Initiation à SQL

SQL

- Langage de base de données relationnelles
- Développé chez IBM (1970-80)
- Devenu une norme (ANSI/ISO) en 1986
- A la fois LDD (Langage de Définition de Données) et LMD (Langage de Manipulation de Données)
- Toute interface SQL à un SGBD est une adaptation de la norme à ce SGBD
- Utilisable en mode interactif comme dans un langage de programmation
- Langage assertionnel (non procédural) : on décrit les caractéristiques des données recherchées et non le chemin d'accès
- ici, ORACLE SQL versus norme SQL.

Les standards SQL

- SQL8
- SQL89
- SQL92 ou SQL2 (3 niveaux : entrée / intermédiaire / plein)
- SQL99 ou SQL3

SQL ORACLE

Définition des Données

Ce que décrit SQL

- Niveaux :
  - Logique : tables domaines et attributs
  - Externe : vues et privilèges
  - Interne : rien mais tous les SGBD ont CREATE INDEX

Logique SQL

- Environnement SQL
  - Catalogue C1
    - Schéma
      - Table
  - 1 catalogue et 1 schéma par défaut
- Oracle n'a pas intégré la notion de catalogue
- La norme SQL intègre le concept de métabase mais les règles sont peu suivies par les règles des éditeurs de SGBD

SQL ORACLE
Définition des Données

- permet de
  - créer,
  - modifier,
  - supprimer,
  - renommer,
- les éléments du schéma d'une base de données :
  - les tables,
  - les vues,
  - les index.

1. Les tables

- ce sont les relations du schéma relationnel

1.1. Création :

```
CREATE TABLE nomtable
( nomcol1 typecol1 [contraintecol1],
  nomcol2 typecol2 [contraintecol2],
  ...,
  contraintetable1,
  contraintetable2,...);
```

1. Les tables (suite)

Exemple :

FOURNISSEUR
(F#,FNOM,STATUT,VILLE)

```
CREATE TABLE FOURNISSEUR
                (F# CHAR(5) NOT NULL UNIQUE,
                 FNOM CHAR(20),
                 STATUT NUMBER(3) DEFAULT 10,
                 VILLE CHAR(15));
```


Types de données

- *caractères :* CHAR ou VARCHAR
  au maximum 255

  exemple : NOM CHAR (15)

- *numériques :* NUMBER(précision,échelle)

  (norme : DECIMAL, REAL, FLOAT,INTEGER)

  exemple : NUMBER (8,2)
  8 chiffres dont 2 après la virgule maximum

Types de données (suite)

- *dates :* DATE
  - * hors norme
  - * format standard : DD-MON-YY
  - * stocke : siècle, année, mois, jour, heure, minutes et secondes
  - * fonctions de conversion de format avec masques

- *d'autres types hors norme :* LONG, RAW
  - * RAW binaire
    chaîne d'octet de longueur variable manipulée sous forme hexadécimale
  - * LONG chaîne longue
    maximum 64 K
    1 seule par table
    inutilisable dans les expressions, les prédicats et dans les tris

Types de données (suite)

- Données de grande taille (SQL3) :
  BLOB Binary Large Object
  CLOB Character Large Object

- Types définis par l' utilisateur (SQL3) :
  CREATE TYPE

Contraintes

1°) Colonnes obligatoires : nomcol type NOT NULL
exemple : CREATE TABLE FOURNISSEUR(... ,
F# NOT NULL CONSTRAINT NN_CNT);

* "CONSTRAINT nom" permet de nommer la contrainte

2°) Unicité d'une colonne : nomcol type UNIQUE
exemple : F# NOT NULL UNIQUE

* Il faut que la colonne soit NOT NULL
* Il faut que la colonne ne soit pas une clé primaire

Contraintes (suite)

3°) Unicité de plusieurs colonnes :
UNIQUE (nomcol1,nomcol2,...)
exemple : UNIQUE (FNOM,VILLE)

4°) Clé primaire : nomcol type PRIMARY KEY
ou PRIMARY KEY (nomcol1,nomcol2,..)

* 1 seule par table, il faut que la colonne soit NOT NULL
* sera comparée à la clé étrangère par la contrainte référentielle

Contraintes (suite)

5°) Contrainte référentielle :

nomcol REFERENCES nomtable (nomcol)

ou FOREIGN KEY (listecolonnes) REFERENCES table(listecolonnes)

* la (ou les) colonne référencée doit être clé primaire ou colonne unique dans l'autre table.

6°) Contrainte sémantique : CHECK condition

* compare les colonnes d'une même table

exemple : CHECK (AGE BETWEEN 7 AND 77)

1. Les tables (suite)

1.2. Modification :

ALTER TABLE nomtable modification;

* permet de modifier la structure d'une table

Modifications autorisées :

a) ajouter une colonne :

ADD nomcol type contrainte

exemple : ALTER TABLE FOURNISSEUR

ADD PAYS CHAR(15);

1. Les tables (suite)

1.2. Modification :

b) ajouter une contrainte : ADD contrainte

exemple : ALTER TABLE FOURNISSEUR ADD UNIQUE (FNOM,VILLE);

c) modifier la définition d'une colonne :

MODIFY nomcol modif

- modifier la taille, le type (pour modifier le type ou réduire la taille, il faut que toute la colonne soit vide)
- ajouter NOT NULL (s'il n'y a pas déjà des valeurs nulles)

exemple : ALTER TABLE FOURNISSEUR MODIFY VILLE CHAR(20);

1. Les tables (suite)

1.2. Modification :

d) supprimer des contraintes nommées :

DROP CONSTRAINT nomcontrainte

e) supprimer des colonnes :

DROP nomcolonne

1. Les tables (suite)

1.3. Suppression :

DROP TABLE nomtable [RESTRICT|CASCADE];

exemple : DROP TABLE FOURNISSEUR;

* on ne peut pas supprimer une table créée par un autre utilisateur sauf le DBA
* DROP nomtable supprime aussi les index sur la table
- DROP nomtable ne supprime pas les vues associées à cette table mais les rend bien sûr indisponibles
- RESTRICT bloque si une FOREIGN KEY ou un trigger dépend de cette table
- CASCADE supprime aussi tout ce qui dépend
- RESTRICT par défaut

1. Les tables (suite)

1.4. Renommage :

RENAME anciennom TO nouveaunom;

exemple : RENAME FOURNISSEUR TO F;

2. Les vues

- ce sont des tables virtuelles
- ce sont des questions stockées
- permet le contrôle des accès
- permet l'indépendance logique des données (plusieurs vues d'un même ensemble de données)
- Matérialisation des vues dans le concept des entrepôts de données

2. Les vues

2.1. Création :

CREATE VIEW nomvue [alias1,alias2,...] AS question;

* les alias permettent de renommer les colonnes de la table dans la vue

exemple : Fournisseurs de Londres

CREATE VIEW FOURN_LOND AS SELECT F#,FNOM,STATUT FROM FOURNISSEUR WHERE VILLE="Londres";

* pas de clause ORDER BY

2. Les vues (suite)

2.2. Suppression : DROP VIEW nomvue;

exemple : DROP VIEW FOURN_LOND;

Utilisation des vues

- En interrogation, on procède comme avec une table réelle : l'ordre SQL de création de la vue est ré-exécuté à chaque référence à la vue.

  ⚠ ***Performances !***

- En mise à jour, la vue est utilisable uniquement si :
  * le SELECT définissant la vue ne comporte pas de jointure, ni de GROUP BY
  * toutes les colonnes de la table définies en NOT NULL sont dans la vue
  * les colonnes du SELECT ne sont pas des expressions
- Si la vue doit être utilisée en mise à jour, on peut ajouter WITH CHECK OPTION à la fin de CREATE pour contrôler que les modifications respectent la définition de la vue

3. Les index

* hors norme
* pour accélérer l'accès aux données
* pour garantir l'unicité de certaines données
* peuvent être créés sur une ou plusieurs colonnes
* les index sont créés par l'utilisateur (ou plutôt le DBA) mais ils n'ont pas à être référencés lors de la manipulation, c'est l'optimiseur qui se charge de les utiliser, le cas échéant
* index ≠ clé : index = physique / clé = logique
* un index peut être unique ou non unique
* un index permet d'accélérer les interrogations, mais peut pénaliser les mises à jour
* structure interne : principalement B-trees

3. Les index (suite)

3.1. *Création :*

CREATE [UNIQUE] INDEX nomindex ON table (nomcol1 [ASC/DESC], nomcol2 [ASC/DESC], ...);

* *UNIQUE :* 2 lignes de la table indexée ne peuvent pas prendre la même valeur pour le champ d'indexation

* *exemples :* FOURNISSEUR(F#,...,VILLE) PRODUIT(P#,....) COMMANDE(F#,P#,DATE,...)

CREATE UNIQUE INDEX I1 ON FOURNISSEUR(F#);

CREATE INDEX I2 ON FOURNISSEUR(VILLE);

non unique car plusieurs fournisseurs peuvent être localisés dans la même ville

CREATE UNIQUE INDEX I3 ON PRODUIT(P#);

CREATE INDEX I4 ON COMMANDE(F#,P#,DATE DESC);

non unique car un même produit peut être commandé plusieurs fois le même jour (par différents clients)

3. Les index (suite)

3.2. Suppression : DROP INDEX nomindex;

exemple : DROP INDEX I4;

Utilisation des index

- *en pratique, on crée des index sur :*
  - les colonnes servant souvent de critère de recherche (clé d'accès)
  - les colonnes de jointure
  - les colonnes identifiantes
- *on les évite pour :*
  - les colonnes souvent modifiées
  - les colonnes contenant peu de valeurs distinctes

SQL ORACLE

Manipulation des Données

INTERROGATION :

SELECT

MISE A JOUR :

INSERT	ajout
UPDATE	modification
DELETE	suppression

Schéma relationnel servant pour les exemples

Table FOURNISSEUR :

F(FNO,FNOM,STATUT,VILLE)

Table PIECE :

P(PNO,PNOM,COULEUR,POIDS,VILLE)

Table PROJET :

J(JNO,JNOM,VILLE)

Table FOURNISSEUR-PIECE-PROJET :

FPJ(FNO,PNO,JNO,QTE)

1. Interrogation sur une table

1.1. Interrogations simples

SELECT liste-colonnes
FROM nomtable
WHERE condition;

Sélectionne :

- les colonnes précisées dans le SELECT (avec alias : nomcol nouveaunom (Oracle) ou nomcol AS nouveaunom (norme SQL))
- provenant de la table précisée dans le FROM
- dont les lignes vérifient la condition précisée dans le WHERE.
- *Remarques :*
- si on veut toutes les colonnes : *
- si on ne veut pas les doubles : DISTINCT
- dans le SELECT, on peut aussi mettre des expressions calculées ou des chaînes de caractères (entre ' ')
- si la condition est complexe, on la construit à l'aide des opérateurs AND, OR et NOT

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

a) Numéros et statuts des fournisseurs localisés à Paris
SELECT FNO,STATUT
FROM F
WHERE VILLE='Paris';

b) Liste des numéros des pièces
SELECT PNO FROM P;

c) Liste des numéros de pièces effectivement fournies
SELECT DISTINCT PNO FROM FPJ;

d) Numéros des pièces et poids en grammes
SELECT PNO,'Poids en grammes : ', POIDS*454 FROM P;

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

e) Toutes les informations relatives aux fournisseurs
SELECT *
FROM F;

f) Numéros des fournisseurs localisés à Paris avec un statut supérieur à 20
SELECT FNO FROM F
WHERE VILLE='Paris' AND STATUT>20;

g) Numéros des fournisseurs localisés en France (Paris et Marseille)
SELECT FNO FROM F
WHERE VILLE='Paris' OR VILLE='Marseille';

Les opérateurs de comparaison

=

!= ^= <> (différent)

>= > <= <

BETWEEN valeur minimale AND valeur maximale (au sens large)

IS NULL (ne contient pas de valeur)

IS NOT NULL (a une valeur, même 0)

LIKE chaîne de caractères (avec jokers)

caractères jokers : % remplace n'importe quelle suite de caractères

'-' (souligné) remplace n'importe quel caractère

IN (, , ,) inclusion d'une valeur dans une liste NOT IN (, , ,)

= ANY (, , ,) équivaut à IN < > ALL (, , ,) équivaut à NOT IN

on peut accoler n'importe quel opérateur de comparaison simple (=, !=, >=, >, <=, <) avec ANY ou ALL

F(FNO,FNOM,STATUT,VILLE)

P(PNO,PNOM,COULEUR,POIDS,VILLE)

J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

h) Numéros des fournisseurs qui ne sont pas de Londres

```
SELECT FNO
FROM F
WHERE NOT(VILLE='Londres');
ou
SELECT FNO
FROM F
WHERE VILLE != 'Londres';          (!= ou ^= ou <>)
```

i) Numéros des fournisseurs qui ont un statut entre 10 et 30

```
SELECT FNO
FROM F
WHERE STATUT >= 10 AND STATUT <=30;
ou
SELECT FNO
FROM F
WHERE STATUT BETWEEN 10 AND 30;
```

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

j) Numéros des fournisseurs dont le statut est 10,20 ou 30

```
SELECT FNO
FROM F
WHERE STATUT=10 OR STATUT=20 OR STATUT=30;
ou
SELECT FNO
FROM F
WHERE STATUT IN (10,20,30);  (ou = ANY(10,20,30))
```

k) Numéros des fournisseurs dont on ne connait pas la ville

```
SELECT FNO
FROM F
WHERE VILLE IS NULL;
```


F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

l) Numéros des fournisseurs dont le nom commence par C

```
SELECT FNO
FROM F
WHERE FNOM LIKE 'C%';
```

m) Numéros et noms des fournisseurs dont la deuxième lettre est un C

```
SELECT FNO,FNOM
FROM F
WHERE FNOM LIKE '_C%';
```

n) Numéros des fournisseurs dont le statut n'est ni 10, ni 20, ni 30

```
SELECT FNO FROM F WHERE STATUT NOT IN (10,20,30);
ou
SELECT FNO FROM F WHERE STATUT ! = ANY (10,20,30);
```

Table de vérité

X	Y	X and Y	X or Y	not x
Vrai	Vrai	Vrai	Vrai	faux
Vrai	Faux	Faux	Vrai	faux
Faux	Faux	Faux	Faux	Vrai
Vrai	Inconnu	Inconnu	Vrai	Faux
Faux	Inconnu	Faux	Inconnu	vrai
Inconnu	Inconnu	Inconnu	Inconnu	inconnu

1. Interrogation sur une table (suite)

1.2. Interrogations avec tri du résultat

SELECT liste-colonnes
FROM nomtable
WHERE condition
ORDER BY liste-colonnes;

Dans la clause ORDER BY, on peut avoir des :

- des noms de colonnes
- des expressions avec noms de colonnes
- des numéros de position des colonnes dans la clause SELECT.

On peut préciser le sens (croissant ou décroissant) : ASC ou DESC. Par défaut, c'est croissant.

Les valeurs nulles sont à la fin par ordre croissant, au début par ordre décroissant.

Si ORDER BY et DISTINCT sont spécifiés, la clause ORDER BY ne peut pas se référer à des colonnes non mentionnées dans la clause SELECT.

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

o) Numéros des fournisseurs localisés à Paris, dans l'ordre décroissant des statuts

```
SELECT FNO
FROM F
WHERE VILLE='Paris'
ORDER BY STATUT DESC;
```

p) Numéros des fournisseurs dans l'ordre décroissant des statuts et, pour les fournisseurs de même statut, par ordre alphabétique des noms

```
SELECT FNO
FROM F
ORDER BY STATUT DESC, FNOM ASC;
```


F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

q) Supposons que pour la pièce on a PRIXACHAT et PRIXVENTE, donner la liste des numéros de pièces avec les marges unitaires classées de la plus grande à la plus faible

```
SELECT PNO, PRIXVENTE-PRIXACHAT
FROM P
ORDER BY 2 DESC;
```

1. Interrogation sur une table (suite)

1.3. Interrogations avec fonctions

Fonctions numériques

ABS(n)	valeur absolue
CEIL(n)	partie entière + 1
FLOOR(n)	partie entière
MOD(m,n)	modulo n
POWER(m,n)	puissance n
ROUND(n[,m])	arrondi à m décimales
SIGN(n)	signe de n : -1, 0 ou 1
SQRT(n)	racine carrée
TRUNC(n[,m])	tronqué à m décimales

1. Interrogation sur une table (suite)

1.3. Interrogations avec fonctions (suite)

Quelques fonctions sur les caractères

CHR(n)	**caractère de code n**
INITCAP(c)	**transforme la chaîne en 'Titre' (1ère lettre de chaque mot en majuscules)**
LENGTH(c)	**longueur de c**
LOWER(c)	**tout en minuscules**
REPLACE(c,chaîne à remplacer,chaîne de remplac.)	
SOUNDEX(c)	**représentation phonétique**
SUBSTR(c,m,n)	**extrait n caractères à partir du m-ième caractère**
TRANSLATE(c,c1,c2)	**remplace partout dans c le caractère c1 par le caractère c2**
UPPER(c)	**tout en majuscules**

1. Interrogation sur une table (suite)

1.3. Interrogations avec fonctions

Quelques fonctions sur les dates

MONTHS_BETWEEN(d1,d2)
nombre de mois entre 2 dates
SYSDATE **date système**

Quelques autres fonctions

TO_DATE(chaîne,format) convertit la chaîne en date dans le format spécifié
ex de format : DD/MM/YY
TO_CHAR(date,format) convertit la date en chaîne
TO_NUMBER(chaîne) convertit une chaîne en numérique

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

r) Statut du fournisseur Dupont (on ne sait pas si les noms ont été saisis en minuscules ou majuscules)

```
SELECT STATUT
FROM F
WHERE UPPER (FNOM)='DUPONT';
    (ou LOWER (FNOM)='dupont')
```

s) Supposons que l'on ajoute la date de fin de projet DATPROJ dans la table J, rechercher la liste des projets terminés

```
SELECT *
FROM J
WHERE DATPROJ < SYSDATE;
```

1. Interrogation sur une table (suite)

1.4. Interrogations avec agrégats de lignes

SELECT ...
FROM ...
WHERE ...
GROUP BY liste-colonnes
HAVING condition;

Les lignes ayant les mêmes valeurs pour l'ensemble des colonnes du GROUP BY sont regroupées.

Le SELECT contient une fonction qui porte sur un ensemble de valeurs.

Le HAVING permet de tester une condition contenant une fonction agrégat.

Liste des principales fonctions agrégats :

AVG(colonne)
COUNT(DISTINCT colonne) COUNT(*) COUNT(colonne)
MAX(colonne) MIN(colonne)
SUM(colonne)

1. Interrogation sur une table (suite)

Les fonctions agrégats peuvent figurer dans le SELECT ou dans le HAVING

- calcul statistique sur un groupe de lignes vérifiant une condition

```
SELECT fonction statistique
FROM table
WHERE condition;
```

- calcul sur tous les groupes

```
SELECT fonction statistique
FROM table
GROUP BY col1, col2, …;
```

1. Interrogation sur une table (suite)

- calcul sur différents groupes avec condition sur le groupe

```
SELECT …
FROM table
GROUP BY colonnes
HAVING condition;
```

Remarques :

- pas de HAVING sans GROUP BY
- quand il y a GROUP BY, la clause SELECT ne peut contenir que des calculs statistiques et/ou les colonnes du GROUP BY

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

t) Nombre total de fournisseurs

```
SELECT COUNT(*)
FROM F;
```

u) Nombre total de fournisseurs qui fournissent effectivement des pièces

```
SELECT COUNT(DISTINCT FNO)
FROM FPJ;
```

v) Nombre de fournisseurs qui fournissent des pièces de numéro 'P2'

```
SELECT COUNT(DISTINCT FNO)
FROM FPJ
WHERE PNO='P2';
```

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

w) Quantité totale de pièce 'P2' vendue

```
SELECT SUM(QTE)
FROM FPJ
WHERE PNO='P2';
```

x) Valeurs minimale et maximale du statut de fournisseur

```
SELECT MIN(STATUT),MAX(STATUT)
FROM F;
```


F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

y) Quantité totale par type de pièce

```
SELECT PNO, SUM(QTE) FROM FPJ
GROUP BY PNO;
```

z) Quantité moyenne par type de pièce et par projet

```
SELECT PNO,JNO,AVG(QTE) FROM FPJ
GROUP BY PNO,JNO;
```

aa) Numéros des pièces fournies par plus d'un fournisseur

```
SELECT PNO FROM FPJ
GROUP BY PNO
HAVING COUNT(DISTINCT FNO) > 1;
```

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

ab) Villes dont les fournisseurs ont tous le même statut

```
SELECT VILLE FROM F
GROUP BY VILLE
HAVING COUNT(DISTINCT STATUT)=1;
```

ac) Villes ayant des fournisseurs d'au moins 2 statuts

```
SELECT VILLE FROM F
GROUP BY VILLE
HAVING COUNT(DISTINCT STATUT)>1;
ou
SELECT VILLE FROM F
GROUP BY VILLE
HAVING MIN(STATUT)<>MAX(STATUT);
```


2. Interrogation sur plusieurs tables

2.1. Sous-interrogations

Dans la clause WHERE, on peut faire référence à une clause SELECT.

ad) Noms des fournisseurs qui fournissent la pièce 'P2'

```
SELECT FNOM FROM F
WHERE FNO IN (SELECT FNO FROM FPJ
              WHERE PNO='P2');
```

ae) Numéros des fournisseurs localisés dans la même ville que 'F1'

```
SELECT FNO FROM F
WHERE VILLE = (SELECT VILLE FROM F
               WHERE FNO='F1');
```

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

af) Noms des fournisseurs qui fournissent au moins une pièce rouge

```
SELECT FNOM FROM F WHERE FNO IN
    (SELECT FNO FROM FPJ WHERE PNO IN
    (SELECT PNO FROM P WHERE COULEUR='rouge'));
```

ag) Numéros des fournisseurs ayant un statut inférieur au statut maximal

```
SELECT FNO FROM F WHERE
    STATUT < (SELECT MAX(STATUT) FROM F);
```


Commentaires sur les agrégats

- On doit parfois mettre une colonne dans le GROUP BY pour pouvoir l'afficher
- Le WHERE est appliqué avant le HAVING
- Généralisations du GROUP BY : CUBE et ROLLUP

2. Interrogation sur plusieurs tables (suite)

2.2. Jointures

SYNTAXE Oracle :

Dans la clause FROM, on précise la liste des tables à joindre.

Dans la clause WHERE, on précise, en plus de la restriction, les critères de jointure.

On peut donner éventuellement un nom d'alias à chaque table.

Le nom d'alias est obligatoire pour les auto-jointures (jointures d'une table avec elle-même).

2. Interrogation sur plusieurs tables (suite)

2.2. Jointures (suite)

SYNTAXE SQL2 :

SELECT listecolonnes FROM table1 NATURAL JOIN table2;

SELECT listecolonnes FROM table1 JOIN table2 ON critèredejointure;

SELECT listecolonnes FROM table1 CROSS JOIN table2;

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

ah) Noms des fournisseurs qui fournissent la pièce 'P2'

```
SELECT F.FNOM FROM F,FPJ
WHERE F.FNO=FPJ.FNO AND FPJ.PNO='P2';
```

ai) Numéros des fournisseurs localisés dans la même ville que 'F1'

```
SELECT FOUR2.FNO FROM F FOUR1, F FOUR2
WHERE FOUR2.VILLE=FOUR1.VILLE
AND FOUR1.FNO='F1';
```

aj) Noms des fournisseurs qui fournissent au moins une pièce rouge

```
SELECT F.FNOM FROM F,P,FPJ
WHERE F.FNO=FPJ.FNO AND P.PNO=FPJ.PNO
AND P.COULEUR='rouge';
```


F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

ak) Donner les combinaisons fournisseurs-pièces localisés au même endroit

```
SELECT F.*, P.* FROM F,P
WHERE F.VILLE=P.VILLE;
ou
SELECT F.*, P.* FROM F,P,FPJ
WHERE F.FNO=FPJ.FNO AND FPJ.PNO=P.PNO AND F.VILLE=P.VILLE;
```

al) Donner les combinaisons fournisseurs-pièces localisés au même endroit, en ignorant les fournisseurs de statut 20

```
SELECT F.*, P.* FROM F,P
WHERE F.VILLE=P.VILLE AND F.STATUT !=20;
```

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

am) Donner la paire de villes telle que le fournisseur localisé dans la première ville fournit la pièce stockée dans la seconde ville

```
SELECT DISTINCT F.VILLE,P.VILLE
FROM F,P,FPJ
WHERE F.FNO=FPJ.FNO AND FPJ.PNO=P.PNO;
```

an) Noms des fournisseurs qui fournissent au moins une pièce fournie par le fournisseur 'F1'

```
SELECT FOUR1.FNOM
FROM F FOUR1, FPJ C1, FPJ C2
WHERE FOUR1.FNO=C1.FNO AND C1.PNO=C2.PNO
AND C2.FNO='F1';
```


F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

ao) Triplets FNO,PNO,JNO tels que fournisseur, pièce et projet soient situés dans la même ville

```
SELECT DISTINCT F.FNO, P.PNO, J.JNO
FROM F, P, J
WHERE F.VILLE=P.VILLE AND P.VILLE=J.VILLE;
```

ap) Numéros des projets dont au moins un des fournisseurs ne se trouve pas dans la même ville que celle où le projet se déroule

```
SELECT DISTINCT J.JNO
FROM FPJ,F,J
WHERE FPJ.FNO=F.FNO AND FPJ.JNO=J.JNO
AND F.VILLE< >J.VILLE;
```

2. Interrogation sur plusieurs tables (suite)

2.3. Jointures externes

Permettent d'ajouter (avec des valeurs nulles) les lignes n'ayant pas de correspondant dans l'autre table

-> ajouter (+) après la colonne à conserver (syntaxe Oracle)

aq) Liste des quantités fournies par pièce avec les caractéristiques de la pièce

```
SELECT P.*, FNO, JNO
FROM P, FPJ
WHERE P.PNO=FPJ.PNO;
```

ne donne pas ces quantités si la pièce n'est pas décrite dans la table P

2. Interrogation sur plusieurs tables (suite)

2.3. Jointures externes (suite) – syntaxe Oracle

```
SELECT P.*, FPJ.*                  donne toutes les pièces
FROM P, FPJ                        même si elles n'ont pas
WHERE P.PNO=FPJ.PNO (+);                 été commandées

SELECT P.*, FPJ.*                  donne toutes les commandes
FROM P, FPJ                        même celles de pièces non
WHERE P.PNO (+)=FPJ.PNO;           référencées dans P

SELECT P.*, FPJ.*                  interdit : une seule
FROM P, FPJ                        jointure externe par
WHERE P.PNO (+)=FPJ.PNO(+);        prédicat
```

2. Interrogation sur plusieurs tables (suite)

2.3. Jointures externes (suite) – syntaxe norme SQL

SELECT * FROM P NATURAL LEFT OUTER JOIN FPJ;	donne toutes les pièces même si elles n'ont pas été commandées
SELECT * FROM P NATURAL RIGHT OUTER JOIN FPJ;	donne toutes les commandes même celles de pièces non référencées dans P
SELECT * FROM P NATURAL FULL OUTER JOIN FPJ;	les deux

2. Interrogation sur plusieurs tables (suite)

2.4. Opérateur EXISTS

Avec l'opérateur EXISTS, on peut tester le contenu d'une clause SELECT. Il vaut vrai si le résultat du SELECT contient au moins une ligne, faux sinon.

ar) Noms des fournisseurs qui fournissent la pièce 'P2'

```
SELECT FNOM
FROM F
WHERE EXISTS (SELECT *
                FROM FPJ
                WHERE FNO=F.FNO
                AND PNO='P2');
```

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

as) Noms des fournisseurs qui ne fournissent pas 'P2'

```
SELECT FNOM
FROM F
WHERE NOT EXISTS (SELECT * FROM FPJ
                  WHERE FNO=F.FNO
                  AND PNO='P2');
```

at) Noms des fournisseurs qui fournissent toutes les pièces

```
SELECT FNOM
FROM F
WHERE NOT EXISTS (SELECT * FROM P
                  WHERE NOT EXISTS
                        (SELECT * FROM FPJ
                         WHERE FNO=F.FNO
                         AND PNO=P.PNO));
```


3. Opérations ensemblistes sur les interrogations

On peut réaliser des opérations ensemblistes sur les clauses SELECT.

3 opérations ensemblistes

UNION	union de deux ensembles
INTERSECT	intersection de deux ensembles
MINUS	différence de deux ensembles (norme : EXCEPT)

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

au) Numéros des pièces fournies par le fournisseur F2 ou qui pèsent plus de 16 livres

avec opérateur ensembliste

```
SELECT PNO
FROM P
WHERE POIDS > 16
UNION
SELECT PNO
FROM FPJ
WHERE FNO='F2';
```

sans opérateur ensembliste

```
SELECT PNO
FROM P
WHERE POIDS > 16
   OR PNO IN (SELECT PNO
              FROM FPJ
              WHERE FNO='F2');
```


F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

av) Numéros des pièces fournies par le fournisseur F2 et qui pèsent plus de 16 livres

avec opérateur ensembliste

```
SELECT PNO
FROM P
WHERE POIDS > 16
INTERSECT
SELECT PNO
FROM FPJ
WHERE FNO='F2';
```

sans opérateur ensembliste

```
SELECT PNO
FROM P
WHERE POIDS > 16
   AND PNO IN (SELECT PNO
               FROM FPJ
               WHERE FNO='F2');
```

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

aw) Numéros des fournisseurs qui ne fournissent pas la pièce 'P2'

avec opérateur ensembliste	*sans opérateur ensembliste*
SELECT FNO **FROM F** **MINUS** **SELECT FNO** **FROM FPJ** **WHERE PNO='P2';**	**SELECT FNO** **FROM F** **WHERE FNO NOT IN** **(SELECT FNO FROM FPJ** **WHERE PNO='P2');**

4. Opérations de mise à jour

* modification de lignes UPDATE :

UPDATE nomtable
SET nomcol1=exp1
[,nomcol2=exp2]....
[WHERE condition];

* suppression de lignes DELETE :

DELETE FROM nomtable
[WHERE condition];

4. Opérations de mise à jour

* insertion de lignes INSERT :

```
INSERT INTO nomtable
        [(nomcol1[,nomcol2]...)]
VALUES (val1[,val2]...);
```

* si on ne précise pas le nom des colonnes, elles sont insérées dans l'ordre de leur création dans la table
* si on ne met pas toutes les colonnes, les autres contiennent la valeur nulle.

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

ax) Changer la couleur de la pièce 'P2' pour le jaune, accroître son poids de 5 et mettre sa ville à "inconnu"

```
UPDATE P
SET   COULEUR='jaune',
      POIDS=POIDS+5,
      VILLE=NULL
WHERE PNO='P2';
```

ay) Doubler le statut de tous les fournisseurs de Londres

```
UPDATE F
SET STATUT=2*STATUT
WHERE VILLE='Londres';
```

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

az) Mettre à 0 la quantité de pièces achetées pour tous les fournisseurs de Londres

```
UPDATE FPJ SET QTE=0 WHERE 'Londres'=(SELECT
      VILLE FROM F WHERE F.FNO=FPJ.FNO);
```

ou bien

```
UPDATE FPJ SET QTE=0 WHERE FNO IN (SELECT
      FNO FROM F WHERE VILLE = 'Londres');
```

ba) Changer le numéro du fournisseur F2 en F9

```
UPDATE F SET FNO='F9'
WHERE FNO='F2';
```

et

```
UPDATE FPJ SET FNO='F9'
WHERE FNO='F2';
```


F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

bb) Supprimer le fournisseur 'F1'

```
DELETE FROM F
WHERE FNO='F1';
```

bc) Supprimer tous les fournisseurs de Paris

```
DELETE FROM F
WHERE VILLE='Paris';
```

bd) Supprimer toutes les lignes de FPJ

```
DELETE FROM FPJ;
```

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

be) Supprimer toutes les informations relatives aux fournisseurs de Londres

```
DELETE FROM FPJ
WHERE 'Londres'= (SELECT VILLE FROM F
                  WHERE F.FNO=FPJ.FNO);
```

puis

```
DELETE FROM F WHERE VILLE='Londres';
```

bf) Rajouter la pièce P7 (VILLE='Athènes', POIDS=24, PNOM="inconnu", COULEUR="inconnu")

```
INSERT INTO P(PNO,VILLE,POIDS)
VALUES ('P7','Athènes',24);
```

ou bien

```
INSERT INTO P VALUES ('P7', NULL, NULL, 24, 'Athènes');
```


On peut aussi insérer dans une table une sélection d'autres tables

bg) Construire une table donnant pour chaque pièce fournie le numéro de la pièce et la quantité totale fournie

```
CREATE TABLE TEMP
      (PNO CHAR(6), QTETOT NUMBER(4));

INSERT INTO TEMP (PNO,QTETOT)
      SELECT PNO,SUM(QTE)
      FROM FPJ
      GROUP BY PNO;
```

F(FNO,FNOM,STATUT,VILLE)
P(PNO,PNOM,COULEUR,POIDS,VILLE)
J(JNO,JNOM,VILLE)

FPJ(FNO,PNO,JNO,QTE)

bh) Construire une table contenant la liste des références de pièces livrées soit par un fournisseur londonien soit pour un projet se déroulant à Londres

```
CREATE TABLE LP (PNO CHAR(6));
INSERT INTO LP (PNO)
    SELECT DISTINCT PNO FROM FPJ
    WHERE FNO IN (SELECT FNO FROM F
                  WHERE VILLE='Londres')
    OR JNO IN (SELECT JNO FROM J
                  WHERE VILLE='Londres');     ou bien
INSERT INTO LP (PNO)
    SELECT DISTINCT PNO FROM FPJ, F
        WHERE FPJ.FNO=F.FNO AND F.VILLE='Londres'
    UNION
    SELECT DISTINCT PNO FROM FPJ,J
        WHERE FPJ.JNO=J.JNO AND J.VILLE='Londres';
```


5. Autres possibilités de SQL

- SELECT dans le FROM
  - SELECT ncom FROM
    ((SELECT ncom,npro FROM lignecommande)
    MINUS
    (SELECT ncom,npro FROM lignelivraison));
    Commandes dont une ligne au moins n'a pas de livraison
    Permet de simplifier le résultat

5. Autres possibilités de SQL

- Requête récursive : courses cyclistes

  Etape (villedépart, villearrivée)

  SELECT villearrivée FROM Etape

  START WITH villedépart='Dunkerque'

  CONNECT BY PRIOR villearrivée=villedépart;

Donne toutes les villes d'arrivée après Dunkerque

Idem avec LEVEL dans le SELECT qui donne le niveau de l'élément dans le parcours

6. Autorisations d'accès

* Le créateur d'une table (ou d'une vue) en est le propriétaire.

* Par défaut, les autres utilisateurs n'ont pas accès à ces objets (ni interrogation, ni mise à jour).

* Le propriétaire d'une table ou d'une vue peut autoriser d'autres utilisateurs à effectuer certaines opérations.

* Seul le droit de suppression (DROP VIEW ou DROP TABLE) ne peut se céder.

6. Autorisations d'accès (suite)

6.1. L'autorisation

GRANT liste de droits
ON objet
TO liste d'utilisateurs
[WITH GRANT OPTION];

Les droits

SELECT	**ALTER**
UPDATE	**INDEX**
INSERT	**ALL**
DELETE	

UPDATE peut être suivi d'une liste de colonnes
ALL autorise toutes les opérations (sauf DROP)

6. Autorisations d'accès (suite)

6.1. L'autorisation (suite)

Les objets : table ou vue

pour donner un droit à tous les utilisateurs, il faut préciser PUBLIC au lieu d'une liste d'utilisateurs

WITH GRANT OPTION donne en plus la possibilité de transmettre le droit

6. Autorisations d'accès (suite)

6.2. Suppression d'une autorisation

REVOKE liste de droits
ON objet
FROM liste d'utilisateurs;

- **commande symétrique de GRANT**
- **un droit ne peut être retiré que par l'utilisateur qui l'a accordé (ou par le DBA)**
- **si le droit avait été transmis avec l'option WITH GRANT OPTION, la suppression est effectuée en cascade**

Exemple : **interdire toute opération à tout utilisateur sur la table F**

REVOKE ALL
ON F
FROM PUBLIC;

7. Conclusion

- Un langage simple et puissant
- La normalisation induit une logique commune et une évolution homogène
- Chaque SGBD garde, malgré la norme, des spécificités qui sont plutôt des atouts que des limites
- Difficulté d'inscrire un langage assertionnel/ déclaratif dans une logique procédurale