SQL

Versions

- SQL-92 est développé à partir de SEQUEL de IBM et pour le moment a deux standard publiés dans :
  - ANSI X3.135-1992, “Database Language SQL”
  - ISO/IEC 9075:1992, “Database Language SQL”
- Il défini 4 niveaux de complexité
  - Entry
  - Transitional
  - Intermediate
  - Full
- Chaque implémentation de SQL doit maintenir au moins le niveau « Entry »

Sous-langages

- Langage de définition des données
  - Définition des domaines
  - Définition et modifications du schéma
  - Définition des contraintes
  - Définition des vues
  - Définition des droits d'accès
- Langage de manipulation des données
  - Faire des requêtes (Query)
  - Insérer des uplets
  - supprimer des uplets
  - modifier des uplets

Définition des domaines

- Domaines élémentaires
  - ***Types caractères***
    - **Char** pour les caractères
    - **Char(n)** pour les chaînes de n caractères (**Varchar** sous DB2)
    - **Varchar** (sous ORACLE) pour les chaînes interfaçables avec des langages procéduraux.
    - **Long** (ORACLE) pour les chaînes de 65655 caractères maxi.
    - En ACCESS ils sont **Text, Memo**.
  - ***Types numériques***
    - **number**
    - **number(n)** (**float** sous SQL/DS) pour les entiers de longueur variable ou fixe
    - **number(m,n)** (**Décimal** sous SQL/DS) pour les réels de longueur m et de n décimales
    - En ACCESS ils sont **Number(Integer, LongInteger, Byte, Single, Double) , Currency, Autonumber**

Définition des domaines

- Domaines élémentaires
  - Date (Time sous SQL/DS) représente une date sous une structure complexe à champs.
    - En ACCESS ils sont Date/Time
  - Raw (ORACLE V6) pour les données binaires, bitmaps entre autre.
  - BLOB (Binary Large OBjects).
    - En ACCESS ils sont OLE objets
  - Logical
- Domaines nommés – в Interbase
  - CREATE DOMAIN

Définition de schéma

- Création d'une base de données
  - **CREATE DATABASE** за създаване БД (не и в ACCESS)
  - **CREATE SCHEMA** [*SchemaName*] [[authorisation] *Authorization*] { *SchemaElementDefinilion* }
- Création d'un tableau
  - **CREATE TABLE** име (Attribut1 type1, Attribut2 type2, ...);
  - В ACCESS
    CREATE TABLE *table* (*field1 type* [(*size*)] [NOT NULL] [index1] [, field2 type [(size)] [NOT NULL] [index2] [, ...]] [, CONSTRAINT multifieldindex [, ...]])
  - En Interbase
    CREATE TABLE table [EXTERNAL [FILE] " <filespec>"]
    (<col_def> [, <col_def> | <tconstraint> ...]);
    <col_def> = col { datatype | COMPUTED
  - Définition par requête
    CREATE TABLE nom AS SELECT ....

Définition de schéma

- Définition d'une valeur par défaut default(expression)
- Définition des contraintes
  - UNIQUE
    - CONSTRAINT UNIQUE (nomattr,nomattr,…)
  - NOT NULL
  - Clés
    - Clé primaire
      CONSTRAINT nom_clé PRIMARY KEY (nomattr,nomattr,…)
    - Clés étrangère
      CONSTRAINT nom_clé FOREIGN KEY(nomattr,…)
      REFERENCES nom_table (nomattr,…)
  - CHECK (expression)

Modification d'un schéma

- Détruire un tableau
  - DROP TABLE *name*;
- Modification d'un tableau
  - ALTER TABLE *name* ADD COLUMN(*name_of_column type*, ...); - ajouter d'un attribut
  - ALTER TABLE *name* ADD CONSTRAINT ...;
  - ALTER TABLE *name* MODIFY(*column type*. ...); modifier du type d'un attribut
  - ALTER TABLE *name* DROP *column* ,. ...; supprimer un attribut

Index

- Créer d'un index pour qu'on puisse accéder vite un tableau

  CREATE [UNIQUE] INDEX name_index ON name_table (attribut [ASCIDESC],...);

- Supprimer d'un index

  DROP INDEX nom_index;

Vues

- Créer une vue
  - CREATE VIEW *name* [(names **of** columns)] AS SELECT ....;
- Supprimer une vue
  - DROP VIEW *name* ;
- La vue ne contient pas physiquement les données. Il est le nom d'une autre requête mais il peut être utilisé à la place d'un tableau. Son contenue est calculé au temps de exécution de la requête. Les noms des attributs ne sont nécessaires que quand les noms des attributs des tableaux-opérands sont changés

EXEMPLE

```
CREATE TABLE DEPT
(
  DEPTNO            INTEGER NOT NULL,
  DNAME VARCHAR(14) CHARACTER SET ISO8859_1,
  LOC     VARCHAR(13) CHARACTER SET ISO8859_1,
CONSTRAINT PK_DEPT PRIMARY KEY (DEPTNO)
);
CREATE TABLE EMP
(
  EMPNO INTEGER NOT NULL,
  ENAME  VARCHAR(10) CHARACTER SET ISO8859_1,
  JOB     VARCHAR(9) CHARACTER SET ISO8859_1,
  MGR     INTEGER
   CHECK (MGR isNULL or
              DeptNo = (select DeptNo from Employee E where E.DeptNo = MGR) ,
  HIREDATE          TIMESTAMP,
  SAL     NUMERIC(9, 2),
  COMM   NUMERIC(9, 2),
  DEPTNO            INTEGER,
CONSTRAINT PK_EMP PRIMARY KEY (EMPNO)
);
ALTER TABLE EMP ADD CONSTRAINT FK_DEPTNO FOREIGN KEY (DEPTNO)
     REFERENCES DEPT (DEPTNO);
```


EXEMPLE

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17.12.1980	800		20
7499	ALLEN	SALESMAN	7698	20.2.1981	1600	300	30
7521	WARD	SALESMAN	7698	22.2.1981	1250	500	30
7566	JONES	MANAGER	7839	02.4.1981	2975		20
7654	MARTIN	SALESMAN	7698	28.9.1981	1250	1400	30
7698	BLAKE	MANAGER	7839	01.5.1981	2850		30
7782	CLARK	MANAGER	7839	09.6.1981	2450		10
7788	SCOTT	ANALYST	7566	19.4.1987	3000		20
7839	KING	PRESIDENT		17.11.1981	5000		10
7844	TURNER	SALESMAN	7698	08.9.1981	1500		30
7876	ADAMS	CLERK	7788	23.5.1987	1100		20
7900	JAMES	CLERK	7698	03.12.1981	950		30
7902	FORD	ANALYST	7566	03.12.1981	3000		20
7934	MILLER	CLERK	7782	23.1.1982	1300		10

Requêtes

- Instruction SELECT

SELECT [DISTINCT | ALL]{* | expression | attribut },...
FROM <table [alias]>,...
[WHERE condition de sélection ou jointure]
[GROUP BY liste d'attributs]
[HAVING condition pour sélectionner de groupes]
[UNION | INTERSECT | MINUS SELECT...]
[ORDER BY liste d'attributs [ASC | DESC]];

Projection

SELECT job, mgr FROM emp;
/* Query12 */

JOB	MGR
CLERK	7902
SALESMAN	7698
SALESMAN	7698
MANAGER	7839
SALESMAN	7698
MANAGER	7839
MANAGER	7839
ANALYST	7566
PRESIDENT	
SALESMAN	7698
CLERK	7788
CLERK	7698
ANALYST	7566
CLERK	7782

SELECT DISTINCT job, mgr
FROM emp; /* Query13 */

JOB	MGR
CLERK	7902
SALESMAN	7698
MANAGER	7839
ANALYST	7566
PRESIDENT	
CLERK	7788
CLERK	7698
CLERK	7782

Opérateurs d'expressions

Код	Операция
+, -	Положително, обратен знак
*, /	Умножение, деление
+, -, || (&)	Събиране, изваждане, конкатенация

SELECT ENAME || '(' || EMPNO || ')' 'NAME',
2 * SAL 'DoubleSal'
FROM EMP; (Query 39)

'NAME'	'DoubleSal'
SMITH(7369)	1600
ALLEN(7499)	3200
WARD(7521)	2500
JONES(7566)	5950
MARTIN(7654)	2500
BLAKE(7698)	5700
CLARK(7782)	4900
SCOTT(7788)	6000
KING(7839)	10000
TURNER(7844)	3000
ADAMS(7876)	2200
JAMES(7900)	1900
FORD(7902)	6000
MILLER(7934)	2600

Restriction

SELECT * FROM emp WHERE deptno=10; /* Query14 */

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7782	CLARK	MANAGER	7839	09.6.1981	2450		10
7839	KING	PRESIDENT		17.11.1981	5000		10
7934	MILLER	CLERK	7782	23.1.1982	1300		10

Sélection

- Restriction et projection
  SELECT ename, job, sal FROM emp WHERE job = 'MANAGER' AND sal>2500; /* Query15 */

ENAME	JOB	SAL
JONES	MANAGER	2975
BLAKE	MANAGER	2850

Операции

Opérateur	Opération
+, -	plus, négation
*, /	multiplication, division
+, -, || (&)	addition, soustraction, concaténation
=, !=, <, >, <=, >=, IS NULL, LIKE, BETWEEN, IN	comparaison
NOT	booléenne négation
AND	conjonction
OR	disjonction

Prédicats

BETWEEN / NOT BETWEEN
SELECT ename FROM emp WHERE hiredate BETWEEN 1.1.81 AND 31.12.81; /* Query1 */

ENAME
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
KING
TURNER
JAMES
FORD

IN / NOT IN
SELECT ename FROM emp WHERE job In ('ANALYST','MANAGER'); /* Query2 */

ENAME
SMITH
JONES
BLAKE
CLARK
ADAMS
JAMES
MILLER

Prédicats

- LIKE/NOT LIKE compare des chaînes de caractères remplace un nombre quelconque de caractères, _ n'en remplace qu'un.
- En ACCESS les caractères correspondants sont * et ?.

exemple: LIKE 'TARKO%', LIKE '%WSKI', LIKE 'A_C'
SELECT ename, job FROM Emp WHERE.ename LIKE "b*";/* Query4 */

ENAME	JOB
BLAKE	MANAGER

SELECT ename, job,sal FROM emp WHERE ename LIKE '%mi%'; /*Query16*/

ename	job	sal
SMITH	CLERK	800
MILLER	CLERK	1300

UNION et c.

Union de 2 requêtes qui ont le même schéma:
SELECT .... UNION SELECT ....
SELECT ename FROM query2
UNION ALL SELECT ename FROM query7;/*Query 8*/

- Intersection de 2 requêtes qui ont le même schéma:
  SELECT .... INTERSECTION SELECT ...
- Différence de 2 requêtes qui ont le même schéma:
  SELECT .... MINUS SELECT ...

ename
SMITH
JONES
BLAKE
CLARK
ADAMS
JAMES
MILLER
SMITH
ADAMS
JAMES
MILLER

Fonctions

- ***Fonctions numériques***
  - ABS(n) – Valeur absolue;
  - CEIL(n) – fonction plafond;
  - FLOOR(n) – fonction plancher
  - ROUND(n[,m]) – arrondir avec m chiffres après le point
  - TRUNC(n[,m]) – couper après la m-ième chiffre
  - MOD(n) – fonction module
  - POWER(m, n) – m^n .
  - SQRT(n) – racine carrée

Fonctions

- ***Fonction de traitement des caractères.***
  - ASCII(char) renvoie le code ASCII du caractère (ASC en ACCESS)
  - CHR(n) renvoi le caractère dont le code ASCII est passé en paramètre.
  - INITCHAR(chaîne) renvoie la chaîne avec son premier caractère en majuscule.(manque en ACCESS)
  - LENGTH(chaîne) renvoie le nombre de caractères de la chaîne.
  - LPAD(chaîne, n, car), RPAD(chaîne, n, car) remplissage à gauche ou à droite de la chaîne par le caractère *car* n fois.(SPACES(n) renvoie n espaces en ACCESS)
  - LTRIM(chaîne, car), RTRIM(chaîne, car) retire tout ce qui se trouve à gauche ou à droite du caractère *car* s'il est prisent dans la chaîne.
  - TRANSLATE(chaîne, cl, c2) remplace dans la chaîne le caractère cl par c2, sur toutes ses occurrences. .(manque en ACCESS)
  - • SUBSTR(chaîne, pos, longueur) renvoie la sous-chaîne de longueur spécifiée à partir de la position donnée.(MID$(chaîne, pos, longueur) en ACCESS)
  - . UPPER(chaîne), LOWER(chaîne) passe la chaîne en majuscules ou en minuscules.(UCASE, LCASE en ACCESS)
  - • || est un opérateur de concaténation des chaînes de caractères.(En ACCESS (VB) l'opérateur est &)

Fonctions

- ***Fonctions d'agrégat - elles permettent d'effectuer des traitements de globalisation sur un ensemble de uplets.***
  - ***COUNT : retourne le nombre de uplets sélectionnés***
  - ***SUM : retourne la somme des valeurs d'un attribut.***
  - ***AVG : retourne la moyenne des valeurs d'un attribut.***
  - ***MIN, MAX : retourne les valeurs mini et maxi, respectivement, d'un attribut***
  - ***VARIANCE : retourne la variance d'un attribut.(VAR en ACCESS)***

Fonctions

- ***Fonctions d'agrégat.***

SELECT count(*) FROM emp WHERE deptno=20; /* Query10 */

Count
5

SELECT AVG(sal) average FROM emp WHERE deptno=20; /* Query11 */

Average
2175

SELECT count(Job) as Jobs FROM emp;
SELECT count(DISTINCT Job) as Jobs FROM emp;

N'existe pas en Access

Jobs
14

Jobs
5

Fonctions

- ***Fonctions de conversion***
  - TO_CHAR(nombre[,format]) convertit un nombre en chaîne selon un format donné(STR$ et FORMAT en ACCESS)
  - TO_CHAR(date[, format]) conversion de date en chaîne(STR en ACCESS)
  - TO_DATE(chaîne,format) convertit une chaîne en date(CDATE en ACCESS)
  - TO_NUMBER(chaîne) convertit une chaîne en valeur numérique entière ou réelle.(VAL en ACCESS)
  - **En Interbase la fonction de conversion et CAST(*value* AS *datatype*)**
- ***Autres fonctions***
  - DECODE(expression, v1,r1[,v2,r2[,v3,r3]]) donne à l'expression la valeur r1 si elle vaut v1, sinon la valeur r2 si elle vaut v2, etc.
  - NVL(expression1, expression2) retourne l'expression 2 si l'expression 1 vaut NULL (NZ in ACCESS)
  - GREATEST(e1, e2 ...), LEAST(e1, e2,...) retourne les valeurs extrêmes d'une liste de valeurs.
- ***En Interbase on n.a que les fonctions COUNT, SUM, CAST, AVG, UPPER, MAX, MIN***

Jointure

- Produit cartésien

SELECT * FROM emp, dept; /* Query17 */

- Jointure avec qualification

SELECT * FROM emp,dept WHERE emp.deptno = dept.deptno; /* Query18 */
En ACCESS ou Interbase:
SELECT * FROM emp INNER JOIN dept ON emp.deptno = dept.deptno; /* Query19 */
On peut renommer le nom d'une table (ou d'une colonne) dans une requête à l'aide d'un alias (pseudonyme) plus simple à manipuler. Spécialement pratique avec les jointures.
SELECT * FROM emp E,dept D WHERE E.deptno = D.deptno; /*Query20 */
SELECT * FROM emp as E INNER JOIN dept as D ON E.deptno = D.deptno; /* Query21 */

Jointure

EMPNO	ENAME	JOB	MGR	HIRED	SAL	COMM	E.DEPTNO	D.DEPTNO	DNAME	LOC
7782	CLARK	MANAGER	7839	09.6.1981	2450		10	10	ACCOUNTING	NEW YORK
7839	KING	PRESIDENT		17.11.1981	5000		10	10	ACCOUNTING	NEW YORK
7934	MILLER	CLERK	7782	23.1.1982	1300		10	10	ACCOUNTING	NEW YORK
7369	SMITH	CLERK	7902	17.12.1980	800		20	20	RESEARCH	DALLAS
7566	JONES	MANAGER	7839	02.4.1981	2975		20	20	RESEARCH	DALLAS
7788	SCOTT	ANALYST	7566	19.4.1987	3000		20	20	RESEARCH	DALLAS
7876	ADAMS	CLERK	7788	23.5.1987	1100		20	20	RESEARCH	DALLAS
7902	FORD	ANALYST	7566	03.12.1981	3000		20	20	RESEARCH	DALLAS
7499	ALLEN	SALESMAN	7698	20.2.1981	1600	300	30	30	SALES	CHICAGO
7521	WARD	SALESMAN	7698	22.2.1981	1250	500	30	30	SALES	CHICAGO
7654	MARTIN	SALESMAN	7698	28.9.1981	1250	1400	30	30	SALES	CHICAGO
7698	BLAKE	MANAGER	7839	01.5.1981	2850		30	30	SALES	CHICAGO
7844	TURNER	SALESMAN	7698	08.9.1981	1500		30	30	SALES	CHICAGO
7900	JAMES	CLERK	7698	03.12.1981	950		30	30	SALES	CHICAGO

Restriction sur jointure

```
SELECT ename, job, dept.deptno, dname FROM emp, dept
WHERE emp.deptno = dept.deptno AND job = 'CLERK'; /* Query22 */
SELECT ename, job, dept.deptno, dname FROM emp INNER JOIN dept ON
    emp.deptno = dept.deptno WHERE  job = 'CLERK';
```

ename	job	deptno	dname
SMITH	CLERK	20	RESEARCH
ADAMS	CLERK	20	RESEARCH
JAMES	CLERK	30	SALES
MILLER	CLERK	10	ACCOUNTING

Sous-requêtes

```
SELECT ename, deptno  FROM emp
WHERE deptno IN ( SELECT deptno FROM dept WHERE dname LIKE '%S%')
    ; /* Query23 */
SELECT ename, e.deptno  FROM emp E INNER JOIN dept D ON
    e.deptno=d.deptno WHERE dname LIKE '%S%';
```

ename	deptnoo
SMITH	20
ALLEN	30
WARD	30
JONES	20
MARTIN	30
BLAKE	30
SCOTT	20
TURNER	30
ADAMS	20
JAMES	30
FORD	20

Sous-requêtes

- SOME, ANY

```
SELECT ename FROM emp WHERE
    sal >ANY (SELECT sal FROM emp
    WHERE deptno = 20); /* Query5*/
SELECT ename FROM emp WHERE
    sal > (SELECT MIN(sal) FROM emp
    WHERE deptno = 20);
```

ENAME
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS
JAMES
FORD
MILLER

- ALL

```
SELECT ename FROM emp WHERE
    sal > ALL (SELECT sal FROM emp
    WHERE deptno = 20); /* Query6 */
SELECT ename, sal  FROM emp
    WHERE deptno = 10 AND sal > (
    SELECT MAX(sal) FROM emp
    WHERE deptno = 20); /* Query24 */
```

ename	sal
KING	5000

- EXISTS

```
SELECT dname, deptno FROM dept
    WHERE EXISTS (SELECT * FROM
    emp WHERE dept.deptno =
    emp.deptno); /*Query 35*/
```

dname	deptno
ACCOUNTING	10
RESEARCH	20
SALES	30

Sous-requêtes

- Sous-requêtes corrélatives - La requête principale fournit l'une après l'autre des valeurs à la requête secondaire

```
SELECT ename, deptno  FROM emp E1 WHERE E1.sal>(SELECT AVG(E2.sal)
    FROM emp E2 WHERE E1.deptno=E2.deptno); /* Query25 */
```

- Sous-requêtes de deuxième niveau

```
SELECT dname, deptno FROM dept AS D2 WHERE EXISTS (SELECT * FROM
    emp E WHERE D2.deptno = D.deptno and E.sal > (SELECT AVG(sal) FROM
    emp E1, dept D1 WHERE E1.deptno=D1.deptno AND  D1.dname LIKE
    'ACCOUNTING')) /* Query36 */;
```

ename	deptno
ALLEN	30
JONES	20
BLAKE	30
SCOTT	20
KING	10
FORD	20

dname	deptno
ACCOUNTING	10
RESEARCH	20

Regroupements

- GROUP BY permet de regrouper selon un critère donné, les uplets manipulés par une fonction d'agrégat. **Cette clause ne s'applique que sur un attribut qui n'est pas manipulé par une fonction, d'agrégat !!**
- **L'ordre d'exécution:**
  - **S'il y a une clause WHERE tous les uplets violant la condition sont éliminés.**
  - **Le regroupement est fait et les valeurs d'agrégat sont calculées**
  - **Tout groupe violant la condition de la cause HAVING sont éliminés**

```
SELECT deptno, MIN(sal), MAX (sal) FROM emp GROUP BY deptno; /*
    Query26 */
SELECT deptno, MIN(sal), MAX (sal)FROM emp WHERE job = 'CLERK'
    GROUP BY deptno; /* Query27 */
```

deptno	Minsal	Maxsal
10	1300	5000
20	800	3000
30	950	2850

deptno	Minsal	Maxsal
10	1300	1300
20	800	1100
30	950	950

Regroupements

- HAVING

  permet d'exprimer des conditions sur les groupes (opérateurs d'agrégat) et ne s'emploie qu'avec GROUP BY

```
SELECT deptno, MIN(sal), MAX(sal) FROM emp GROUP BY deptno HAVING
    MAX(sal)< 4000; /* Query28 */
SELECT deptno, MIN(sal), MAX(sal) FROM emp WHERE job = 'CLERK'
    GROUP BY deptno HAVING MIN(sal)<1000; /* Query29 */
```

deptno	Minsal	Maxsal
20	800	3000
20	950	2850

deptno	Minsal	Maxsal
20	800	1100
30	950	950

Un exemple compliqué

- Trouver le quelle partie du nombre total d'employées est dans chaque département et quelle partie du salaire total touchent ils
  - ORACLE

```
SELECT a.deptno "Department", a.num_emp/b.total_count "%Employees",
    a.sal_sum/b.total_sal "%Salary"
FROM (SELECT deptno, COUNT(*) num_emp, SUM(SAL) sal_sum FROM
    emp
GROUP BY deptno) a,
(SELECT COUNT(*) total_count, SUM(sal) total_sal FROM emp) b ;
```


Un exemple compliqué

```
CREATE VIEW X AS SELECT deptno, COUNT(*) num_emp, SUM(SAL)
    sal_sum FROM emp
GROUP BY deptno; /*QueryA*/

CREATE VIEW  Y AS SELECT COUNT(*) total_count, SUM(sal) total_sal
    FROM emp); /*QueryB*/

SELECT x.deptno AS Department, x.num_emp/y.total_count AS
    Pr_Employees, x.sal_sum/y.total_sal AS Pr_Salary
FROM X, Y; /* QueryC */
```

deptno	num_emp	sal_sum
10	3	8750
20	5	10875
30	6	9400

total_count	total_sal
14	29025

Department	Pr_Employees	Pr_Salary
10	0.214285714285714	0.301464254952627
20	0.357142857142857	0.374677002583979
30	0.428571428571429	0.323858742463394

Jointure externe

- Syntaxe

FROM table …]{LEFT | RIGHT | FULL } [OUTER]} JOIN table ON лог.израз

– Пример

```
SELECT emp.ENAME, emp.JOB,
    dept.DEPTNO, dept.DNAME
FROM emp RIGHT JOIN dept ON
    emp.DEPTNO = dept.DEPTNO;
```

```
SELECT dept.DNAME,
    Count(emp.EMPNO) AS
    CountOfEMPNO
FROM emp RIGHT JOIN dept ON
    emp.DEPTNO = dept.DEPTNO
GROUP BY dept.DNAME;  /*Query 40*/
```

ENAME	JOB	DEPTNO	DNAME
JONES	MANAGER	20	RESEARCH
SCOTT	ANALYST	20	RESEARCH
ADAMS	CLERK	20	RESEARCH
FORD	ANALYST	20	RESEARCH
ALLEN	SALESMAN	30	SALES
WARD	SALESMAN	30	SALES
MARTIN	SALESMAN	30	SALES
BLAKE	MANAGER	30	SALES
TURNER	SALESMAN	30	SALES
JAMES	CLERK	30	SALES
		40	OPERATIONS

DNAME	CountOfEMPNO
ACCOUNTING	3
OPERATIONS	0
RESEARCH	5
SALES	6

Ordonnancement

- ORDER BY

ORDER BY {ASC | DESC}

– Exemple

```
SELECT ename, deptno, sal FROM emp ORDER BY deptno, sal DESC; /*
    Query30 */
```

ename	deptno	sal
KING	10	5000
CLARK	10	2450
MILLER	10	1300
FORD	20	3000
SCOTT	20	3000
JONES	20	2975
ADAMS	20	1100
SMITH	20	800
BLAKE	30	2850
ALLEN	30	1600
TURNER	30	1500
MARTIN	30	1250
WARD	30	1250

Hiérarchie

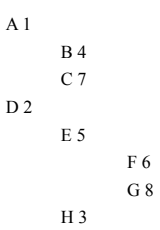
- Présentation des données dans une structure arboresque

SELECT [level] ... CONNECT BY PRIOR expr = expr START WITH expr

SELECT Lpad(' ', 2*level)|| nom, Num FROM EMP CONNECT BY PRIOR chef= num START WITH nom='A';

```
SELECT LPAD(' ',2*(LEVEL-1)) || ename,empno, mgr, job
FROM emp START WITH job = 'PRESIDENT'
CONNECT BY PRIOR empno = mgr;
```

NUM	NOM	CHEF
1	A	NULL
2	D	NULL
3	H	2
4	B	1
5	E	2
6	F	5
7	C	1
8	G	5



Hiérarchie

- Dans notre exemple

```
SELECT LPAD(' ',2*(LEVEL-1)) || ename,empno, mgr, job
FROM emp START WITH job = 'PRESIDENT'
CONNECT BY PRIOR empno = mgr;
```

ENAME	EMPNO	MGR	JOB
KING	7839		PRESIDENT
BLAKE	7698	7839	MANAGER
ALLEN	7499	7698	SALESMAN
WARD	7521	7698	SALESMAN
MARTIN	7654	7698	SALESMAN
TURNER	7844	7698	SALESMAN
JAMES	7900	7698	CLERK
CLARK	7782	7839	MANAGER
MILLER	7934	7782	CLERK
JONES	7566	7839	MANAGER
SCOTT	7788	7566	ANALYST
ADAMS	7876	7788	CLERK
FORD	7902	7566	ANALYST
SMITH	7369	7902	CLERK

Requêtes paramétrisées

- Paramètres dans la requête - :varname
- Access – property *parameters*

Mise en jour

- **Insertion**

  INSERT INTO table [(col1[,col2…])] VALUES(liste de valeurs); ou :

  INSERT INTO table [(col1[,col2…])] VALUES SELECT ... ;

  – Exemple :

  INSERT INTO Emp (EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)

  SELECT [EMPNO]+20 AS Expr5, ENAME, "CLERK" AS Expr2, #9/1/99# AS Expr4, 800 AS Expr3, COMM, 40 AS Expr1 FROM Emp WHERE DEPTNO=10;/*Query 31*/

Mise en jour

- **Modification**

  UPDATE table или view SET {column = expression | (list of columns)=(list of expressions)} [WHERE condition] ;

  – L'expression peut être une instruction SELECT qui produit la liste des valeurs

  – Exemple:

  UPDATE Emp SET Emp.SAL = [Sal]+100 WHERE DEPTNO=40; /Query32*/

- **Suppression**

  DELETE FROM table [WHERE condition];

  – Exemple :

  DELETE FROM emp WHERE DEPTNO=40;/*Query33*/

Gestion des droits d'accès

- ***Droits d'accès généraux :***

  – GRANT CONNECT | RESOURCE | ROLE | DBA(ADMIN) TO usager IDENTIFIED BY mot de passe;

  - CONNECT, RESOURCE des rôles prédéfinis dans SGBD Oracle. Leur utilisation n'est pas recommandée dans les dernières versions.

  – ROLE est un ensemble nommé des privilèges qui peuvent être donnés comme un tout. Il est manipulé par les commandes :

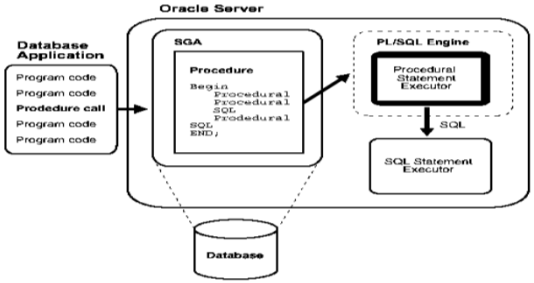
  - CREATE ROLE
  - ALTER ROLE
  - SET ROLE

Gestion des droits d'accès

- Accès aux objets (avec droit de retransmission éventuel) :
  - GRANT SELECT | INSERT | DELETE | UPDATE | ALTER | INDEX | CLUSTER ON tableau ou vue TO usager ou rôle [WITH GRANT OPTION] ;
  - GRANT OPTION donne à l'usager le droit de retransmettre ses droits.
- Privation des droits
  - REVOKE
  - REVOKE droit ON tableau ou vue FROM usager

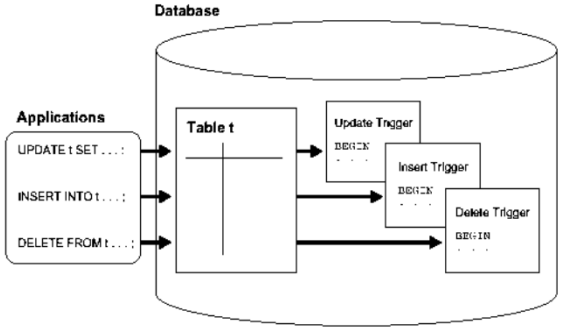
Outils divers

- Curseurs (Cursor)
  - Les curseurs sont des pointeurs vers d'ensembles des lignes considérées comme enregistrements pour les traiter une par une à l'aide d'un langage algorithmique. En ACCESS existe l'objet RECORDSET.
- Procédures mémorisées (Stored procedures)



Outils divers

- Procédures déclencheurs (Triggers)



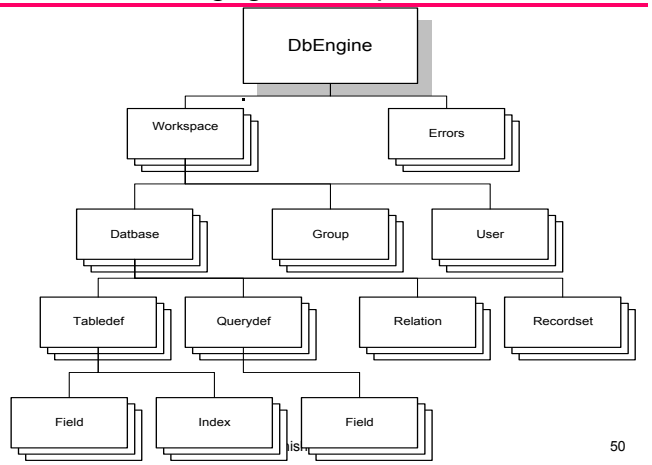
Transactions

- Définition – une suite d'action qui sont considérées est exécutées comme un tout.
- Propriétés
  - Isolation
  - Atomicité
- Instructions pour gérer les transactions
  - begin trans
  - commit
  - rollback

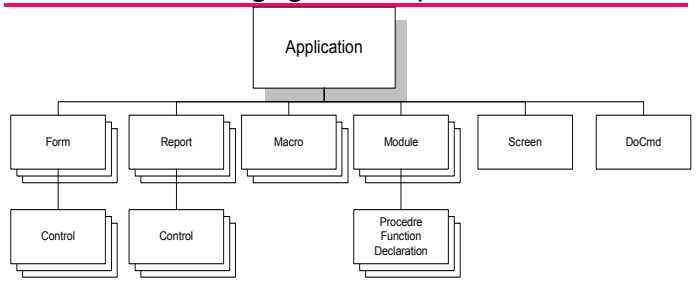
Langages incorporés

- PL/SQL – Oracle
- VBA – ACCESS
- VB, C# - SQL Server

Langages incorporés



Langages incorporés



Les événements qui sont traités sont **Open**, **Close**, **Load**, **Current** Record, **Before** et **After Insert**, **Before et After Update**, **Delete** pour les formulaires, **Got et Lost Focus**, **Before et After Update** pour les contrôles. Pour les reports les événements importants sont **Open, Format, Print, Close**. En plus les déplacements de la souris, les clicks de la souris sur les contrôles et les pressions des touches du clavier peuvent être traités

Exercice

- Schéma

Fournisseur(FurnNo, Nom, Pays, Adresse, Téléphone)
Marchandise(MarNo, Nom, Unite)
Prix(FurnNo, MarNo, PrixUn)
Livraison(LivNo,Date, FurnNo)
Detail(LivNo,MarNo,Quant)
Ecrire les opérateurs SQL qui vont répondre aux questions suivantes :

1. Qui sont les fournisseurs avec nom contenant « ma »?

```
SELECT * FROM Fournisseur WHERE Nom LIKE '%ma%';
```

2. Quelles sont les marchandises fournies par des fournisseurs de France ?

```
SELECT DISTINCT M.Nom FROM Fournisseur F , Prix P, Marchandise M
WHERE M.MarNo=P.MarNo and F.FurnNo=P.FurNo and F.Pays='France';
```

```
SELECT DISTINCT M.Nom FROM Marchandise M, Prix P WHERE
M.MarNo=P.MarNo and P.FurnNo in (SELECT F.FurnNo FROM Fournisseur F
WHERE F.Pays='France');
```

3. Qui sont les prix minimaux, maximaux et moyens des marchandises ?

```
SELECT M.Nom, Max(PrixUn), Min(PrixUn), Avg(PrixUn)
FROM Prix P INNER JOIN Marchandise M ON P.MarNo = M.MarNo
GROUP BY M.Nom;
```

Exercice

Fournisseur(FurnNo, Nom, Pays, Adresse, Téléphone)
Marchandise(MarNo, Nom, Unite)
Prix(Furno, MarNo, PrixUn)
Livraison(LivNo,Date,FourNo)
Detail(LivNo,MarNo,Quant)

4. Quel et le nombre des marchandises fournis par chacun des fournisseurs ?

```
SELECT F.FurnNo, F.Nom, Count(MarNo)
FROM Fournisseur F LEFT JOIN Prix P ON F.FurnNo = P.FurnNo
GROUP BY F.FurnNo, F.Nom;
```

5. Lesquels des fournisseurs ne fourni aucun fromage (tous les noms de fromages contiennent le mot fromage) ?

```
SELECT  F.FurnNo F.Nom FROM Fournisseur F , Marchandise M , Prix P
WHERE F.FurnNo=P.FurnNo and M.MarNo=P.MarNo and
NOT (M.Nom LIKE '%fromage%');
```

```
SELECT F.FurnNo, F.Nom FROM Fournisseur F
WHERE F.FurnNo Not IN
(SELECT  P.FurnNo FROM Marchandise M , Prix P
WHERE M.MarNo=P.MarNo and M.Nom LIKE '%fromage%');
```


Exercice

Fournisseur(FurnNo, Nom, Pays, Adresse, Téléphone)
Marchandise(MarNo, Nom, Unite)
Prix(FurnNo, MarNo, PrixUn)
Livraison(LivNo,Date, FurnNo)
Detail(LivNo,MarNo,Quant)

6.Faites comparaison entre les prix moyens des fournisseurs de France et les fournisseurs du reste du monde

```
CREATE VIEW France (Nom, avg) AS
SELECT M.Nom, Avg(PrixUn)
FROM (Fournisseur F INNER JOIN Prix P ON F.FurnNo=P.FurnNo)
INNER JOIN Marchandise M ON P.MarNo = M.MarNo
WHERE F.Pays LIKE 'France'
GROUP BY M.Nom;
```

```
CREATE VIEW NoFrance (Nom, avg) AS
SELECT M.Nom, Avg(PrixUn)
FROM (Fournisseur F INNER JOIN Prix P ON F.FurnNo=P.FurnNo)
INNER JOIN Marchandise M ON P.MarNo = M.MarNo
WHERE NOT (F.Pays LIKE 'France')
GROUP BY M.Nom;
```

```
SELECT NVL(F.Nom,N.Nom), F.Avg as France, N.Avg AS NoFrance
FROM France F FULL JOIN NOFrance N ON F.Nom = F.Nom;
```


Exercice

Fournisseur(FurnNo, Nom, Pays, Adresse, Téléphone)
Marchandise(MarNo, Nom, Unite)
Prix(FurnNo, MarNo, PrixUn)
Livraison(LivNo,Date, FurnNo)
Detail(LivNo,MarNo,Quant)

7. Faites un rapport mensuel par fournisseur – la somme de tous les livraisons pour chaque fournisseur

```
SELECT F.FurnNo, F.Nom, Sum(Quant*PrixUn)
FROM Fournisseur F , Prix P, Livraison L, Detail D
WHERE F.FurnNo = P.FurnNo and F.FurnNo = L.FurnNo and L.LivNo= D.LivNo and
P.MarNo=D.MarNo and Date Between 1.4.03 and 30.4.03
GROUP BY F.FurnNo, F.Nom
ORDER By F.Nom;
```

8. Trouvez la moyenne des sommes des livraisons par jour pour le dernier mois

```
CREATE VIEW Jour AS SELECT Date, Sum(Quant*PrixUn) AS Suma
FROM Prix P, Livraison L, Detail D
WHERE  D.LivNo= L.LivNo and P.MarNo=D.MarNo and P.FurnNo=FurnNo
and Date Between 1.4.03 and 30.4.03
GROUP BY Date;
```

```
SELECT AVG(Suma) FROM Jour;
```


Exercice 2

Schéma relationnel

Jet (JetNum, JetName, Cap) **JetName** est le nom du modèle de l'avion i.e. AirBus 320A, et **Cap** est sa capacité.

Pilotes(NumPil, Name, Birth,City).

Fly(FlyNum, CityL, CityA, DateL, DateA, NumPil, JetNom, Price) **Price** est le prix minimal pour ce vol.

Class (JetNum,Class,CoeffPlace, CoeffPrice) **CoeffPlace** est dans [0, 1], est donne le pourcentage des places pour la classe dans ce modèle d'avion. **CoefPrice** (>=1) c'est le multiplicateur, qui multiplié par Price, donne le prix réel pour le vol dans la classe **Class.**

Clients(NumCl, NameCl, Street, StrNum, PostCode, CityCl)

Reservations(NumCl, FlyNum, Class, NbPlaces)

Exercice 2

Requêtes

1. Les noms des pilotes qui conduisent tous les Boings.
2. Les numéros et les noms des clients qui ont fait plus de 3 réservations et la somme totale des réservations de chacun d'eux.
3. Les numéros et les noms des clients qui ont fait réservations de place pour un vol pour lequel il y a réservation de M. Grandtoupe
4. Le numéro et le nom du pilote le plus âgé qui conduit un Airbus
5. Le nombre des villes desservis par la société.
6. Les numéros des vols assurant le trajet inverse du vol F101.
7. Les numéros et les noms des pilotes qui ne réalisent aucun vol (2 moyens).
8. Augmentez de 10% les prix de tous vols qui partent de Sofia
9. Qui sont les vols les plus profitables
10. Qui sont les clients loyaux . (Qui ont payé les somme les plus importants)