

Structured Query Language

PostgreSQL

Alexis NEDELEC

LISYC EA 3883 UBO-ENIB-ENSIETA
Centre Européen de Réalité Virtuelle
Ecole Nationale d'Ingénieurs de Brest

enib ©2007



Langage SQL

Historique

Langages pour décrire et manipuler des données relationnelles

- QUEL (INGRES, université de Berkeley)
- QBE, SQL (IBM)

Accès aux SGBD Relationnels

- Langage de Description de Données (LDD)
- Langage de Manipulation de Données (LMD)

Structured Query Language

SQL : Langage “universel” pour

- Concevoir, créer des ensembles de données (LDD)
- Manipuler des informations inter-dépendentes (LMD)

Normalisation SQL

ANSI/ISO (www.iso.org)

- SQL 1 (1986) : standard de description/manipulation
- SQL 2 (1992) : supporte totalement le modèle relationnel
- SQL 3 (1999) : doit supporter les modèles objets,
- SQL 4 (200 ?) : modèle déductif...

Objectifs principaux

- Création du schéma de la base (tables et associations)
- Recherche d'informations (jointures entre tables)
- Modification d'informations (manipulation des n-uplets)

Structure du langage

SQL Déclaratif : gestion de données

- DDL : Data Definition Language
- DML : Data Manipulation Language
- DCL : Data Control Language
- TCL : Transaction Control Language

SQL Procédural : traitement internes,externes

- PSM : Persistent Stored Module (triggers ...)
- CLI : Call Level Interface (ODBC ...)
- Embedded SQL : ordre SQL dans un langage hôte

Syntaxe SQL

Préciser **quoi** sans savoir **comment** y accéder

Opérations de bases

- **Recherche** (SELECT en SQL / RETRIEVE en QUEL)
- **Insertion** (INSERT en SQL / APPEND en QUEL)
- **Suppression** (DELETE en SQL / SUPPRESS en QUEL)
- **Modification** (UPDATE en SQL / REPLACE en QUEL)

Recherche d'informations en SQL

SELECT sur attributs de n-uplets	(projection, Π)
FROM relation(s)	(jointure, \bowtie)
WHERE qualification	(restriction, σ)

Syntaxe SQL/OQL

Recherche d'informations en OQL

```
SELECT sur proprietes (attributs) d'objets
FROM classe(s)
WHERE qualification (navigation classes, methodes)
```

Terminologie

	Relationnel	Objet
Relation	Table	Classe
Attribut	Colonne	Propriété
Occurrence	Ligne	Instance
Domaine	Type	Type élémentaire
Unicité	Clé Primaire	Identité d'Objet
Association	Clé Etrangère	Référence sur Classe

Langage de Définitions de Données (LDD)

Créations

- CREATE SCHEMA : Espace de nommage, création de droits ...
- CREATE TABLE : création d'une entité structurée
- CREATE VIEW : "encapsulation" de requêtes dans une vue
- ALTER, DROP TABLE : modification, destruction de tables
- CREATE INDEX : efficacité de recherche sur des colonnes
- CREATE TYPE, DOMAIN : création de types de données
- CREATE FUNCTION : codé en SQL procédural
- CREATE PROCEDURE : procédures stockées
- CREATE TRIGGER : déclencheurs sur modification de tables

Langage de Définitions de Données (LDD)

Contraintes de colonnes

- **DEFAULT** : valeur par défaut dans une colonne
- **NOT NULL** : contrainte de colonne non vide
- **UNIQUE** : pas de doublons dans une colonne
- **PRIMARY KEY** : clé primaire (sur 1 ou plusieurs colonnes)
- **FOREIGN KEY** : référence sur une colonne de table “mère”
- **CHECK** : contrainte de validité sur une colonne
 - **BETWEEN** : plage de valeurs
 - **LIKE, SIMILAR, -, %** : comparaison partielle
 - **IN** : liste de valeurs possibles
 - **CASE** : branchement sur des valeurs possibles
 - **OVERLAPS** : recouvrement de périodes

Types de données

Chaînes de caractères

- CHAR : taille fixe codé sur 1 octet (ASCII, EBCDIC)
- VARCHAR : taille variable codé sur 1 octet
- NCHAR : taille fixe codé sur 2 octets (Unicode)
- NVARCHAR : taille variable codé sur 2 octets (Unicode)
- CLOB, NCLOB : grande taille (Character Large Object)

Datation

- DATE : date (jour,mois,année)
- TIME : heure
- TIMESTAMP : date et heure
- TIME (TIMESTAMP) WITH TIME ZONE : avec fuseau horaire
- INTERVAL : intervalle de temps, durée

Types de données

Types entiers

- INTEGER : entier sur 32 bits (en principe)
- SMALLINT : entier sur 16 bits (< INTEGER)
- BIGINT : nouveauté SQL :2003 pour 64 bits (> INTEGER)

Types réels : approximation

- FLOAT : on peut fixer la précision
- REAL : sans fixer de précision
- DOUBLE PRECISION : plage de valeurs (> REAL)

Types décimaux : valeur exacte

- NUMERIC[(n[,p])] : sur n chiffres, p chiffres après la virgule
- DECIMAL[(n[,p])] : id NUMERIC, interne au SGBDR

Types de données

Types binaires

- **BOOLEAN** : booléen
- **BIT[(n)]** : chaîne de bits de taille fixe (n bits)
- **VARBIT[(n)]** : chaîne de bits de taille variable (n bits)
- **BLOB** : Binary Large Object (en K,M,G octets)

Opérateurs de comparaison

Op.	Numérique	Caractère	Date
<	inférieur	classé avant	plus tôt que
=	égal	équivalent à	en m temps que
>	supérieur	classé après	plus tard que
<=	inf. ou égal	classé avant ou équ.	pas plus tard que
<>	non égal	différent de	pas en m temps que
>=	sup. ou égal	classé après ou équ.	pas plus tôt que

Langage Manipulation de Données (LMD)

Modifications de données

- INSERT INTO ... VALUES : insertion d'enregistrements
- UPDATE ... SET ... WHERE : modification d'enregistrements
- DELETE FROM ... WHERE : destruction d'enregistrements

Recherche de données

```
SELECT [DISTINCT] <liste de colonnes>  
FROM <liste de tables>  
[WHERE <conditions de recherche>]  
[GROUP BY <partitionnement horizontal>]  
[HAVING <conditions de recherche sur les groupes>]
```

Recherche de données

Ordre d'exécution de requête

- 1 FROM : concaténation de chaque ligne de chaque table.
- 2 WHERE : élimination des lignes ne vérifiant pas les conditions (FALSE, UNKNOWN) sur la table de travail.
- 3 GROUP BY : répartition des lignes résultantes dans des groupes où les valeurs dans une même colonne sont identiques
- 4 HAVING : restriction des lignes vérifiant les conditions (TRUE) sur les regroupements.
- 5 SELECT : élimination des colonnes non mentionnées.
- 6 DISTINCT : élimination des lignes dupliquées.
NB : Si GROUP BY le DISTINCT est inutile.

Recherche de données

Résultat de recherche

- ALL : on garde tout
- DISTINCT : on élimine les doublons
- ORDER BY : tri de résultat
 - noms de colonnes : par ordre de “niveau de détails”
 - ASC, DESC : tri ascendant ou descendant par colonne
 - COLLATE : sensible à la casse, aux accents ... par colonne

Opérations ensemblistes booléennes

Entre résultats de deux requêtes

- UNION : concatène les 2 requêtes
- INTERSECT : les données communes aux 2 requêtes
- EXCEPT : les données qui ne sont pas dans l'autre requête

Recherche de données

Recherches imbriquées

- IN, =ANY : égalité avec un élément de la requête imbriquée
- NOT IN, <> ALL : : différent de tous les éléments
- EXISTS, NOT EXISTS : existence d'un élément

Jointures entre tables

- CROSS JOIN, NATURAL JOIN
- INNER JOIN, OUTER JOIN (LEFT, RIGHT, FULL)

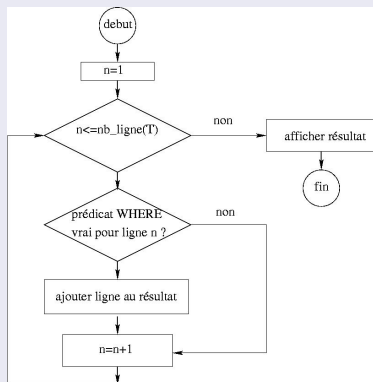
Fonctions de calcul

- COUNT, SUM : nombre, somme d'occurrences
- MIN, MAX, AVG : valeur min, max et moyenne
- STDEV_POP, STDEV_SAMP : écart-type de population, échantillon
- VAR_POP, VAR_SAMP : variance de population, échantillon

Recherche de données

Opérations sur une table

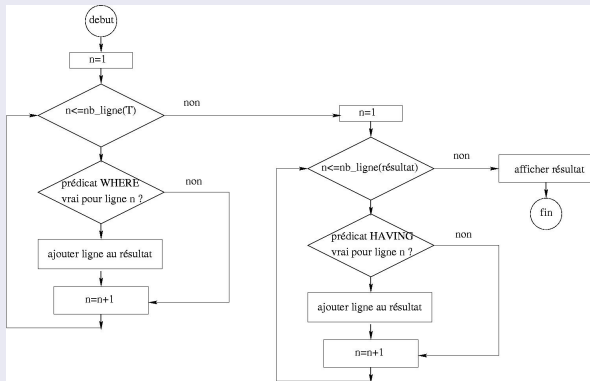
- SELECT : projection sur colonnes (Π)
- FROM : produit cartésien entre tables (\times)
- WHERE : restriction sur les enregistrements (σ)



Recherche de données

Opérations sur une table

- GROUP BY : partitionnement du résultat
- HAVING : restriction sur les partitionnements du résultat

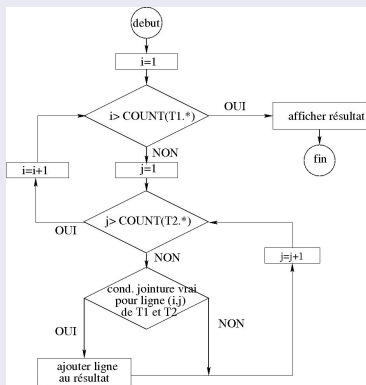


Recherche de données

Opérations sur deux tables

```

SELECT *                (résultat)
FROM T1 CROSS JOIN T2 (produit cartésien, jointure)
WHERE T1.a=T2.b         (condition de jointure)
  
```



Requêtes imbriquées

Retour d'une valeur

```
SELECT *  
FROM T1  
WHERE T1.a = (SELECT COUNT(T2.b)  
              FROM T2 )
```

Retour d'un ensemble de valeurs

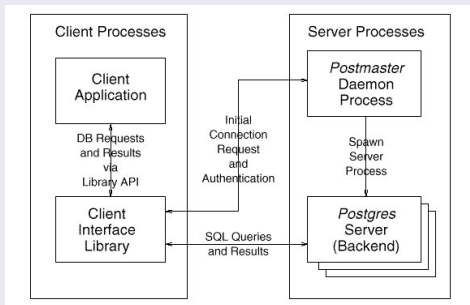
```
SELECT *  
FROM T1  
WHERE T1.a IN (SELECT T2.b  
              FROM T2 )
```

IN : appartenance (\in) à un ensemble

Client-Serveur (www.postgresql.org)

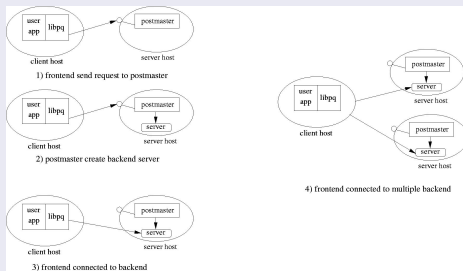
Modèle “un process par client”

- ① un démon “superviseur” : **postmaster**
- ② une application cliente, **front-end** (psql par exemple)
- ③ un ou plusieurs serveurs de BD, **back-end** (postgres)



Client-Server (www.postgresql.org)

Connexion Client-Server

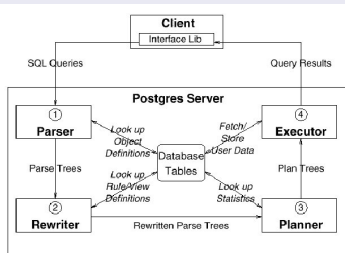


Interfaces de programmation d'applications (API)

- C (`libpq`), Embedded C (`ecpg`)
- C++ (`libpq++`), Java (JDBC), ODBC
- Python (`pygresql`), Tcl/Tk (`libpgtcl`)
- Perl (`pgsql_perl5`), ...

Client-Serveur (www.postgresql.org)

Traitement de Requête



Dictionnaire de données

- vérification syntaxique, arbre de requête (**Parser**)
- vérification de règles, transformation d'arbre (**Rewriter**)
- plan d'exécution, utilisation d'index (**Planner**)
- accès disque, récupération d'enregistrements (**Executor**)

Dictionnaire de Données

Méta Base de Données

Classes (tables) principales

- `pg_class` : description des tables de la base
- `pg_attribute` : un enregistrement par colonne de table
- `pg_index` : un enregistrement par index
- `pg_relcheck` : informations sur les contraintes de colonnes
- `pg_proc` : description des fonctions
- `pg_language` : langage d'implémentation des fonctions
- `pg_operator` : opérateurs utilisables dans des expressions
- `pg_type` : les types du dictionnaire (`CREATE TYPE`)

Création et connexion

Commandes PostgreSQL

```
{logname@hostname} createdb BdM
```

```
{logname@hostname} psql BdM
```

```
Welcome to the POSTGRESQL interactive sql monitor:
```

```
Please read the file COPYRIGHT ... POSTGRESQL
```

```
type \? for help on slash commands
```

```
type \q to quit
```

```
type \g or terminate with semicolon to execute query
```

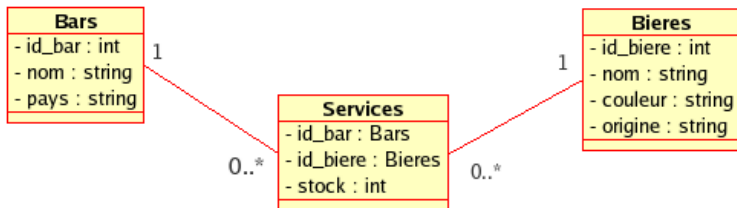
```
You are currently connected to the database: BdM
```

```
BdM=> SELECT * FROM pg_class;
```

```
...
```


Modèle de données de la Base

Services : Association(n-m) Bars-Bieres



Description (LDD) SQL

- CREATE SCHEMA : création (de schémas) de Bases de Données
- CREATE TABLE : création de tables
- CREATE VIEW : définitions de vues sur tables

Création de tables

Relation bars

```
CREATE TABLE bars (id_bar INTEGER NOT NULL
                    PRIMARY KEY,
                    bar  VARCHAR(20),
                    pays VARCHAR(20) );
```

Relation bieres

```
CREATE TABLE bieres (id_biere INTEGER NOT NULL
                     PRIMARY KEY,
                     biere  VARCHAR(20),
                     couleur VARCHAR(10),
                     origine VARCHAR(20) );
```

Création de tables

Relation services

```
CREATE TABLE services (id_bar INTEGER NOT NULL,  
                        id_biere INTEGER NOT NULL,  
                        stock     SMALLINT,  
                        PRIMARY KEY(id_bar,id_biere),  
                        FOREIGN KEY(id_bar)  
                            REFERENCES bars,  
                        FOREIGN KEY(id_biere)  
                            REFERENCES bieres);
```

Création de tables dans la base

```
BdM=>CREATE TABLE bars (...)  
BdM->;  
CREATE TABLE  
BdM=>
```

Insertion, mises à jour d'informations

Insertion d'enregistrements

```
BdM=>INSERT INTO bars VALUES (1,'Bar du Coin','France');  
INSERT ...  
BdM=>INSERT INTO bieres VALUES (1, Kronenbourg',  
BdM->'blonde', 'France');  
INSERT ...  
BdM=> INSERT INTO services VALUES (1,2,1000);  
INSERT ...
```

Mises à jour d'enregistrements

```
UPDATE services  
SET stock = 2*stock  
WHERE stock=1000;
```

Etat de la Base de Données

Relations bieres

bieres			
id_biere	biere	couleur	origine
1	Kronenbourg	Blonde	France
2	Guinness	Brune	Irlande
3	Heineken	Blonde	Hollande
4	Newcastle	Rousse	UK
5	Spaten	Blonde	Allemagne
6	Bush	Blonde	USA
7	Kanterbrau	Blonde	France
8	Kronenbourg	Brune	France

Etat de la Base de Données

Relations bars

bars		
id_bar	bar	pays
1	Bar du Coin	France
2	Corners Pub	USA
3	Cafe der Ecke	Allemagne
4	Cafe des Amis	France

Etat de la Base de Données

Relations services

services		
id_bar	id_biere	stock
1	1	1000
1	2	250
1	3	50
1	4	10
2	1	100
2	6	1500
3	5	5000

Récupération d'informations

Projection : $\Pi_{id_bar}(services)$

```
SELECT id_bar
FROM services;
```

```
id_bar
-----
  1
  1
  1
  1
  2
  2
  3
(7 rows)
```

```
SELECT DISTINCT id_bar
FROM services;
```

```
id_bar
-----
  1
  2
  3
(3 rows)
```


Filtrage d'informations

Restriction : $\sigma_{(origine='France')}(bieres)$

```
SELECT *  
FROM bieres  
WHERE origine='France';
```

id_biere	biere	couleur	origine
1	Kronenbourg	Blonde	France
7	Kanterbrau	Blonde	France
8	Kronenbourg	Brune	France

(3 rows)

Récupération et filtrage d'informations

Projection et Restriction : $\Pi_{biere, couleur}(\sigma_{(origine='France')}(bieres))$

```
SELECT biere, couleur
FROM bieres
WHERE origine='France';
```

biere		couleur
Kronenbourg		Blonde
Kanterbrau		Blonde
Kronenbourg		Brune

(3 rows)

Filtrage d'informations : connecteurs logiques

Restriction : $\sigma_{(origine='France' \wedge couleur='Brune')}(bieres)$

```
SELECT *
FROM bieres
WHERE origine='France' AND couleur='Brune';
```

id_biere	biere	couleur	origine
8	Kronenbourg	Brune	France

(1 row)

Connecteurs logiques : priorité

$$\Pi_{biere,couleur}(\sigma_{(origine='France' \wedge couleur='Brune' \vee couleur='Blonde')}(bieres))$$

```
SELECT biere, couleur
FROM bieres
WHERE origine='France'
      AND couleur='Brune' OR couleur='Blonde';
```

biere		couleur
-----	+	-----
Kronenbourg		Blonde
Heineken		Blonde
Spaten		Blonde
Bush		Blonde
Kanterbrau		Blonde
Kronenbourg		Brune

(6 rows)

Connecteurs logiques : priorité

$$\Pi_{biere, couleur}(\sigma_{(origine='France' \vee couleur='Brune' \wedge couleur='Blonde')}(bieres))$$

```
SELECT biere, couleur
FROM bieres
WHERE origine='France'
      OR couleur='Brune' AND couleur='Blonde';
```

biere	couleur
-----+-----	
Kronenbourg	Blonde
Kanterbrau	Blonde
Kronenbourg	Brune
(3 rows)	

Recherche sur plusieurs tables

Produit cartésien (CROSS JOIN) : \times (*bars*, *bieres*)

```
SELECT pays, origine
```

```
FROM bars, bieres;
```

```
-- FROM bars CROSS JOIN bieres
```

```
pays      | origine
```

```
-----+-----
```

```
France   | France
```

```
France   | Irlande
```

```
France   | Hollande
```

```
France   | UK
```

```
...      | ...
```

```
France   | France
```

```
USA      | France
```

```
...      | ...
```

```
(32 rows)
```

Recherche sur plusieurs tables

Jointure Naturelle : $\bowtie_{(ba.id_bar=s.id_bar)} (bars, services)$

```
SELECT *
FROM bars ba, services s
WHERE ba.id_bar = s.id_bar;
```

id_bar	bar	pays	id_bar	id_biere	stock
1	Bar du Coin	France	1	4	10
1	Bar du Coin	France	1	3	50
1	Bar du Coin	France	1	2	250
1	Bar du Coin	France	1	1	1000
2	CornersPub	USA	2	6	1500
2	CornersPub	USA	2	1	100
3	Cafe der Ecke	Allemagne	3	5	5000

(7 rows)

Recherche sur plusieurs tables

```
⋈(s.id_biere=bi.id_biere) (⋈(ba.id_bar=s.id_bar) (bars, services), bieres)
```

```
SELECT *  
FROM bars ba, services s, bieres bi  
WHERE ba.id_bar=s.id_bar AND s.id_biere=bi.id_biere
```

Résultat sous forme de colonnes :

```
(id_bar, bar, pays,  
id_bar, id_biere, stock,  
id_biere, biere, couleur, origine)
```

```
SELECT *  
FROM bars NATURAL JOIN services NATURAL JOIN bieres
```

Résultat sous forme de colonnes :

```
(id_biere, id_bar, bar, pays, stock, biere, couleur, origine)
```


Recherche sur plusieurs table

Création de vues

```
CREATE VIEW BarDuMonde AS
  SELECT bar, pays, stock, biere, couleur, origine
  FROM bars NATURAL JOIN services NATURAL JOIN bieres;
SELECT * FROM BarDuMonde
```

bar	pays	stock	biere	couleur	origine
Bar du Coin	France	1000	Kro...	Blonde	France
Bar du Coin	France	250	Guinness	Brune	Irlande
Bar du Coin	France	50	Heineken	Blonde	Hollande
Bar du Coin	France	10	Newcastle	Rousse	UK
Corners Pub	USA	100	Kro...	Blonde	France
Corners Pub	USA	1500	Bush	Blonde	USA
Cafe der Ecke	All...	5000	Spaten	Blonde	Allemagne

(7 rows)

Opérations ensemblistes booléennes

Opérateurs de l'algèbre relationnelle

- $T(X) = R(X) \cup S(X) = \{ \langle x \rangle \mid \langle x \rangle \in R \vee \langle x \rangle \in S \}$
- $T(X) = R(X) \cap S(X) = \{ \langle x \rangle \mid \langle x \rangle \in R \wedge \langle x \rangle \in S \}$
- $T(X) = R(X) - S(X) = \{ \langle x \rangle \mid \langle x \rangle \in R \wedge \langle x \rangle \notin S \}$

Opérateurs SQL

- UNION : concaténation de deux requêtes
- INTERSECT : données communes aux 2 requêtes
- EXCEPT : données ne concernant pas la 2ème requête

DISTINCT (par défaut) : tri avec élimination des doublons

Union (\cup)

Concaténation et tri de deux requêtes

Nom des pays ayant des bars et/ou des bières

```
(SELECT pays FROM bars)
UNION
(SELECT origine FROM bieres) ;
    pays
```

Allemagne

France

Hollande

Irlande

UK

USA

(6 lignes)

Union (\cup)

Concaténation sans tri de deux requêtes

```
(SELECT pays FROM bars)  
UNION ALL  
(SELECT origine FROM bieres) ;
```

Concaténation de deux requêtes triées

```
(SELECT DISTINCT pays FROM bars)  
UNION ALL  
(SELECT DISTINCT origine FROM bieres) ;
```

Intersection (\cap)

Données communes à deux requêtes

Nom des pays ayant des bars et fabriquant de la bière

```
(SELECT pays FROM bars)
INTERSECT
(SELECT origine FROM bieres) ;
    pays
```

```
-----
Allemagne
France
USA
(3 lignes)
```

Différence (−)

Données d'une requête sans correspondant dans l'autre

Nom des pays fabriquant de la bière et n'ayant pas de bars

```
(SELECT origine FROM bieres)
```

```
EXCEPT
```

```
(SELECT pays FROM bars);
```

```
origine
```

```
-----
```

```
Hollande
```

```
Irlande
```

```
UK
```

```
(3 lignes)
```

Appartenance (\in) à un ensemble

IN : Données communes à deux requêtes

Nom des pays ayant des bars et fabriquant de la bière

$$T(X) = R(X) \cap S(X) = \{ \langle x \rangle \mid \langle x \rangle \in R \wedge \langle x \rangle \in S \}$$

```
SELECT pays
FROM bars
WHERE pays IN (SELECT origine
               FROM bieres);
```

pays

France

USA

Allemagne

France

Appartenance (\in) à un ensemble

IN ou =ANY

Nom des pays ayant des bars et fabricant de la bière

```
SELECT pays
FROM bars
WHERE pays=ANY(SELECT origine
                FROM bieres);
```

Test d'égalité sur une des valeurs d'enregistrements :

```
SELECT pays
FROM bars
WHERE pays='France'
      OR pays='...' OR pays='...' OR ..
      OR ...
```


Appartenance (\in) à un ensemble

IN ou EXISTS

Nom des pays ayant des bars et fabricant de la bière

```
SELECT pays
FROM bars
WHERE EXISTS (SELECT *
              FROM bieres
              WHERE pays=origine);
```

Vérifier l'existence : sous-requête corrélée (pays=origine)

Requête imbriquée ou jointure ?

```
SELECT DISTINCT pays
FROM bars, bieres
WHERE pays = origine;
```

Non-appartenance (\notin) à un ensemble

NOT IN : Données ne concernant pas la 2ème requête

Nom des pays fabriquant de la bière et n'ayant pas de bars

$$T(X) = R(X) - S(X) = \{ \langle x \rangle \mid \langle x \rangle \in R \wedge \langle x \rangle \notin S \}$$

```
SELECT origine
FROM bieres
WHERE origine NOT IN (SELECT pays
                      FROM bars );
```

origine

Irlande

Hollande

UK

(3 lignes)

Non-appartenance (\notin) à un ensemble

NOT IN ou <>ALL

Nom des pays fabriquant de la bière et n'ayant pas de bars

```
SELECT origine
FROM bieres
WHERE origine <> ALL (SELECT pays
                      FROM bars );
```

Test de **non-égalité sur toutes les valeurs** d'enregistrements :

```
SELECT origine
FROM bieres
WHERE origine <> 'France'
      AND origine <> '...' AND origine <> '...' AND ...
      AND ...
```

Non-appartenance (\notin) à un ensemble

NOT IN ou NOT EXISTS

Nom des pays fabriquant de la bière et n'ayant pas de bars

```
SELECT origine
FROM bieres
WHERE NOT EXISTS (SELECT *
                  FROM bars
                  WHERE pays=origine);
```

Vérifier la non-existence : sous-requête corrélée (pays=origine)

Requête imbriquée ou jointure ?

```
SELECT DISTINCT origine
FROM bieres bi, bars ba
WHERE bi.origine <> ba.pays;
```

Calculs et groupement

Fonctions de calcul sur un ensemble

- COUNT, SUM : nombre, somme d'occurrences
- MIN, MAX, AVG : valeur min, max et moyenne
- STDEV_POP, STDEV_SAMP : écart-type de population, échantillon
- VAR_POP, VAR_SAMP : variance de population, échantillon

Regroupement : partitionnement horizontal

- GROUP BY : créer des sous-ensembles de l'ensemble

Conditions d'applications des calculs

- BLOB, CLOB, NCLOB : calculs inapplicables
- NULL : non-comptabilisés dans les calculs

Calculs et groupement

Calcul sur un ensemble

```
SELECT COUNT(bar) FROM BarDuMonde;
```

```
count
```

```
-----
```

```
7
```

```
(1 ligne)
```

Calcul sur les sous-ensembles d'un ensemble

```
SELECT pays, COUNT(bar) FROM BarDuMonde GROUP by pays
```

```
pays | count
```

```
-----+-----
```

```
Allemagne | 1
```

```
USA | 2
```

```
France | 4
```

```
(3 lignes)
```

Calculs et groupement

Portée des fonctions de calculs

Noms des bars et de leur quantité de Kronenbourg en stock supérieur au minimum des stocks de Kronenbourg

```
SELECT bar, stock
FROM BarDuMonde
WHERE biere='Kronenbourg'
AND stock > (SELECT MIN(stock)
              FROM services NATURAL JOIN bieres
              WHERE biere='Kronenbourg');
```

```
bar | stock
-----+-----
Bar du Coin | 1000
(1 ligne)
```

Calculs et groupement

Portée des fonctions de calculs

Nom et stock de bières des bars fournissant au moins 2 services en France ou en Allemagne

```
SELECT pays, bar, SUM(stock)
FROM services NATURAL JOIN bars
WHERE pays='France' OR pays='Allemagne'
GROUP BY pays, bar
HAVING COUNT(services.id_biere) > 1
```

pays	bar	sum
France	Bar du Coin	1310

(1 ligne)

Gestion du Temps

Types Date, Time, Timestamp

```
SELECT CURRENT_DATE AS today_is;  
    today_is
```

```
-----  
2007-04-02
```

```
SELECT CURRENT_TIME AS time_is;  
    time_is
```

```
-----  
15:25:14.266754+02
```

```
SELECT CURRENT_TIMESTAMP AS timestamp_is;  
    timestamp_is
```

```
-----  
2007-04-02 15:26:09.228392+02
```

Gestion du Temps

ALTER TABLE : modification de relation

```
=>ALTER TABLE services  
->ADD COLUMN depot DATE DEFAULT CURRENT_DATE;
```

```
ALTER TABLE
```

```
=>SELECT * FROM services;
```

id_bar	id_biere	stock	depot
1	1	1000	2007-04-02
1	2	250	2007-04-02
1	3	50	2007-04-02
1	4	10	2007-04-02
2	1	100	2007-04-02
2	6	1500	2007-04-02
3	5	5000	2007-04-02

(7 lignes)

Création d'utilisateurs

utilisateurs et droits

```
=>CREATE GROUP A3S2;  
CREATE GROUP  
=>CREATE USER A3S2B1 WITH PASSWORD 'A3S2B1'  
->NOCREATEDB NOCREATEUSER IN GROUP A3S2;  
CREATE USER
```

Modification des droits utilisateurs

```
=>ALTER USER A3S2B1 CREATEDB  
ALTER USER  
=>GRANT SELECT ON services TO A3S2B1;  
GRANT
```

Transactions utilisateurs

Transaction et verrouillage de tables

```
=>BEGIN TRANSACTION;  
BEGIN  
=>LOCK TABLE services IN ACCESS SHARE MODE;  
LOCK TABLE
```

Modification de table dans une transaction

```
=>INSERT INTO services VALUES (4,2,10000);  
INSERT 17297 1
```

Validation de transaction

- COMMIT : nouvel état de la BD
- ROLLBACK : état de la BD avant lancement de transaction

Le langage SQL

Intérêts

- Langage de Description de Données
- Langage de Manipulation de Données
- Langage standard pour les Bases de Données Relationnelles
- normalisation successives (SQL1, SQL2, SQL3)

Inconvénients

- pauvreté de la modélisation
- pauvreté de la représentation
- syntaxe lourde, optimisations

PostgreSQL

- SGBDR libre et en constante évolution

Bibliographie

Livres

- **A. G. TAYLOR** : “SQL pour les Nuls”
- **F. BROUARD, C.SOUTOU** : “SQL”
Collection Synthex, Pearson Education (2005)
- **C.SOUTOU** : “de UML à SQL : conception de bases de données”
Éditions Eyrolles(2002)
- **J. CELKO** : “SQL Avancé”
International Thomson Publishing (1997)
- **S. MARIEL** : “PostgreSQL : Services Web avec PostgreSQL et PHP/XML”
Les cahiers du programmeur, Éditions Eyrolles.(2002)

Bibliographie

Adresses “au Net”

- www.postgresql.org : le site officiel
- www.lamsade.dauphine.fr/~manouvri : Maude Manouvrier (www.librecours.org)
- sqlpro.developpez.com/biblio/SQL_bibl.html : des références SQL
- georges.gardarin.free.fr : le site de Georges Gardarin
- www.developpez.com : entre autre du SQL et des SGBD ...