;
```

Renommer les attributs

Présentation

On peut renommer les attributs projetés : le nouveau nom s'affichera à la place du nom de l'attribut. Ce nouveau nom n'est valable que le temps de la requête.

```
SELECT ancien_nom  nouveau_nom FROM table;
```

Si le nouveau nom contient des espaces, on écrira :

```
SELECT ancien_nom "nouveau nom" FROM table;
```

Remarques

On peut mettre des apostrophes à la place des guillemets

On peut mettre **« as »** entre l'ancien nom et le nouveau nom

```
SELECT ancien_nom  AS  nouveau_nom FROM table;
```

Exemple

```
SELECT NE, nom, sal, comm, sal + comm as salaireTotal
FROM emp;
```

Les opérations possibles pour calculer un attribut

Opérateurs et fonctions classiques

On peut utiliser tous les opérateurs et toutes les fonctions comme on l'a vu dans les premiers exemples.

Les chapitres suivants montreront d'autres opérateurs et d'autres fonctions.

IF (pas standard)

Le IF permet de faire des tests dans le SELECT avec 2 alternatives, 2 embranchements. Le passage par un embranchement permet de renvoyer une valeur et une seule : celle de l'attribut calculé.

Le IF est un cas particulier du CASE qui réduit les valeurs possibles à 2 :

IF(expr1,expr2,expr3) : retourne expr2 si expr1 est différent de 0 et de null, expr3 sinon.

Exemple 1 : IF simple

On crée un attribut, « categorie » qui aura deux valeurs : « bas » et « gros » en fonction du salaire. On met la limite à 2000.

Syntaxe : if(condition, résultat si vrai, résultat si faux)

```
SELECT NE, nom, sal,
  IF(sal<2000,"1:bas","3:gros")) AS categorie
FROM emp
ORDER BY categorie, nom;
```

Exemple 2 : IF imbriqué

Pour avoir plus de deux alternatives, on peut imbriquer les IF

```
SELECT NE, nom, sal,
  IF(sal<1200,"1:bas",if(sal<2500,"2:moyen","3:gros"))
  AS categorie
FROM emp
ORDER BY categorie, nom;
```

CASE WHEN

Présentation

Le "case when" est une façon plus lisible d'écrire des IF imbriqués.

En général, il a deux usages et deux syntaxes :

- usage « else-if » classique : succession de tests indépendants.
- usage « switch » classique : comparaison d'égalité entre une expression unique de départ.

Dans tous les cas, le CASE WHEN renvoie une seule valeur : c'est celle qui sera prise en compte dans la projection.

En général le ELSE final ou un DEFAULT permet de garantir que tous les cas possibles seront traités.

Première syntaxe : équivalent de IF imbriqués

La première syntaxe permet de calculer le nouvel attribut à partir d'une succession de tests distincts les uns des autres. C'est un « else if » algorithmique classique.

Syntaxe :

```
CASE
   WHEN expression opérateur valeur THEN résultat
   [* WHEN … THEN …]
   [ELSE résultat]
END
```

Exemple :

```
SELECT NE, nom, sal,
CASE
   WHEN sal<1200 THEN "1:< mille"
   WHEN sal<2500 THEN "2:moyen"
   ELSE "3:gros"
END AS categorie
FROM emp;
```

Deuxième syntaxe : switch classique

La deuxième syntaxe permet de calculer le nouvel attribut à partir d'une **succession de comparaisons d'égalité** entre une expression unique de départ et différentes valeur : c'est un « case » ou un « switch » algorithmique classique.

Syntaxe :

```
CASE expression
   WHEN valeur THEN résultat1
   [* WHEN … THEN …]
   [ELSE résultat]
END
```

Valeur et résultat sont des expressions (constantes littérales, attributs, expressions arithmétiques, etc.)

Le ELSE permet de garantir que tous les tuples seront pris en compte. Si il n'y a pas de ELSE et que tous les cas ne sont pas pris en compte dans les alternatives du ELSE, le tuple ne sera pas éliminé (seul le WHERE peut faire ça), mais la valeur de l'attribut calculé vaudra NULL.

Exemple :

```
SELECT NE, nom, sal,
CASE sal div 1000
   WHEN 0 THEN "< mille"
   WHEN 1 THEN "mille et plus"
   WHEN 2 THEN "2 mille et plus"
   WHEN 3 THEN "3 mille et plus"
   ELSE "plus de 4 mille"
END AS categorie
FROM emp;
```

Select imbriqué dans la projection

On peut imbriquer un select dans les attributs projetés à condition que ce select ne renvoie qu'une seule valeur.

Exemple :

```
SELECT NE, nom, sal,
   (SELECT max(sal)FROM emp) AS maxSal
FROM emp;
```

Usage :

MIEUX VAUT EVITER LES SELECT IMBRIQUES !

Autres possibilités

Selon les SGBD-R, on peut trouver d'autres possibilités.

ORACLE propose la clause DECODE qui est à peu près équivalente à un CASE WHEN, et aussi la clause WIDTH_BUCKET qui permet de créer un attribut catégoriel à partir d'un attribut continu.

MySQL propose la fonction « IF» qui permet de coder l'équivalent d'un CASE.

3. Restriction = filtre des lignes

Restriction = filtre des lignes : certains tuples, tous les attributs

Présentation

La restriction d'une table est une nouvelle table constituée de certains tuples et de tous les attributs de la table de départ.

TABLE	attribut 1	attribut 2	Attribut 3	• • •	attribut n
tuple 1					
tuple 2					
tuple n					

En gris : les lignes sélectionnées.

Syntaxe AR

Tres = Rest(*table* ; *formule logique de sélection des tuples)*

Syntaxe SQL

La syntaxe d'une restriction est la suivante :

```
SELECT * FROM table
WHERE formule logique de sélection des tuples ;
```

La formule de sélection des tuples fait intervenir les opérateurs habituels de l'algèbre booléenne:

=, !=, >, >=, <, <=, or, and, not

Cf. paragraphe 3.6 sur les opérations sur les opérateurs de comparaison

Exemples

Tous les employés du département 30 avec tous leurs attributs :

```
SELECT * FROM emp
WHERE ND = 30;
```

Il existe aussi trois opérateurs spécifiques au SQL :

in, between, like

Cf. paragraphe 5 sur les opérations sur les opérateurs de comparaison.

4. Restriction et projection = filtre des lignes et des colonnes

Restriction- projection = filtre des lignes et des colonnes : certains tuples, certains attributs

Présentation

La restriction-projection d'une table est une nouvelle table constituée de certains tuples et de certains attributs de la table de départ.

On fait d'abord la restriction, puis on projette.

C'est l'opération la plus courante.

TABLE	attribut 1	attribut 2	Attribut 3	• • •	attribut n
tuple 1					
tuple 2					
tuple n					

En gris clair : les lignes et les colonnes sélectionnées.

En gris foncé : le croisement des lignes et des colonnes sélectionnées.

2 types de restriction-projection

Quand on crée une nouvelle table, il faut éviter qu'il y ait des tuples en double. Il y a deux manières d'éviter les doubles qui correspondent à trois types de restriction-projection :

- la restriction-projection primaire (avec projection de la clé primaire)
- la restriction-projection avec élimination des tuples en doubles

Syntaxe AR

T1 = Rest (*table ; formule logique de sélection des tuples)*

Tres = Proj (T1 ; *liste d'attributs)*

Les deux opérations sont faites l'une après l'autre : d'abord la restriction, puis la projection.

On peut aussi écrire :

Tres = Proj (Rest (*table ; formule logique de sélection des tuples)* ; *liste d'attributs)*

Cependant, on évitera cette écriture qui est peu lisible.

Restriction-projection primaire

Présentation

La syntaxe restriction-projection primaire est :

```
SELECT clé primaire, liste d'attributs  FROM table
where formule logique de sélection des tuples ;
```

La restriction-projection avec clé projette la clé de la table de départ. Conceptuellement, elle produit donc des tuples qui sont des objets restreints (ayant moins d'attributs) et appartenant à la table de départ.

Exemple

tous les employés dont le salaire est compris entre 1200 et 1400

```
SELECT NE, nom, sal FROM emp
where sal between 1200 and 1400 ;
```

Le résultat est une table d'employés avec moins d'attributs.

Attributs projetés

La liste des attributs projetés est la suivante, dans l'ordre

CP – (CS) – demandés – (restriction)

Les attributs entre parenthèses sont facultatifs mais fortement recommandés.

1) CP : on projette la clé primaire en premier pour savoir de quoi-qui on parle et pour éviter les doublons.

2) CS : on projette la « clé significative », CS en second. La CS est l'attribut qui fait office de clé primaire dans le langage courant. Projeter la CS n'est pas obligatoire d'un point de vue mathématique, par contre c'est nécessaire pour rendre les résultats lisibles. Dans nos exemple, la CS, c'est le nom.

3) Attributs demandés : on projette obligatoirement les attributs demandés.

4) Attributs de restriction : on projette les attributs qui participent aux restriction pour vérifier que les conditions sont bien réalisées. Département, salaire et fonction dans nos exemples.

Syntaxe générale SQL d'une restriction-projection primaire

La syntaxe générale d'une restriction-projection primaire est donc :

```
SELECT clé primaire, clé significative,
       attributs demandés,
       attributs de restriction
FROM table
where formule logique de sélection des tuples ;
```

Restriction-projection avec élimination des tuples en double : la clause distinct

Syntaxe SQL

La clause **distinct** permet, à partir d'une projection, d'éliminer les tuples en doubles

```
SELECT distinct liste d'attributs  FROM table  ;
```

Exemple

Pour obtenir les différents métiers de la société avec un sal > 2000 , on écrira :

```
SELECT distinct fonction FROM emp WHERE sal >2000;
```

Pour obtenir les différents métiers de la société par numéro de département, on écrira :

```
SELECT distinct ND, fonction FROM emp WHERE sal >2000;
```

Remarques

1) A la différence d'un restriction-projection primaire, dans le cas de l'utilisation d'un distinct, on ne veut pas - et on ne doit pas - projeter la clé primaire. En effet, si on projette la clé primaire, les tuples produits seront des objets restreints de la table de départ, tandis qu'avec la clause distinct on produit des objets qui, conceptuellement, n'appartiennent pas à la table de départ. Les métiers ne sont pas des employés, les couples (numéro de département, métier) non plus.

2) La table est classée dans l'ordre d'apparition de la première occurrence des valeurs projetées.

3) Dans ce cas, à la différence d'une restriction-projection primaire, les tuples obtenus ne correspondent pas à un et un seul tuple de la table de départ.

4) Si la liste d'attributs projetés contient la clé primaire, alors le distinct ne sert à rien :

```
SELECT distinct clé primaire, liste d'attributs
FROM table  ;
```

équivaut à :

```
SELECT clé primaire, liste d'attributs  FROM  table ;
```

Syntaxe AR

Le distinct n'existe pas en algèbre relationnelle car il est considéré comme étant effectué systématiquement : une table résultat ne contient jamais de doublons, de toute façon.

Syntaxe générale SQL d'une restriction-projection avec distinct

La syntaxe générale d'une restriction-projection avec distinct est la suivante :

```
SELECT distinct clé primaire, clé significative,
           attributs demandés,
           attributs de restriction
FROM table
where formule logique de sélection des tuples ;
```

C'est la même que pour une restriction-projection primaire au distinct près. A noter que la clé primaire devient clé primaire après élimination des doublons.

5. Opérateurs et fonctions

Présentation

Principes

Il existe de nombreuses fonctions et de nombreux opérateurs fournis par le SQL standard mais aussi des fonctions et des opérateurs spécifiques à chaque SGBD. Ces outils permettent de faire des calculs puissants de façon simple : il faut s'y intéresser en fonction de du SGBD qu'on utilise.

La présentation ci-dessous fixe les principales catégories de fonctions et d'opérateurs. Toutefois, pour les découvrir de façon plus exhaustive, il faut se référer au manuel de référence du SGBD accessible sur internet.

http://www.lirmm.fr/~jq/Cours/MASS/Licence/BaseDeDonnees/doc/calculEnMySQL.html

Manuels de référence

Oracle

http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/operators.htm#SQLRF003

http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/functions.htm#SQLRF006

MySQL

http://dev.mysql.com/doc/refman/5.6/en/functions.html

PostgreSQL

http://www.postgresql.org/docs

Dans un moteur de recherche : chercher postgresql docs et le ou les mots clés

Les opérations arithmétiques : la calculette arithmétique

Présentation

On peut faire des opérations arithmétiques en utilisant les opérateurs arithmétiques habituels :

+, **-**, *, /, div, (,)

div est l'opérateur de division entière (modulo). Il est équivalent à floor( ) en MySQL.

Mais aussi toutes les fonctions mathématiques standards :

sin, cos, log, exp, power, etc.

La calculette arithmétique

A noter que la calculette SQL est aussi une calculette arithmétique.

On peut écrire :

```
SELECT 4*3 ;
```

Ou bien

```
SELECT 4*log(2,8) ;
```

Ou bien

```
SELECT 24*sin(3.1416/6) ;
```

On obtiendra le résultat du calcul : 12.

Quelques fonctions mathématiques en vrac

http://dev.mysql.com/doc/refman/5.6/en/mathematical-functions.html

abs (nb), **ceil** (nb) : arrondi vers le haut ; **floor** (nb) : arrondi vers le bas ; **round** (nb) arrondi au plus proche ; **power** (nb, puissance) : élévation à la puissance ; **sqrt** (nb) : racine ; **truncate** (nb, nbDécimales) : arrondi en vers le bas en précisant le nombre de décimales conservées ; **pi**( ) : renvoie PI.

Manuel de référence

MySQL

http://dev.mysql.com/doc/refman/5.6/en/functions.html

http://dev.mysql.com/doc/refman/5.6/en/mathematical-functions.html

http://dev.mysql.com/doc/refman/5.6/en/arithmetic-functions.html

Les opérateurs de comparaison

Présentation

On peut faire des comparaisons en utilisant les opérateurs de comparaisons habituels :

'=', '<>', ' !=', '<', '<=', '>', '>='

On peut aussi utiliser les opérateurs booléens

'**AND'**, '**OR**', '**NOT**'

On peut aussi utiliser les opérateurs spéciaux :

In, Not in

Between … and…

Like

in est vrai si le terme de gauche se trouve dans la liste proposée par le terme de droite.

between est vrai si le terme de gauche se trouve entre deux valeurs proposées.

like est vrai si la chaîne de caractères du terme de gauche est au format proposé dans le terme de droite.

Exemples

Tous les employés dont le salaire est compris entre 1200 et 1400

```
SELECT NE, nom, sal FROM emp
where sal between 1200 and 1400 ;
```

Tous les employés travaillant dans les départements 10 ou 30 :

```
SELECT NE, nom, ND FROM emp
where ND in (10, 30) ;
```

like

like est vrai si la chaîne de caractères du terme de gauche est au format proposé dans le terme de droite.

Tous les employés qui ont un E comme troisième lettre de leurs noms :

```
SELECT NE, nom FROM emp
where nom like '_ _ E %' ;
```

le "_" signifie : n'importe quel caractère. Le "%" signifie n'importe quelle chaîne de caractères, dont la chaîne vide.

Attention à l'ordre de priorité des opérateurs !!!

L'ordre de priorité des opérateurs est l'ordre habituel de l'arithmétique

not, *, +, <, and, or

Les opérations sur les valeurs NULL

Présentation

Quand une valeur NULL intervient dans une opération, quelle qu'elle soit, le résultat de cette opération vaut NULL

Opérateurs

Pour savoir si une valeur vaut NULL, on utilise les opérateurs :

is

is not

Remplacer la valeur NULL par 0

On peut remplacer la valeur NULL par une valeur au choix (0 le plus souvent) avec la fonction suivante :

Ifnull (attribut, 0)

L'intérêt de cette transformation est de forcer les opérations malgré la présence d'une valeur NULL.

Equivalence dans les autres SGBD : avl (oracle), isnul (sqlserveur), ifnull(mysql), coalesce (postgres)

Exemple :

On veut remplacer les NULL par un 0

```
SELECT NE, nom, sal, ifnull(comm, 0)  FROM  emp;
```

Equivalence dans les autres SGBD

avl (oracle), **isnul** (sqlserveur), **ifnull** (mysql), **coalesce** (postgres)

Les fonctions de traitement de chaîne

Manuel de référence

MySQL

http://dev.mysql.com/doc/refman/5.6/en/string-functions.html

http://dev.mysql.com/doc/refman/5.6/en/string-comparison-functions.html

Les constantes

Les constantes chaînes de caractères sont entre guillemets, apostrophes ou accents grave (altGr+7, espace).

Les trois fonctions de base du traitement de chaînes de caractères

length fournit la longueur d'un attribut chaîne de caractères :

```
SELECT length (attribut ) FROM table ;
```

substr fournit un morceau d'un attribut chaîne de caractères :

```
SELECT substr (attribut, début, longueur )
FROM table ;
```

concat est une fonction de concaténation

```
SELECT concat (chaine1, chaine2, chaine3) ;
```

Avec ces trois fonctions, on peut faire tous les traitements possibles sur les chaînes (sauf les traitements liés à la distinction entre minuscule et majuscule).

Autres fonctions de traitement de chaînes

upper() : passe la chaîne en majuscule.

lower() : passe la chaîne en minuscule.

Substring ⇔ Mid() ⇔ Substr()

Reverse() : renverse l'ordre d'une chaîne.

Hex() : convertit une chaîne en code Hexa. L'intérêt de Hex() est de distinguer entre les minuscules et les majuscules.

Etc.

Autres fonctions de traitement de chaînes en vrac

http://dev.mysql.com/doc/refman/5.6/en/string-functions.html

http://dev.mysql.com/doc/refman/5.6/en/string-comparison-functions.html

Les constantes chaînes de caractères sont entre guillemets, apostrophes ou accents grave (altGr+7, espace).

Like, length(attribut) ; **substring**(attribut from déb for lgr) : sous-chaîne commençant en déb de longueur lgr ; **substring**(att from 11) : la sous-chaîne à partir du 11ème caractère ; **strcmp**(texte1, texte2) : renvoie 0 si les deux textes sont identiques, -1 si le premier est premier en ordre alphabétique, 1 sinon ; **trim**(texte) : élimine les espaces au début et à la fin ; **upper**(texte) ; **lower**(texte) ; **replace**(texte, ancien, nouveau) ; **locate**(mot, texte) : position d'un mot dans un texte ; **locate**(mot, texte, début) : position d'un mot dans un texte à partir de début ; **concat**(text1, text2, etc.) ; **concat_ws**(séparateur, text1, text2, etc.) : concaténation avec un séparateur ; **binary** : opérateur unaire qui rend un texte sensible à la casse : 'paris' = binary 'PARIS' est faux.

Regexp : expression régulière

Présentation

De nombreux SGBD propose un opérateur d'analyse d'expressions régulière.

MySQL propose le REGEXP, qui est un « LIKE » dont les caractères spéciaux ne se limiteront pas aux « % » et « _ » mais à ceux qu'on trouve dans les expressions régulières standards.

Des fonctions supplémentaires de manipulation des expressions (comptage, remplacement, etc.) existent et sont spécifiques à chaque SGBD.

Syntaxe du select

WHERE attribut REGEXP 'expression régulière'

Syntaxe de l'expression régulière

Classe de caractères : [1236], [1-4], [^1-4]

Nombre de caractères : {5}, {5,6}, {0,1}, ?, {1,}, +, {0,}, *

Début : ^

Fin : $

N'importe quel caractère : « . »

Caractère « ^ » : \\^

Caractère « . » : \\.

Alternative (ou) : |

Classes prédéfinies : [:alpha :], [:lower :], [:upper :], [:alnum :], [:space :], [:punct :],

Début de mot : [[:< :]]

Fin de mot : [[:> :]]

Exemples

'^[BL] | [0-9]-?[0-9]{4}$' veut dire : un B ou un L ou un chiffre, suivi d'un tiret ou pas, suivi de 4 chiffres.

'^[SA][A-Z]*\\.[^0-9]{3}[[:alnum:]]*$' veut dire : un S ou un A, suivi de plusieurs majuscules ou aucune, suivi de un point, suivi de pas de chiffre sur 3 caractères, suivi des chiffres ou des lettres ou rien.

Sensibilité aux accents et à la casse

Les expressions régulières sont sensibles ou pas à la casse selon la collation utilisée.

Elles sont toujours sensibles aux accents.

Manuel de référence

http://dev.mysql.com/doc/refman/5.7/en/regexp.html

Les opérateurs de traitement de date

La gestion des dates est la partie la moins standard entre les différents SGBD.

En général, les dates sont aussi considérées comme des chaînes de caractères. On peut donc leur appliquer tous les opérateurs de traitement de chaînes.

Il y a deux aspects spécifiques à prendre en compte :

- Les fonctions de manipulation des dates
- La question des formats

Mieux vaut régler le problème au niveau du SQL plutôt qu'au niveau du langage seveur.

La présentation ci-dessous présente quelques éléments dans différents SGBD

Les dates dans MySQL

Manuel de référence

http://dev.mysql.com/doc/refman/5.6/en/date-calculations.html

http://dev.mysql.com/doc/refman/5.6/en/date-and-time-functions.html

Principales fonctions de traitement de dates

year(), month(), day() : renvoient l'année, le mois et le jour dans le mois.

current_date : renvoie la date du jour

to_days(date) : renvoie le nombre de jours depuis le 1er janvier 0. to_days(« 0-1-1 ») vaut 1.

From_days(nb jours) : renvoie la durée correspondant à un nombre de jours. Cette durée a le format date.

Fonctions de traitement de date en vrac

http://dev.mysql.com/doc/refman/5.6/en/date-and-time-functions.html

curdate(), **current_date** : date ; **curtime**() , **current_time** : heure ; **now**(), **current_timestamp** : date et heure ; **sysdate()** : equivalent now mais calculé en temps réel ;

year(date) ; **month**(date) ; **day**(date) ; **hour**(date) ; **minute**(date) ; **second**(date) ;

to_days(date), **from_days**(entier) : conversion d'une date en jours. to_days(date) : renvoie le nombre de jours depuis le 1er janvier 0. to_days(« 0-1-1 ») vaut 1. to_days démarre au 1 janvier 0. From_days(nb jours) : renvoie la durée correspondant à un nombre de jours. Cette durée a le format date.from_days démarre à 366 : 1 janvier de l'année 1.

datediff(date1, date2) : nb jours entre 2 dates ;

date_add(date, **interval** nb ***type***) : ajout d'un nombre de mois, jours, heures, etc.(***type*** : year, month, day, week, hour, minute, second, microsecond, etc.) ;

+ interval nb ***type*** : augmente ou diminue directement une date, ou date et heure. Ne marche pas pour les heures seules ; exemple : select now + interval -1 month ;

extract (***type*** **from** date) : extraction d'une année, mois, jour, heure, etc. (***type*** : year, month, etc. + year_month, hour_minute, etc.)

utc_date : date ; **utc_time** : heure ; **utc_timestamp** : date et heure ; date et heure de greenwich (« universel »)

Exemples

Aujourd'hui

```
SELECT curdate();   //  2016-10-31
```

Le mois du jour

```
SELECT month(curdate());     // 10
```

Aujourd'hui, l'année dernière :

```
SELECT from_days(to_days(curdate())-365);
```

Nombre de jours entre deux dates

```
SELECT datediff('2016-10-31','2016-10-21');     // 10
```

Gestion des formats d'affichage : date_format ( )

http://dev.mysql.com/doc/refman/5.6/en/date-and-time-functions.html#function_date-format

Exemple

```
SELECT date_format(
    now( ),
    '%W %d %M %H heures %i minutes %s sec.'
);
```

Monday 31 October 10 heures 23 minutes 24 secSyntaxe

Formats

%k : heure sur 1 chiffre si possible ; %w : jour de 0 à 6, 0 pour dimanche ; %a : jour abrégé ; %e : numéro du jour du mois sur 1 chiffre si possible ; %m : mois sur deux chiffres ; %c : mois sur 1 chiffre si possible ; %Y : année sur 4 chiffres ; %y : année sur 2 chiffres.

Affichage au format français

http://dev.mysql.com/doc/refman/5.7/en/locale-support.html

Pour passer en français côté client :

SET lc_time_names = 'fr_FR';

Pour passer en américain côté client :

SET lc_time_names = 'en_US';

Pour afficher la langue côté client :

SELECT @@lc_time_names;

Pour afficher la langue côté serveur :

SELECT @@global.lc_time_names;

Pour modifier la langue côté serveur :

SET global lc_time_names = 'fr_FR';

Pour afficher la date en français :

```
SET lc_time_names = 'fr_FR';
SELECT date_format(
    now( ),
    '%W %d %M %H heures %i minutes %s sec.'
);
```

lundi 31 octobre 11 heures 45 minutes 04 sec.

```
SELECT date_format(now(),'%d/%m/%Y %Hh%imin%ss') ;
```

31/10/2016 11h48min04s

Ajout d'intervalles de temps

date + INTERVAL valeur YEAR

date + INTERVAL valeur HOUR

hour_second, microsecond, second, minute, hour, day, week, month, quarter, year, hour_second

Exemples:

```
SELECT now( ) + INTERVAL 1 YEAR
```

```
SELECT now( ) + INTERVAL 1 HOUR
```

Interval s'applique uniquement aux dates et dates et heures, pas aux heures seulement.

SELECT curtime( ) + INTERVAL 1 HOUR donne de mauvais résultats.

Quelques conversions

Date vers jours et réciproquement

to_days (date) : Nombre de jours depuis le 1er janvier 0

from_days (nb jours) : durée en année, mois, jour à partir d'un nombre de jour

Remarque: from_days démarre à 366 : 1 janvier de l'année 1 (from_days(0 à 365) vaut 0.

Heure vers seconde et réciproquement

Nombre de seconde depuis minuit : time_to_sec (heure)

Inverse : sec_to_time (nb secondes)

Conversion d'une date en chaine de caractères et réciproquement

Curdate( ) + 0 = 20161010 ; (2016 10 10)

Date(20161010) = 20 octobre 2016

On peut mettre 20161010 ou 20161010' ou "20161010"

Date(20161010)+10 = 20 octobre 2016

Conversion d'une heure en nombre et réciproquement

Curtime( ) + 0 = 161055 ;

SELECT time(3666) ; = 16 heures 10 minutes et 55 secondes

Conversions de date : Unix_timestamp

Date vers secondes et réciproquement

Nombre de secondes depuis le 1er janvier 1970 : unix_timestamp (date)

Inverse : from_unixtime(nb secondes)

Remarque : unix_timestamp ( ) ⇔ unix_timestamp (now())

Heure vers seconde et réciproquement

Nombre de seconde depuis minuit : time_to_sec (heure)

Inverse : sec_to_time (nb secondes)

Conversion d'une date en nombre et réciproquement

Curdate( ) + 0 = 20090220 ;

Date(20090220) = 20 février 2009

Conversion d'une heure en nombre et réciproquement

Curtime( ) + 0 = 161055 ;

Time(161055) = 16 heures 10 minutes et 55 secondes

Les dates dans Oracle

Dates et heures système

```
Select current_date from dual ; // date client
Select localetimestamp ; // date et heure client
Select systimestamp ; // date et heure serveur
```

Extraction d'une partie de la date ou de l'heure

Extract (*format* from *expression*)

Formats de base : year, month, day, hour, minute, second.

Exemple :

```
Select extract (year from current_date) from dual ;
```

Les opérateurs de traitement de chaîne

Les dates sont aussi considérées comme des chaînes de caractères. On peut donc leur appliquer tous les opérateurs de traitement de chaînes.

Conversion d'une chaîne en date : to_date

To_date (*chaîne*, *format*)

La fonction permet de transformer une chaîne de caractères en date selon le format proposé.

Le format utilise les parties : YYYY, MM, DD

Il existe plusieurs formats de date :

mm/dd/yyyy

dd.mm.yyyy

yyyy-mm-dd

Ces formats permettent de gérer les conversions de chaînes de caractères en date.

Plusieurs formats d'écriture des dates sont acceptés :

15 mars 2005

15 mar 2005

15/3/2005

15-3-2005

15-MAR-2005

Exemple :

```
Select to_date(current_date, 'DD-MM-YY') from dual ;
```

Current_date est ici considéré comme une chaîne

Conversion d'une date en texte : to_char

To_char (*date*, *format*)

La fonction permet de transformer un attribut date en une chaîne de caractère selon le format précisé. On pourra à cette occasion n'extraire, par exemple, que l'année.

Exemple :

```
Select to_char(current_date, 'DD-YYYY-MM') from dual ;
```

Current_date est ici considéré comme une date

Manuel de référence

http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/expressions007.htm#i1047500

http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/functions001.htm#i88893

Les dates dans PostgreSQL

Principales fonctions de traitement de dates

now() : renvoie la date et heure du moment

current_date : date du jour

current_time : heure du moment

extract (type from valeur) : le type peut être : century, year, month, etc.

exemple : select extract(century from now( )) ;

Les opérateurs de traitement de chaîne

Les dates sont aussi considérées comme des chaînes de caractères. On peut donc leur appliquer tous les opérateurs de traitement de chaînes.

Manuel de référence

http://www.postgresql.org/docs

Dans un moteur de recherche, chercher : postgresql docs extract

Le transtypage : conversion de type

Présentation

Les résultats produits par les différentes opérations sont typés, selon le typage classique : entier, réel, chaîne de caractères, etc.

Les types seront abordés plus précisément dans le chapitre sur le DDL.

Toutefois, on a la possibilité de changer le type d'un résultat ou d'une variable : c'est le transtypage.

Bien sur le transtypage n'est possible que s'il est logique : on ne peut pas transformer 'bonjour' en entier !

Les fonctions ou opérateurs de transtypage sont variables selon les SGBD.

Exemple MySQL

http://dev.mysql.com/doc/refman/5.6/en/cast-functions.html

cast

```
Select cast('19.4' as signed );          // 19
Select cast( 4/3 as decimal(2,1) );      // 1.3
```

convert

```
Select convert ( 4/3, decimal(3,2) );    // 1.33
Select convert ( 4/3, decimal(2,2) );    // 0.99 !!! ATTENTION !!!
Select convert ( 4/3, decimal(1,2) );    // ERREUR !!!
```

6. Présentation des résultats : tris, limites, mode page

Trier les résultats

Présentation

Les tuples d'une table sont présentés dans n'importe quel ordre. On peut choisir de les trier selon les valeurs de certains attributs avec la clause **order by**.

Syntaxe AR

Tres = Tri(*table* ; *liste d'attributs)*

Syntaxe SQL

La syntaxe du tri est la suivante :

```
SELECT liste_1 d'attributs  FROM table
order by liste_2 d'attributs  ;
```

Dans la liste 2 d'attributs triés, on peut préciser l'ordre du tri, ascendant ou descendant, par les mots-clés **asc** et **desc** qu'on fait suivre chaque attribut de tri. Par défaut, le tri est ascendant (il n'est donc pas nécessaire de préciser **asc**).

Attributs projetés

Pour que le résultat affiché soit compréhensible, on utilisera la forme syntaxique suivante :

```
SELECT liste_1 d'attributs , liste_2 d'attributs FROM table
order by liste_1 d'attributs ;
```

La liste des attributs projetés est donc la suivante, dans l'ordre

Tri - CP – (CS) – Demandés – (Restriction)

Ou bien :

CP – (CS) – Demandés – (Restriction) – Tri

Tri : attributs de tri : on les projette en premier pour rendre lisible le résultat. Ce choix est discutable : on peut aussi les mettre à la fin, ou juste après CP et CS.

Exemple

tous les employés classés par fonctions, salaires décroissants et ordre alphabétique de noms :

```
SELECT fonction, sal, nom, NE
FROM emp
ORDER BY fonction asc, sal desc, nom;
```

Remarque

On voit sur cet exemple que cet ordre d'affichage n'est pas forcément le meilleur : ici on préférerait projeter soit : fonction, sal, NE, nom; soit : NE, nom, fonction, sal.

Usage d'alias dans le tri

Un alias défini dans la projection peut être réutilisé dans le tri :

```
SELECT NE, nom, sal+ifnull(sal) as salaireTotal
FROM emp
ORDER BY salaireTotal;
```

Limiter le nombre de tuples du résultat

Présentation

Les SGBD offrent différentes solutions permettant de limiter le nombre de tuples projetés dans un select.

Limiter la projection aux 5 premiers éléments :

Solution MySQM et PostgreSQL :

```
SELECT NE, nom, sal FROM emp
limit 5;
```

Solution Oracle :

```
SELECT NE, nom, sal FROM emp
Where rownum < 5;
```

Récupération du minimum ou du maximum

La technique consiste à récupérer le minimum ou le maximum en triant et en ne gardant que le premier.

Solution MySQM et PostgreSQL :

Exemple : quel est l'employé qui a le plus haut salaire :

```
SELECT NE, nom, sal FROM emp
order by sal desc
limit 1 ;
```

A noter que cette technique n'est pas très pertinente puisqu'elle ne pemet pas d'afficher les ex aequo.

On verra une solution avec les fonctions de groupe et les requêtes imbriquées. La solution Oracle est plus complexe à mettre en œuvre avec le rownum.

Récupération des 10 suivants

Solution MySQL :

```
SELECT fonction, sal, nom, NE FROM emp
order by fonction asc, sal desc, nom
limit 10, 5;
```

Ici, on commence au 11ème et on va jusqu'au 15ème.

Autre solution :

```
SELECT fonction, sal, nom, NE FROM emp
order by fonction asc, sal desc, nom
limit 5 offset 10;
```

Tri aléatoire : order by rand()

Principe

On peut souhaiter sortir les résultats dans n'importe quel ordre

Exemple

Tous les employés avec tous leurs attributs classés aléatoirement :

Solution MySQL :

```
SELECT * FROM emp
order by rand();
```

Sélection d'un élément au hasard : order by rand() et limit

Exemple : un employé au hasard avec tous ses attributs :

Solution MySQL :

```
SELECT * FROM emp
order by rand() limit 1;
```

Afficher les résultats en mode « page »

Présentation

Un Select renvoie normalement une liste de résultat sous la forme d'un tableau avec un individu par ligne et plusieurs attributs pour chaque ligne.

Présentation « page » de mysql : \G

Mysql permet d'afficher les résultats en mode « page », avec un attribut par ligne.

Il faut placer un \G à la fin de la requête plutôt qu'un ;

Solution MySQL :

```
SELECT * FROM emp \G
```

4 - CALCULS STATISTIQUES EN SQL : LES FONCTIONS DE GROUPE ET LES AGREGATS

PRINCIPALES NOTIONS	
Fonction de groupe	**Sum()**
Count()	**Max(), Min(), Avg()**
Agrégat	**Group by**
Having	

1. Calculs statistiques : les fonctions de groupe

Calcul statistique élémentaire : les fonctions de groupe

Présentation

- Les fonctions de groupe sont des fonctions qui permettent de faire des calculs statistiques sur un ensemble de tuples.
- Les fonctions de groupe interviennent dans la projection : elles s'intéressent à la colonne projetée.

Attention !!!

- Une fonction de groupe est une fonction qui s'applique non pas à la valeur d'un attribut pour un tuple (comme la fonction sinus ou la fonction logarithme), mais à toutes les valeurs d'un attribut pour tous les tuples de la table traitée, donc à toute la colonne pour un attribut donné.

Exemples

Soit la table des Employés suivantes :

NE	Nom	Salaire
1	Toto	2000
2	Titi	1000
3	Tutu	4000
4	Tata	3000

Pour avoir les employés avec leur salaire, on écrit

```
SELECT NE, nom, salaire
FROM emp ;
```

Pour avoir le nombre d'employés, on obtient la table suivante :

nbEmployes
4

Pour obtenir ce résultat, on écrit :

```
SELECT count(*) as nbEmployes
FROM emp ;
```

Pour avoir la moyenne des salaires, on obtient la table suivante :

moySal
2500

Pour obtenir ce résultat, on écrit :

```
SELECT avg(sal) as moySal
FROM emp ;
```

Remarque importante

- Une fonction de calcul statistique (ici count() et avg() produit donc **un tuple et un seul comme résultat**.
- Elle permet donc de créer un nouvel attribut à partir d'un attribut en entrée.

Conséquences syntaxiques

On ne peut pas projeter d'autres attributs avec une fonction de groupe (sauf s'il y a un Group By : cf. suite du cours).

En effet une table comme :

moySal	nom
2500	Toto

Ne voudrait rien dire : quel nom mettre à côté de la moyenne des salaires ???

On ne doit donc pas écrire :

```
SELECT avg(sal) as moySal, nom
FROM emp ;
```

Plusieurs statistiques en même temps

```
SELECT count(*) as nbEmployes, avg(sal) as moySal
FROM emp ;
```

nbEmployes	moySal
4	2500

Les 5 fonctions de groupe standards

Il existe 5 fonctions de groupe (notées ***fdg*** [3] dans cet exposé) et leur syntaxe est la suivante :

count(), max(), min(), sum(), avg()

3 ***fdg*** (**f**onction **d**e **g**roupe) n'est pas un mot clé du SQL mais une expression de notre métalangage.

Syntaxe SQL

La syntaxe d'une projection de fonction de groupe est la suivante[4] :

```
SELECT fdg ( attribut ) FROM table  ;
```

La fonction count ( )

Présentation

count permet de compter le nombre de tuples renseignés (non NULL) d'une table pour le ou les attributs spécifiés.

Exemple

pour savoir combien il y a de salariés dans la société, on écrira :

```
SELECT count(*) FROM emp ;
```

Remarque 1

On peut aussi écrire :

```
SELECT count(NE) FROM emp ;
```

NE étant la clé primaire, NE ne peut pas être NULL et le résultat est le même qu'avec count(*).

Quand on veut compter toutes les lignes de la table, mieux vaut utiliser count(*).

A priori le count(*) est équivalent au count(id) car dès qu'il y a un attribut not null, le système peut savoir qu'il faut tout compter sans s'intéresser aux autres attributs. Cependant, comme pour toutes les questions d'optimisation, il faut regarder l'usage.

Usages spéciaux

Compter les valeurs non null

Combien y a-t-il de personnes qui sont susceptibles d'avoir une commission ?

```
SELECT count(comm) FROM emp ;
```

Compter les valeurs distinctes

Combien y a-t-il de fonctions différentes dans la société ?

```
SELECT count(distinct fonction) FROM emp ;
```

4 Remarques sur le métalangage utilisé : il ne prétend pas être parfaitement formel! Son objectif est d'associer pédagogie et rigueur formelle. Les mots clés du langage SQL sont en gras (**SELECT**). Les expressions générales sont en italiques (*liste d'attributs*). Les cas particuliers des exemples sont au format standard (NE, nom).

Les fonctions avg, sum, max et min

Présentation

avg, sum, max et min fournissent respectivement la moyenne, la somme, le maximum et le minimum d'un attribut donné :

Exemples

Quel est le salaire moyen de la société :

```
SELECT avg(sal) FROM emp ;
```

Quelle est la somme des salaires de la société :

```
SELECT sum(sal) FROM emp ;
```

Quels sont les salaires minimums et maximums de la société :

```
SELECT min(sal), max(sal) FROM emp ;
```

Autres fonctions de groupe

http://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html

Selon les SGBD, on peut trouver d'autres fonctions de groupe proposant des calculs statistiques plus élaborés : variance, médiane, etc.

variance( ), var_samp( ), std( ), std_samp( ) (écart-type)

Bit_and(), bit_or(), bir_xor

Clé primaire de la table résultat

Quand on a une fonction de groupe, la table résultat n'a pas la même clé primaire que la table de départ.

Quel est la clé primaire de la table résultat ?

L'information récupérée, par exemple le nombre d'employés, est un attribut de quoi ?

La ligne projetée est un attribut d'un objet correspondant à l'ensemble de la table de départ, ici « les employés » considéré comme un objet d'un autre ensemble à déterminer. Ici, il s'agit par exemple de l'ensemble des entreprises. On projette les caractéristiques d'une entreprise.

On pourrait imaginer avoir plusieurs entreprises, chacune ayant un nom, et le nombre d'employés caractériserait le nom de l'entreprise qui serait la clé primaire.

En final, on écrira : CP = Employés (le nom de la tables).

2. Calculs statistiques : agrégats ou group by

Les agrégats ou regroupement : le group by

Présentation de la clause group by

Principe

La clause group by permet de faire des regroupements par valeur possible d'un attribut et de faire des statistiques sur les regroupements ainsi définis.

Exemple

On souhaite connaître, pour chaque fonction, le nombre d'employés et le salaire moyen :

```
SELECT fonction, count(*), avg(sal)
FROM emp
GROUP BY fonction;
```

ou encore quels sont les salaires maximums de chaque département :

```
SELECT ND, max(sal) FROM emp
GROUP BY ND ;
```

On obtient plusieurs tuples avec une fonction de groupe. En effet, la clause group by permet d'appliquer des fonctions de groupes à des sous ensembles de la table de départ.

Syntaxe SQL

La syntaxe de la clause group by est la suivante :

```
4 : SELECT liste d'attributs , liste de  fdg(attribut)
1 : FROM table
2 : [WHERE ...]
3 : GROUP BY liste  d'attributs
5 : [ORDER BY ...]
;
```

WHERE et ORDER BY sont facultatifs.

En général, la liste d'attributs est la même dans le group by et dans le select.

Détail du déroulement d'un goup by

Le group by se divise en 4 étapes :

1) d'abord un "order by" : les tuples de la table de départ sont triés selon les valeurs croissantes de la liste des attributs du group by ;
2) Ensuite les calculs statistiques des fonctions de groupe : ces calculs sont appliqués aux sous-ensembles ayant la même valeur pour la liste d'attribut du group by ;
3) Ensuite on projette les attributs du group by avec un distinct et tirés.
4) Pour chaque ligne de la nouvelle table, on peut mettre le résultat des opérations statistiques.

Clé primaire d'un group by

La clé primaire d'un groupe by, c'est la **concaténation des attributs du group by**.

Elle peut aussi être constituée d'**une partie seulement des attributs du group by.**

Group by sans fonction de groupe : A EVITER !

Un group by sans fonctions de groupe c'est soit une restriction-projection primaire, soit une restriction-projection avec distinct à l'order by près.

Soit le group by suivant :

```
SELECT liste d'attributs  FROM table
GROUP BY liste d'attributs
ORDER BY liste d'attributs;
```

Si la liste d'attributs ne contient pas la clé primaire, la clause group by est équivalente à la clause distinct associée à la clause order by :

```
SELECT distinct liste d'attributs  FROM table ;
```

Si la liste contient la clé primaire, alors il n'y a pas de restriction (pas de distinct) et la clause group by est équivalente à la clause order by :

```
SELECT liste d'attributs  FROM table
ORDER BY liste d'attributs;
```

Dans tous les cas, il faut éviter les group by sans fonctions de groupe.

Restrictions supplémentaires : la clause having

La clause **having** permet d'ajouter une condition sur les résultats des fonctions de groupe associées à une clause group by:

```
SELECT liste d'attributs , liste de fdg(attribut )
FROM table
GROUP BY liste d'attributs
HAVING formule logique de sélection des fdg(attribut );
```

Quelles sont les fonctions (fonction) pour lesquelles travaillent plus de trois personnes :

```
SELECT fonction, count(*) FROM emp
GROUP BY fonction
HAVING count(*) >=3;
```

ou encore:

```
SELECT fonction FROM emp
GROUP BY fonction
HAVING count(*) >=3;
```

Tri après un group by

On peut aussi trier les résultats.

A noter que MySQL tri par défaut selon les attributs du group by.

La clause de tri est placée en dernier

Exemple

```
SELECT fonction, count(*) AS nb
FROM emp
GROUP BY fonction
HAVING count(*) >=3
ORDER BY count(*) desc;
```

Usage d'alias

Usage standard

On peut utiliser un alias dans la clause ORDER BY :

```
SELECT fonction, count(*) AS nb
FROM emp
GROUP BY fonction
HAVING count(*) >=3
ORDER BY nb desc;
```

Usage non standard – spécificité MySQL

On peut utiliser un alias dans la clause HAVING :

```
SELECT fonction, count(*) AS nb FROM emp
GROUP BY fonction
HAVING nb >=3;
```

Attention, c'est une spécificité MySQL qui ne fonctionne que si la variable système SQL_MODE ne contient pas ONLY_FULL_GROUP_BY.

Cet usage n'étant pas standard, il vaut mieux l'éviter.

Le fonctionnement du SQL_MODE est du ONLY_FULL_GROUP_BY est détaillé dans le cours sur les jointures.

Ordre de projection des attributs

Rappel : cas des requêtes sans fonction de groupe :

Tri - CP – (CS) – Demandés – (Restriction)

Ou

CP – (CS) – Demandés – (Restriction) - Tri

Cas des requêtes avec fonction de groupe

On prend plutôt la deuxième version (TRI à la fin)

En général : attributs du group by = CP + CS

D'où l'ordre de projection suivant :

GB - Demandés – (Restriction=Having) – Tri

3. Calculs statistiques : statistiques et agrégations avancées

Présentation

Les SGBD-R offrent tous des fonctions statistiques supplémentaires et aussi des clause qui permettent de faire des regroupements avancés.

Count (distinct attribut) et Group By --> tous les SGBD

Count (distinct attribut) dans un group by

Exemple

Combien d'employés et de départements par fonction

```
SELECT fonction, count(ne) nbEmp, count(distinct nd) nbDept
FROM emp
GROUP BY fonction;
```

fonction	nbEmp	nbDept
ANALYST	2	1
CLERK	4	3
MANAGER	3	3
PRESIDENT	1	1
SALESMAN	4	1

2 analyst répartis dans 2 départements

3 clerk (employé, commis) répartis dans 2 départements, etc.

MAX (SUM (SALAIRE) --> ORACLE

On peut appliquer une fonction de groupe sur les attributs de la table résultant d'un group by.

Valeur maximum des moyennes des salaires par département

```
SELECT MAX( AVG( sal)) FROM EMP GROUP BY deptno;

MAX(AVG(SAL))
-------------
       2387,5
```

Group concat --> MySQL

http://dev.mysql.com/doc/refman/5.0/fr/group-by-functions.html#function_group-concat

Pour une valeur du group by, on regroupe toutes les valeurs d'un attribut :

```
SELECT attributGB, group_concat(liste d'attributs)
FROM table
GROUP BY attributGB;
```

On peut trier et choisir son séparateur :

```
group_concat(attribut ORDER BY attribut separator ' et ')
```

Exemple

Liste des employés par fonction :

```
SELECT fonction, group_concat(ne ORDER BY ne)  AS LesEmployes
FROM emp
GROUP BY fonction;
```

fonction	LesEmployes
ANALYST	7788,7902
CLERK	7369,7876,7900,7934
MANAGER	7566,7698,7782
PRESIDENT	7839
SALESMAN	7499,7521,7654,7844

Fonction OVER() --> ORACLE, SQL Server

La fonction over() permet d'ajouter le résultat d'une fonction de groupe à une table.

Tous les employés avec leur salaire et la moyenne des salaires :

```
select empno, ename, sal, avg(sal) over()
from emp;

     EMPNO ENAME             SAL AVG(SAL)OVER()
---------- ---------- ---------- --------------
      7839 KING             5000     1988,33333
      7782 CLARK            2450     1988,33333
      7934 MILLER           1300     1988,33333
      7935 DUPONT            800     1988,33333
             etc...
```

C'est équivalent à un select dans le select :

```
select empno, ename, sal, deptno, (select avg(sal) from emp)
from emp
where deptno=10;
```

Ou à un select dans le from :

```
select empno, ename, sal, deptno, avgsal
from emp, (select avg(sal) avgsal from emp) t1
where deptno=10;
```

With Rollup --> MySQL

http://dev.mysql.com/doc/refman/5.0/fr/group-by-modifiers.html

Le with rollup permet de faire les totaux par catégorie après un group by.

```
SELECT ….
GROUP BY (au moins deux attributs)
WITH ROLLUP ;
```

Exemple

Les employés travaillent dans un département et ont une fonction. On veut connaître le nombre d'employés par fonction et par département avec le total par fonction.

```
SELECT fonction, nd, count(*) nb
FROM emp
GROUP BY fonction, nd
WITH ROLLUP ;
```

fonction	nd	nb
ANALYST	20	2
ANALYST	*NULL*	2
CLERK	10	1
CLERK	20	2
CLERK	30	1
CLERK	*NULL*	4
MANAGER	10	1
MANAGER	20	1
MANAGER	30	1
MANAGER	*NULL*	3
PRESIDENT	10	1
PRESIDENT	*NULL*	1
SALESMAN	30	4
SALESMAN	*NULL*	4
NULL	*NULL*	14

Les résultats présentent le nombre d'employés par fonction et n°de département et présentent, en gris, le total des employés par fonction.

La dernière ligne donne le total des employés concernés par la requêtes.

Equivalents ORACLE :

GROUP BY ROLLUP, GROUP BY GROUPING SETS, ROUP BY CUBE

Ces clauses fonctionnent avec plus ou moins d'attributs et font plus ou moins de totaux intermédiaires.

PIVOT --> ORACLE 11 G, SQL Server

Le PIVOT permet de produire des tableaux croisés avec deux critères de regroupement.
(Pour comprendre les tableaux croisés, utiliser la fonction tableaux croisés dynamiques d'Excel).

Exemple de résultat produit ORACLE :

On part d'une table des ventes avec un montant, une année et un pays :

idVentes	Montant	Pays	Année
1	1000	France	2015
2	150	France	2015
3	300	Allemagne	2015
4	250	Angleterre	2017
Etc.			

On veut un tableau de total des ventes par année et par pays

C'est un tableau croisé : on croise les années, en ligne, avec les pays, en colonne.

```
Select *
(Select ANNEE, PAYS, MONTANT
Pivot (SUM(MONTANT) for PAYS)
Order by ANNEE ;

ANNEE        FRANCE   ALLEMAGNE   ANGLETERRE
---------------------------------------------
2015         29825     26825        27825
2016         25825     22825        29825
2017         32825     29000        28825
```

5 - BILAN SYNTAXIQUE

Ordre des clauses du Select mono-table

```
4 : SELECT [DISTINCT] liste d'attributs
1 : FROM table
2 : [ WHERE formule de restriction ]
3 : [ GROUP BY liste d'attributs
5 : [ HAVING formule de restriction ] ]
6 : [ ORDER BY liste d'attributs ]
```

A noter que:

Il n'y a qu'un seul WHERE : mais l'expression logique peut être complexe en incluant des AND, des OR, etc.

C'est la même chose pour le HAVING. Toutefois, le HAVING ne peut être présent que s'il y a un GROUP BY

Ordre raisonnable de projection des attributs

TRI – CP – [CS] – Demandés – [Restriction]

Ou

CP – [CS] – Demandés – [Restriction] - TRI

La clé significative (CS) est facultative dans l'absolu. Toutefois, la seule présence d'une clé primaire (un numéro le plus souvent) est rarement suffisante pour rendre le résultat exploitable.

Les attributs de restriction sont facultatifs. On ne les projette que pour vérifier si les restrictions ont bien été prises en compte.

Les attributs de tri peuvent être mis devant ou derrière indifféremment, ou juste après CP et CS. L'objectif est de rendre lisible le résultat.

Ces règles de projection ne sont pas absolues : toutefois, elles permettent dans la grande majorité des cas d'obtenir une bonne lisibilité de la table résultant du Select.

A noter que parfois, on ne souhaite pas projeter la CP : si l'ensemble des attributs projetés est suffisant pour identifier correctement chaque ligne, la CP n'est pas nécessaire.

SGBD – SQL - TP : SELECT MONO-TABLE

Nom :

Fichier texte encodé en UTF8 - BOM

****

****

Présentation et préparation du travail

- Le sujet de TP est dans un fichier texte téléchargeable : MonNom_TP_SQL_monotable.txt
- Changez le nom du fichier : remplacez MonNom par votre nom : exemple : Liaudet_SQL_monotable.txt
- Dans le fichier, à la première ligne, mettez votre nom.
- Vous mettrez ensuite vos réponses directement dans le fichier.
- Il sera envoyé à l'adresse mail : liaudet.bertrand@wanadoo.fr

****

****

****

EXERCICE 1 : charger les tables de la base de données - EmployesTP01.sql

****

Présentation

On travaille sur la table suivante qui se trouve dans le fichier EmployesTP01.sql

EMP(**NE**, NOM, FONCTION, DATEMB, SAL, COMM, ND)

- NE numéro de l'employé. **Clé primaire.**
- NOM nom de l'employé.
- FONCTION intitulé du poste occupé.
- DATEMB date d'embauche.
- SAL salaire de l'employé.
- COMM commission (part de salaire variable).
- ND n° du département dans lequel travaille l'employé.

****

Les requêtes :

1. Téléchargez le script de création de la BD : EmployesTP01.sql
2. Lancez ce script de création de la BD.
3. Vérifiez que la table est créée (visuellement dans WAMP, ou avec la commande « desc »)
4. Affichez tous les tuples de la table (avec WAMP ou avec un select).

****
****
****

EXERCICE 2 : interrogation de la BD

****

Travail à faire

- Répondre à toutes les questions ci-dessous.
- Le fichier répond déjà aux 2 premières questions du TP : vous devez utiliser le même formalisme pour les autres questions.
- Pour chaque question vous devez mettre :
  a L'intitulé de la question
  b Le select de la réponse
  c La CP de la table résultat de chaque question

****

Priorité des questions

- Dans un premier temps, répondez uniquement aux questions en gras qui ne commencent pas par (*),
- Ensuite traitez les questions avec (*)

****

Attention !

- Vous ne devez projeter que les attributs nécessaires : CP, CS, AD, AR – Tri devant ou derrière (Clé Primaire, Clé Significative, Attributs Demandés, Attributs de restriction).
- Vous devez précisez la clé primaire de chaque table de résultats.

****

Outils

- Utilisez un éditeur efficace : notepad++, sublim text : qui fasse de la coloration syntaxique. Mettez une extension .sql à vos fichiers pour avoir la coloration syntaxique.

- Vous pouvez travailler dans la calculette mysql ou celle de phpMyAdmin.

****

Les requêtes

1. **Tous les employés avec tous leurs attributs**

   SELECT *

   FROM employes ;

   -- CP : NE

2. **Tous les employés**

   SELECT NE, nom

   FROM employes ;

   -- CP : NE

3. **Tous les employés triés par ordre alphabétique**
4. **Tous les employés triés par n° de département croissant, ordre alphabétique des fonctions, ancienneté décroissante**
5. (*) Tous les employés avec leurs salaires triés par salaire décroissant
6. (*) Tous les employés du département 30 avec tous leurs attributs
7. **Tous les employés du département 30 triés par ordre alphabétique**
8. **Tous les managers des départements 20 et 30**
9. (*) Tous les employés qui ne sont pas managers et qui sont embauchés en 1981
10. **Toutes les fonctions de la société**
11. **Tous les employés ne travaillant pas dans le département 30 et qui soit ont un salaire > à 2800, soit sont manager.**
12. (*) Tous les employés dont le salaire est compris entre 1000 et 2000
13. **Tous les numéros de département**
14. **Toutes les fonctions par département (10 : président, 10 : manager, etc.)**
15. **Tous les employés ayant ou pouvant avoir une commission**
16. (*) Tous les salaires, commissions et totaux (salaire + commission) des vendeurs
17. **Tous les salaires, commissions et totaux (salaire + commission) des employés**
18. (*) Tous les employés embauchés en 1981
19. **Tous les employés avec leur date d'embauche, la date du jour et leur nombre d'années d'ancienneté (on considérera que toute année commencée vaut pour une année), triés par ancienneté (on utilisera les fonctions de base de traitement de date et de traitement de chaîne).**
20. (*) Tous les employés ayant un A en troisième lettre de leurs noms.
21. **Tous les employés ayant au moins deux A dans leurs noms.**
22. (*) Donner les quatre dernières lettres du nom de chaque employé.
23. **Quel est le plus gros salaire de l'entreprise ? FONCTION MAX interdite.**
24. (*) Quel est le plus gros salaire total des vendeurs (SALESMAN) ? FONCTION MAX interdite.

25. (*) Quels sont les 3 employés qui ont le salaire le plus élevé ?
26. **Lister 3 employés au hasard**
27. (*) Afficher tous les employés en affichant : « anciens » pour ceux embauchés avant le 1 janvier 1982, rien pour ceux embauchés en 1982 et « nouveaux » pour ceux embauchés après le 1 janvier 1983. On utilisera deux méthodes : le case et le if. On tri par date d'embauche et par ordre alphabétique.
28. **Afficher les employés avec le numéro de leur tranche de salaire. Le numéro va de 1 à N. La première tranche va de 0 à 999, la deuxième de 1000 à 1999, la troisième de 2000 à 2999, etc. On considère qu'on ne sait pas à l'avance combien il y aura de tranche. On affichera les résultats par ordre de tranche croissante et par ordre alphabétique à l'intérieur d'une même tranche**.

****
****
****

EXERCICE 3 : interrogation de la BD

****

Travail à faire

La même chose que pour l'exercice 2, à la suite dans le même fichier.

****

Les requêtes

1. **Combien d'employés dans la société**
2. **Combien d'employés embauchés en 1981**
3. **Combien de vendeurs (« Salesman ») dans la société**
4. **Combien de départements dans la société**
5. **Combien de fonctions différents dans la société**
6. (*) Combien y a-t-il d'employés qui n'ont pas et ne peuvent pas avoir de commission ?
7. (*) Salaires minimum, maximum et moyen de la société.
8. **Salaires moyens des vendeurs**
9. **Salaires moyens de tous les employés en tenant compte des commissions**
10. (*) Pourcentage moyen de la commission des vendeurs par rapport à leur salaire
11. (*) Quel est le salaire moyen, les salaires min et max et le nombre d'employé par profession ?
12. **Quels sont les salaires maximums de chaque département ?**
13. (*) Quels sont les départements dans lesquels travaillent plus de deux personnes et quels sont les salaires moyens dans ces départements ?
14. **Quels sont les départements dans lequel il y a plus que 4 personnes ?**

15. Quels sont les fonctions pour lesquels la moyenne du salaire est supérieure à 2000 ?

16. Combien y a-t-il d'employés par département et par fonction et quelle est la moyenne de leurs salaires ?

17. (*) Quel est le nombre d'employés par année d'embauche ?

18. Combien y a-t-il d'employés par tranches de salaire de 1000 (0 à 999, 1000 à 1999, etc.).

19. (*) Combien d'employés et de départements par fonction. Vérifiez bien votre résultat : par exemple, il y a deux ANALYST, tous les deux dans le département 20, la réponse est donc: 2 employés et 1 département pour la fonction ANALYST.

20. Liste des employés par fonction.

21. (*) Liste des employés par départements.

22. Liste des fonctions par département.

23. (*) Liste des départements par fonction.

24. Quel est le nombre d'employés par département et par fonction, trié par département décroissant.

25. La même chose, mais trié par nombre d'employé et par numéro de département croissant. On fera une version avec alias.

****

****

APPENDICE

Gestion des accents avec MySQL

Si on a des problèmes d'accents, faire une show variables like '%char%';

Sous Windows

```
mysql> show variables like '%char%';
+--------------------------+------------------------------------+
| Variable_name            | Value                              |
+--------------------------+------------------------------------+
| character_set_client     | cp850                              |
| character_set_connection | cp850                              |
| character_set_database   | latin1                             |
| character_set_filesystem | binary                             |
| character_set_results    | cp850                              |
| character_set_server     | latin1                             |
| character_set_system     | utf8                               |
| character_sets_dir       |
                c:\wamp64\bin\mysql\mysql5.7.14\share\charsets\ |
+--------------------------+------------------------------------
   -----------+
8 rows in set, 1 warning (0.00 sec)
```

Sous Mac

```
mysql> show variables like '%char%';
```

```
+--------------------------+-------------------------------------+
| Variable_name            | Value                               |
+--------------------------+-------------------------------------+
| character_set_client     | utf8                                |
| character_set_connection | utf8                                |
| character_set_database   | latin1                              |
| character_set_filesystem | binary                              |
| character_set_results    | utf8                                |
| character_set_server     | latin1                              |
| character_set_system     | utf8                                |
| character_sets_dir       |
                      /Applications/MAMP/Library/share/charsets/ |
+--------------------------+-------------------------------------+
8 rows in set (0,00 sec)
```

Remplacement du character_set

Il se peut qu'il faille remplacer les character_set_client cp850 par de l'utf8.

Cela se fait par exemple avec la commande :

SET character_set_client = utf8;

http://dev.mysql.com/doc/refman/5.7/en/set-character-set.html

http://dev.mysql.com/doc/refman/5.7/en/charset-connection.html

Utilisation des commandes MySQL dans la calculette

eee et notee par exemple :

http://dev.mysql.com/doc/refman/5.7/en/mysql-commands.html