Bases de Données Relationnelles

SQL

Le langage de définition des données de SQL

Introduction

- SQL : Structured Query Language
- SQL est normalisé
  - SQL 2: adopté (SQL 92)
  - SQL 3: adopté (SQL 99)
- Standard d'accès aux bases de données relationnelles

SQL : Trois langages

- Langage de définition de données (LDD/DDL)
  - création de relations : CREATE TABLE
  - modification de relations: ALTER TABLE
  - suppression de relations: DROP TABLE
  - vues, index .... : CREATE VIEW ...
- Langage de manipulation de données (LMD /DML)
  - insertion de tuples: INSERT
  - mise à jour des tuples: UPDATE
  - suppression de tuples: DELETE
- Langage de requêtes (LMD/DML)
  - SELECT ..... FROM ..... WHERE .....

Terminologie

- Relation –> table
- Tuple –> ligne
- Attribut –> colonne (column)
- Identifiant –> clé primaire (primary key)

  clé secondaire (unique)
- Identifiant externe

  –> clé externe (external key)

Langage de Définition de Données

- Commandes pour créer, modifier et supprimer les éléments du schéma (pour l'instant table et vue)
- CREATE TABLE : créer une table
- CREATE VIEW : créer une vue particulière sur les données à partir d'un SELECT
  table dérivée
- DROP TABLE / VIEW : supprimer une table ou une vue
- ALTER TABLE / VIEW : modifier une table ou une vue.

CREATE TABLE

- Commande créant une table en donnant son nom, ses attributs et ses contraintes

- CREATE TABLE nom-table
  { (nom-col type-col [DEFAULT valeur]
  [[CONSTRAINT] contrainte-col])*
  [[CONSTRAINT] contrainte-table]*

  | AS requête-SQL };

- Légende :
  - {a | b} : a ou b
  - [option]
  - * : applicable autant de fois que souhaité
  - mot en capitale : mot clé

CREATE TABLE

- CREATE TABLE nom_table
  { (nom-col type-col [DEFAULT val] [[CONSTRAINT] contrainte-col])*
  [[CONSTRAINT] contrainte-table]* | AS requête-SQL };

- Exemples :
  - CREATE TABLE Doctorant
    (nom VARCHAR(20),
    prénom VARCHAR(15),
    année_insc DECIMAL(4) DEFAULT 2003) ;

  - CREATE TABLE Doctorant
    AS SELECT nom, prénom, année_inscr
    FROM Etudiant WHERE statut='Doctorant' ;

Domaines de valeurs

CHAR(nb)

VARCHAR(max)

INTEGER

NUMERIC(n,p)

DATE

TIME

TIMESTAMP

...

Contraintes

- contrainte-col : contrainte sur une colonne
  - NOT NULL
  - PRIMARY KEY
  - UNIQUE
  - REFERENCES nom-table [(nom-col)] [action]
  - CHECK (condition)
- contrainte-table : contraintes sur une table
  - PRIMARY KEY (nom-col*)
  - UNIQUE (nom-col*)
  - FOREIGN KEY (nom-col*) REFERENCES nom-table [(nom-col*)] [action]
  - CHECK (condition)

Attribut obligatoire : NOT NULL

- Contrainte sur une colonne

- CREATE TABLE Pays

  (nom VARCHAR(20) NOT NULL ,

  capitale VARCHAR(20) NOT NULL ,

  surface INTEGER,

  ...)

Identifiants : clé primaire / secondaire

- PRIMARY KEY (A1, A2, ...)
  - La clé primaire (si elle existe) . En SQL les lignes doubles sont possibles !
  - choisir l'identifiant le plus efficace
  - attribut référencé par défaut dans les identifiants externes
  - pas de valeur nulle possible

    c-à-d NOT NULL automatiquement
- UNIQUE (A1, A2, ...)
  - une clé secondaire (s'il en existe)
  - contrainte d'intégrité pour les autres identifiants
  - valeur nulle permise (sauf si NOT NULL)

PRIMARY KEY : exemples

- CREATE TABLE Pays

  (nom VARCHAR(20) PRIMARY KEY ,

  capitale VARCHAR(20) ...)

- CREATE TABLE Employé

  (nom VARCHAR(30) ,

  prénom VARCHAR(30) ,

  adresse VARCHAR(60) , ...

  CONSTRAINT Pk_emp PRIMARY KEY (nom, prénom))

- contrainte de colonne et de table

UNIQUE : exemples

- CREATE TABLE Etudiant

  (AVS CHAR(11) PRIMARY KEY ,

  N°Etudiant CHAR(6) UNIQUE ,

  nom VARCHAR(20) ,

  prénom VARCHAR(30) , ...

  CONSTRAINT UNIQUE (nom, prénom))

- Contrainte de colonne et de table

- PRIMARY KEY et UNIQUE sont incompatibles

Clé externe : FOREIGN KEY

- CREATE TABLE Etudiant (N°E ...)
- CREATE TABLE Cours (NomCours ...)
- CREATE TABLE Suit

  (N°Etud CHAR(9) ,

  NomC VARCHAR(25) ,

  PRIMARY KEY (N°Etud , NomC) ,

  FOREIGN KEY (N°Etud) REFERENCES Etudiant ,

  FOREIGN KEY (NomC) REFERENCES Cours)

FOREIGN KEY (suite)

- Les clés externes référencent par défaut la clé primaire de la table référencée
  - CREATE TABLE Employé
    (AVS CHAR(11) PRIMARY KEY,
    empN° CHAR(6) UNIQUE , ...)
  - CREATE TABLE Département
    (dpt_id VARCHAR(18) PRIMARY KEY,
    manager_id CHAR(11) REFERENCES Employé , ...)
- Une clé externe peut référencer une clé secondaire de la table référencée => à préciser
  - CREATE TABLE Département2
    (dpt_id VARCHAR(18) PRIMARY KEY,
    manager_id CHAR(6) REFERENCES Employé (empN°) , ...)

Intégrité référentielle

- REFERENCES nom_table [(nom-col)] [action]
  - Qu'est ce qui se passe quand on détruit/m.à.j. une clé primaire ou unique qui est référencée par un tuple (foreign key) d'une autre table?

  - CREATE TABLE Département
    (dpt_id VARCHAR(18) PRIMARY KEY,
    manager_id CHAR(11) REFERENCES Employé,...)

- Soit le tuple (dpt_id=Ventes, manager_id=12345, ...) dans la table Département
  Que se passe-t-il si on détruit l'employé d'AVS 12345 dans la table Employé ?

"Referential triggered action"

- Deux circonstances
  - ON DELETE
  - ON UPDATE
- Trois options
  - SET NULL
  - SET DEFAULT: valeur par défaut si elle existe, sinon NULL
  - CASCADE : on répercute la m.à.j.

- CREATE TABLE Département

```
( dpt_id VARCHAR(18) PRIMARY KEY,
 manager_id CHAR(11) REFERENCES Employé (emp_id)
        ON DELETE SET NULL
        ON UPDATE CASCADE ,
... )
```

- Si la clause n'existe pas : refus

Contraintes (rappel)

- contrainte-col : contrainte sur une colonne
  - NOT NULL
  - PRIMARY KEY
  - UNIQUE
  - REFERENCES nom-table [(nom-col)] [action]
  - CHECK (condition)
- contrainte-table : contraintes sur une table
  - PRIMARY KEY (nom-col*)
  - UNIQUE (nom-col*)
  - FOREIGN KEY (nom-col*) REFERENCES nom-table [(nom-col*)] [action]
  - CHECK (condition)

Contrainte CHECK (condition)

- Condition que chaque ligne de la table doit vérifier
- Contrainte de colonne et de table

```
◆ CREATE TABLE Employé
  ( AVS CHAR(11) PRIMARY KEY ,
    nom VARCHAR(20)  NOT NULL,
    prénoms VARCHAR(30) ,
   âge NUMBER CHECK (âge BETWEEN 16 AND 70) ,
   sexe CHAR CHECK (sexe IN ('M', 'F')) ,
   salaire NUMBER ,
   commission NUMBER ,
   CONSTRAINT check_sal
       CHECK (salaire * commission <= 7000) )
```

CREATE TRIGGER

- Contrainte d'intégrité
  - ◆simple => clause CHECK dans CREATE TABLE
  - ◆complexe => un TRIGGER
- CREATE TRIGGER
  - ◆nouvelle instruction
  - ◆QUAND événement
    - INSERT / DELETE /UPDATE

  SI condition-SQL

  ALORS action
    - refus / instructions SQL

TRIGGER : exemple

- Institut FormaPerm : Pour tout tuple de Prérequis <nomC, nomCprérequis>, le cycle de nomCprérequis dans Cours doit être inférieur ou égal à celui de nomC

  QUAND : INSERT INTO Prérequis

  SI : le cycle de nomCprérequis dans Cours est supérieur à celui de nomC

  ALORS : refuser l'insertion

- Format à voir en TP

DROP TABLE

- DROP : supprimer une table
  - supprime la table et tout son contenu

- DROP TABLE nom_table [CASCADE CONSTRAINTS]

- CASCADE CONSTRAINTS
  - Supprime toutes les contraintes de clé externe référençant cette table
  - Si on cherche à détruire une table dont certains attributs sont référencés sans spécifier CASCADE CONSTRAINT: refus

ALTER TABLE

- Modifier la définition d'une table :
  - Changer le nom de la table
    mot clé : RENAME
  - Ajouter une colonne ou une contrainte

    mot clé : ADD
  - Modifier une colonne ou une contrainte
    mot clé : MODIFY
  - Supprimer une colonne ou une contrainte
    mot clé : DROP
  - renommer une colonne ou une contrainte
    mot clé : RENAME

ALTER TABLE (format)

- **ALTER TABLE** nom-table

{ RENAME TO nouveau-nom-table |

ADD ([(nom-col type-col [DEFAULT valeur]
[contrainte-col])*] |

MODIFY (nom-col [type-col] [DEFAULT valeur]
[contrainte-col])* |

DROP COLUMN nom-col [CASCADE CONSTRAINTS] |

RENAME COLUMN old-name TO new-name
}

Exemple : Institut FormaPerm

- CREATE TABLE Personne

```
( n°P  INTEGER NOT NULL ,
nom VARCHAR(30)  NOT NULL ,
adr VARCHAR(100)  NOT NULL ,
PRIMARY KEY (n°P )  )
```

- CREATE TABLE PersonnePrénoms

```
( n°P  INTEGER NOT NULL ,
prénom VARCHAR(30)  NOT NULL ,
n°prénom  INTEGER NOT NULL ,
PRIMARY KEY (n°P , n°prénom ) ,
UNIQUE (n°P , prénom ) ,
FOREIGN KEY (n°P)  REFERENCES Personne  ON DELETE
  CASCADE )
```

Exemple FormaPerm (2)

- CREATE TABLE Etudiant

```
( n°P    INTEGER NOT NULL ,
n°E    INTEGER NOT NULL ,
dateN  DATE  NOT NULL ,
PRIMARY KEY (n°E ) ,
UNIQUE (n°P ) ,
FOREIGN KEY  (n°P)  REFERENCES Personne )
```

- CREATE TABLE EtudiantEtudes

```
( n°E    INTEGER NOT NULL ,
année  INTEGER  NOT NULL ,
diplôme  VARCHAR(30)   NOT NULL ,
PRIMARY KEY (n°E , diplôme ) ,
FOREIGN KEY  (n°E)  REFERENCES Etudiant ON DELETE CASCADE
  )
```

Exemple FormaPerm (3)

- CREATE TABLE Enseignant

```
( n°P    INTEGER NOT NULL ,
tel    INTEGER  NOT NULL ,
statut  VARCHAR(30)  NOT NULL ,
banque  VARCHAR(30)  NOT NULL ,
agence   VARCHAR(30)  NOT NULL ,
compte  INTEGER  NOT NULL ,
PRIMARY KEY (n°P ) ,
FOREIGN KEY  (n°P)  REFERENCES Personne )
```

- CREATE TABLE Cours

```
( nomC  VARCHAR(30)  NOT NULL ,
cycle  INTEGER NOT NULL ,
n°Ens  INTEGER NOT NULL ,
PRIMARY KEY (nomC ) ,
FOREIGN KEY  (n°Ens)  REFERENCES Enseignant )
```

Exemple FormaPerm (4)

- CREATE TABLE Obtenu

```
( n°E   INTEGER NOT NULL ,
nomC  VARCHAR(30)  NOT NULL ,
note  NUMERIC(2,1)  NOT NULL ,
année  INTEGER  NOT NULL ,
PRIMARY KEY (n°E , nomC ) ,
FOREIGN KEY (n°E)  REFERENCES Etudiant.n°E ,
FOREIGN KEY (nomC)  REFERENCES Cours )
```

- CREATE TABLE Inscrit

```
( n°E   INTEGER NOT NULL ,
nomC  VARCHAR(30)  NOT NULL ,
PRIMARY KEY (n°E , nomC ) ,
FOREIGN KEY (n°E)  REFERENCES Etudiant.n°E ,
FOREIGN KEY (nomC)  REFERENCES Cours )
```

Exemple FormaPerm (5)

- CREATE TABLE Prérequis

  (nomC VARCHAR(30) NOT NULL ,

  nomCprérequis VARCHAR(30) NOT NULL ,

  PRIMARY KEY (nomC , nomCprérequis) ,

  FOREIGN KEY (nomC) REFERENCES Cours ,

  FOREIGN KEY (nomCprérequis) REFERENCES Cours)