Chapitre IV
SQL

Année 2023-2024

SQL

schéma

données

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

WORKS

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P5	Engineer	23
E8	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

PAY

TITLE	SALARY
Elect. Eng.	55000
Syst. Anal.	70000
Mech. Eng.	45000
Programmer	60000

EMP(ENO, ENAME, TITLE)
PROJ (PNO, PNAME, BUDGET)
WORKS(ENO,PNO, RESP, DUR)
PAY(TITLE, SALARY)

Les attributs soulignés sont les *clés* (identifiants de n-uplets)

EMP

ENO	ENAME	TITLE

PROJ

PNO	PNAME	BUDGET

WORKS

ENO	PNO	RESP	DUR

PAY

TITLE	SALARY

SQL

SQL permet d'*interroger* et de *créer*, de *modifier* et de *supprimer* des bases de données.

- SQL query : interrogation de données
  - pas de modification dans la BD
- SQL-DDL : création, modification et suppression de *schémas*
  - création, modification et suppression de schémas de relation
  - définition de clés et d'autres contraintes
- SQL-DML : création, modification et suppression de *données*
  - insertion, modification et suppression de n-uplets

SQL

Création de tables :

```
CREATE TABLE nom_table
   (Attribute_1 <Type>[DEFAULT <value>],
    Attribute_2 <Type>[DEFAULT <value>],
    …
    Attribute_n <Type>[DEFAULT <value>]
   [<Constraints>])
```

Définition de la relation **Project**

```
CREATE TABLE Project (
   Pno        CHAR(3),
   Pname      VARCHAR(20),
   Budget     NUMBER(10,2) DEFAULT 0.00,
   City       CHAR(9));
```

Contraintes d'attributs

PRIMARY KEY (<attributs>)

- désigne un *ensemble d'attributs* comme la *clé primaire* de la table

FOREIGN KEY (attributs) REFERENCES <table>

- désigne un *ensemble d'attributs* comme la *clé étrangère* dans une contrainte référentielle (plus tard)

<attribut> NOT NULL

- spécifie qu'un attribut peut ne pas être renseigné

UNIQUE (<attributs>)

- spécifie un *ensemble d'attributs* dont les valeurs doivent être distinctes pour chaque couple de n-uplets (clé).

Créer la table Project(<u>Pno</u>, Pname, Budget, City) :

```
CREATE TABLE Project
       (Pno   CHAR(3),                          ou Pno CHAR(3) PRIMARY KEY
        Pname VARCHAR(20) UNIQUE NOT NULL,
        Budget DECIMAL(10,2) DEFAULT 0.00,
        City CHAR(9),
        PRIMARY KEY (Pno));
```

Commandes DDL

- **Exemple : Clés et clés étrangères**

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Définition des références inter-tables

```
CREATE TABLE Works
(    Eno    CHAR(3),
     Pno    CHAR(3),
     Resp   CHAR(15),
     Dur    INT,
PRIMARY KEY (Eno,Pno),
FOREIGN KEY (Eno) REFERENCES Emp(Eno),
FOREIGN KEY (Pno) REFERENCES Project(Pno));
```

Autres commandes DDL

DROP TABLE *nom_table* [,...] [**CASCADE** | **RESTRICT**]

- RESTRICT : supprime la table seulement si elle n'est référencée par aucune contrainte (clé étrangère) ou vue (par défaut)
- CASCADE : supprime aussi toutes les tables qui « dépendent » de *nom_table*

ALTER TABLE *nom_table* **OPERATION**

- *modifie* la définition de la table
- opérations:
  - Ajouter (ADD), effacer (DROP), changer (MODIFY) attributs et contraintes
  - changer propriétaire, ...

Requêtes d'interrogation SQL

Structure de base d'une requête SQL simples :

SELECT	$var_1.A_{1k}$, ...	attributs
FROM	R_{i1} var_1, R_{i2} var_2...	tables
WHERE	P	prédicat/condition

où:

- var_i désigne la table R_{ij}
- Les variables dans la clause SELECT et dans la clause WHERE doivent être *liées* dans la clause FROM.

Simplifications :

- Si var_j n'est pas spécifiée, alors la variable s'appelle par défaut R_{ij}.
- Si une seule table/variable *var* possède l'attribut A, on peut écrire plus simplement A au lieu de *var*.A.

Requêtes simples

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms de tous les employés ?

```
SELECT t.Ename FROM   Emp t
```

```
SELECT Ename
FROM   Emp
```

Noms des projets avec leurs budgets ?

```
SELECT Pname, Budget FROM   Project
```

Villes où un projet existe ?

DISTINCT enlève les doublons !

```
SELECT DISTINCT City FROM   Project
```

Tous les employés (toutes les informations) ?

```
SELECT * FROM   Emp
```

Requêtes simples

Exemple

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	*CAD/CAM*	250000
P4	Maintenance	310000
P5	*CAD/CAM*	500000

SELECT PNO, BUDGET
FROM PROJ :

PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000
P5	500000

SELECT PNAME FROM
PROJ :

PNAME
Database Develop.
Instrumentation
CAD/CAM
Maintenance
CAD/CAM

SELECT DISTINCT PNAME
FROM PROJ :

PNAME
Maintenance
CAD/CAM
Database Develop.
Instrumentation

Prédicats

Prédicats simples :

- Expression1 θ Expression2
  - où Expression1 peut être un attribut ou une expression arithmétique impliquant des attributs, θ = {<, >, =, <=, >=, <>} et Expression2 une expression ou une valeur de domaine
- Exemples :
  - R.Name = 'J. Doe'
  - (S.Age + 30) >= 65
  - R.A = S.B

Prédicats composés :

- prédicats simples combinés avec les connecteurs logiques AND, OR, NOT

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Professions qui gagnent plus de 50 000 € par an ?

```
SELECT x.Title
FROM   Pay x
WHERE  x.Salary > 50000
```

Numéros des managers d'un projet qui dure plus de 17 mois?

```
SELECT Eno
FROM   Works
WHERE  Dur > 17 AND Resp='Manager'
```

Sélection

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

```
SELECT * FROM EMP
WHERE TITLE = 'Elect. Eng.'
```

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E6	L. Chu	Elect. Eng.

Requêtes avec plusieurs relations : jointure

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms et titres des employés qui travaillent dans un projet pendant plus de 17 mois?

```
SELECT Ename, Title
FROM   Emp, Works
WHERE  Dur > 17
AND Emp.Eno = Works.Eno
```

Noms et titres des employés qui travaillent dans un projet à Paris ?

```
SELECT Ename, Title
FROM   Emp E,Works W,Project P
WHERE  P.City = 'Paris'
AND    E.Eno = W.Eno AND W.Pno = P.Pno
```

Exemple

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

PAY

TITLE	SALARY
Elect. Eng.	55000
Syst. Anal.	70000
Mech. Eng.	45000
Programmer	60000

```
SELECT ENO, ENAME, EMP.TITE, SALARY
FROM EMP, PAY
WHERE EMP.TITLE=PAY.TITLE
```

ENO	ENAME	EMP.TITLE	SALARY
E1	J. Doe	Elect. Eng.	55000
E2	M. Smith	Analyst	70000
E3	A. Lee	Mech. Eng.	45000
E4	J. Miller	Programmer	60000
E5	B. Casey	Syst. Anal.	70000
E6	L. Chu	Elect. Eng.	55000
E7	R. Davis	Mech. Eng.	45000
E8	J. Jones	Syst. Anal.	70000

Tri du résultat : ORDER BY

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms, budgets et villes des projets de budget supérieur à 250 000 euros, en ordonnant le résultat par ordre décroissant de budget puis par nom par ordre alphanumérique croissant ?

```
SELECT Pname, Budget, City
 FROM  Project
 WHERE Budget > 250000
 ORDER BY Budget DESC, Pname
```

Par défaut, l'ordre est ascendant (ASC). L'ordre descendant peut être spécifié par le mot-clé DESC