

Base de données

Séance 4

Base de données relationnelle :
Langage de définition de données (LDD)

Du MCD à la base de données

Modèle conceptuel
de données
(MCD)

Diagramme
de classes

On peut appliquer
des règles de
conversion

Modèle logique de données
(MLD)

Transcription en langage SQL

Script de création des tables
(BDD)

Base de données relationnelle (BDD)

- Les données sont organisées sous forme de **tables** à 2 dimensions dont
 - les lignes sont appelées n-uplet ou enregistrements
 - les colonnes correspondent aux attributs
 - Les données sont manipulées par des opérateurs de l'algèbre relationnelle
 - L'état cohérent de la base est défini par un ensemble de contraintes d'intégrité
 - Le modèle de données est réflexif :
 - Le dictionnaire de données décrit les données sous forme d'une base de données
 - On l'appelle la métabase
-

Base de données : les objets

□ Table

- Une table enregistre des enregistrements qui décrivent une instance d'une entité

□ Vue

- Les vues sont des résultats d'exploration de données que l'on fait apparaître comme une table.

□ Index

- Un index est une table d'encodage qui optimise l'accès aux données

□ User

- Un objet User représente un utilisateur des données.
-

Les clefs : primaire et étrangère

□ Clé primaire :

- permet d'identifier de manière unique **un** enregistrement dans **la** table

□ Clé étrangère :

- permet de vérifier que le champ contient une clef primaire existante d'une autre table.

« *La notion de clef étrangère est directement liée à la notion d'association* »

Langage de définition des données (LDD)

□ **Le LDD sert à décrire la nature des données telles qu'elles vont exister**

□ **CREATE**

Créer des objets de données (des tables, des indexes, des vues)

□ **DROP**

Supprimer des objets de données

□ **ALTER**

Modifier la définition des objets de données

Le « moment » d'utilisation du LDD

- Le LDD s'utilise au moment de la constitution de la base de données
 - Les instructions DROP s'utilisent principalement :
 - Pour supprimer une table obsolète
 - Avant de recréer une nouvelle version d'une table
 - Les instructions ALTER s'utilisent :
 - Pour mettre à jour la structure d'un objet sans perte de données (mise à jour progressive)
-

SQL : Structured Query Language

- Le SQL est un langage standardisé qui regroupe les quatre sous-langages :
 - LDD : langage de définition de données
CREATE, DROP, ALTER
 - LMD : langage de manipulation des données
SELECT, UPDATE, INSERT, DELETE
 - LCD : langage de contrôle de données
GRANT, REVOKE
 - LCT : langage de contrôle de transactions
COMMIT, SAVEPOINT, ROLLBACK
-

Objet TABLE : Définition

- Structure de données composée de :
 - colonnes
 - lignes appelées n-uplets ou **enregistrements**
 - Chaque colonne doit être :
 - nommée
 - typée
 - Exemple : table personne, étudiant ...
-

Exemple

Personne (numsecu : NUMBER, nom : CHAR(50), prenom : CHAR(50))

| numsecu | nom | prenom |
|-----------------|------------|---------------|
| 354338532195874 | Durand | Caroline |
| 345353545435811 | Dubois | Jacques |
| 173354684513546 | Dupont | Lisa |
| 973564213535435 | Dubois | Rose-Marie |

L'instruction CREATE TABLE

- ❑ **Attribuer un nom à la table** (le nom doit être unique dans la base de données)
 - ❑ **Attribuer des colonnes (de 1 à 254 colonnes)**
 - Le nom de la colonne est unique dans la table
 - Attribuer un type à chaque colonne
 - Définir des contraintes sur ces colonnes
 - ❑ Spécifier des caractéristiques de stockage
 - ❑ Spécifier une requête pour créer la table avec des données
-

L'instruction CREATE TABLE : Syntaxe

CREATE TABLE <nom>

(<définition colonne> | <définition
contrainte>,...)

[<spécification stockage>]

[<données provenant d'une requête>]

L'instruction CREATE TABLE :

Exemple

```
CREATE TABLE personne(  
    nom          VARCHAR2(30),  
    age          INTEGER,  
    salaire     NUMBER(10,2)  
);
```

Définition de colonne : Les types

- Syntaxe générale de type :
 - *<nom type>[(<dimensionnement>)]*
 - Type alphanumérique :
 - CHAR(n) : chaîne à longueur fixe
 - VARCHAR2(n) : chaîne à longueur optimisée < 255 caractères
 - LONG : texte long > 255 caractères
 - Exemple :
 - nom VARCHAR2(30)
-

Définition de colonne : Les types

- Types numériques :
 - NUMBER(p,s): nombre décimal
 - INTEGER : entier long
 - SMALLINT : entier court
 - FLOAT : réel à virgule flottante

 - Exemples :
 - age SMALLINT
 - salaire NUMBER(10,2)
-

Définition de colonne : Les types

- Type de gestion du temps et des dates :
 - DATE : date du calendrier grégorien
 - TIMESTAMP : combiné date temps sous forme compacte : 20091109101012

 - Exemple
 - date_commande DATE
-

Définition de colonne : Les types

- Les types « gros objets »
 - CLOB : pour les chaînes de caractères
 - BLOB : pour les données binaires
 - BFILE : pour les données stockées dans des fichiers externes binaires.

La taille maximale est de 4 gigabytes

Les contraintes

- Contrainte de colonne
 - Associée à une colonne de la table
- Contrainte de table
 - Associée à plusieurs colonnes de la table, ou à la totalité des colonnes

« Il est utile de donner aux contraintes un nom qui exprime la raison de la contrainte »

Les contraintes des colonnes

- S'applique sur **une seule** colonne
 - Nullité de colonne : **NULL / NOT NULL**
 - Unicité de valeur dans une colonne : **UNIQUE**
 - Clé primaire : **PRIMARY KEY**
 - Vérification sur un ensemble de valeur : **CHECK**
 - Clé étrangère : **REFERENCES**

« Une contrainte définit des restrictions sur les valeurs possibles »

Exemple de contraintes de colonnes

```
CREATE TABLE personne(  
    id NUMBER CONSTRAINT pk_per PRIMARY KEY,  
    nom VARCHAR2(20),  
    salaire NUMBER(12,2) CONSTRAINT const_sal  
    NOT NULL,  
    codepost NUMBER(5) CHECK(codepost BETWEEN  
    00001 AND 99999)  
);
```

Valeur par défaut d'une colonne

- ❑ Permet d'attribuer une **valeur par défaut** à une colonne si une requête d'ajout de données ne la fournit pas.
- ❑ Sans spécification la valeur par défaut est NULL
- ❑ Peut être une valeur, une valeur calculée standard,...

```
date_jour DATE [DEFAULT] current_date
```

Les contraintes de table

- S'applique sur plusieurs colonnes
 - Clé primaire, si elle est « composite »
 - Clé étrangère, si elle est « composite »
-

Contraintes de table

- ❑ PRIMARY KEY (colonne1, colonne2, ...)
 - désigne la concaténation des attributs cités comme clé primaire de la table
 - ❑ UNIQUE (colonne1, colonne2, ...)
 - désigne la concaténation des attributs cités comme clé secondaire de la table
 - ❑ FOREIGN KEY (colonne1, colonne2, ...)
REFERENCES table [(colonne1, colonne2,...)]
 - ❑ CHECK (condition)
 - ❑ ON DELETE CASCADE / ON DELETE SET NULL
-

Exemple de contraintes de table

```
CREATE TABLE personne(  
    code VARCHAR2(4),  
    nom VARCHAR2(20),  
    salaire NUMBER(12,2),  
    codepost CHAR(5),  
    CONSTRAINT pk_personne PRIMARY KEY  
    (code,nom)  
);
```

Modification de la structure d'une table

- ❑ On pourrait supprimer la table et la recréer

...?

On perd les données qui sont déjà dedans

Modification de la structure d'une table

- Modifier la structure, c'est :
 - supprimer, modifier les caractéristiques ou ajouter des colonnes
 - supprimer, ajouter, modifier des contraintes
- Syntaxe générale

```
ALTER TABLE <nom_table>  
    [<add>][<modify>][<drop>]
```

Exemples de modification

```
ALTER TABLE table1 ADD (col1 type1, col2 type2, ...);
```

```
ALTER TABLE table1 MODIFY (col1 type1, col2 type2,  
...);
```

```
ALTER TABLE NomTable DROP CONSTRAINT NomCont;
```

```
ALTER TABLE nom_table ADD CONSTRAINT nom_contrainte  
FOREIGN KEY (NomCol1, ..., NomColn) REFERENCES  
NomTable(Col1, ..., Coln);
```

```
ALTER TABLE nom_table ADD CONSTRAINT nom_contrainte  
PRIMARY KEY (NomCol1, NomCol2, ...);
```

Destruction d'une table

- Syntaxe générale :

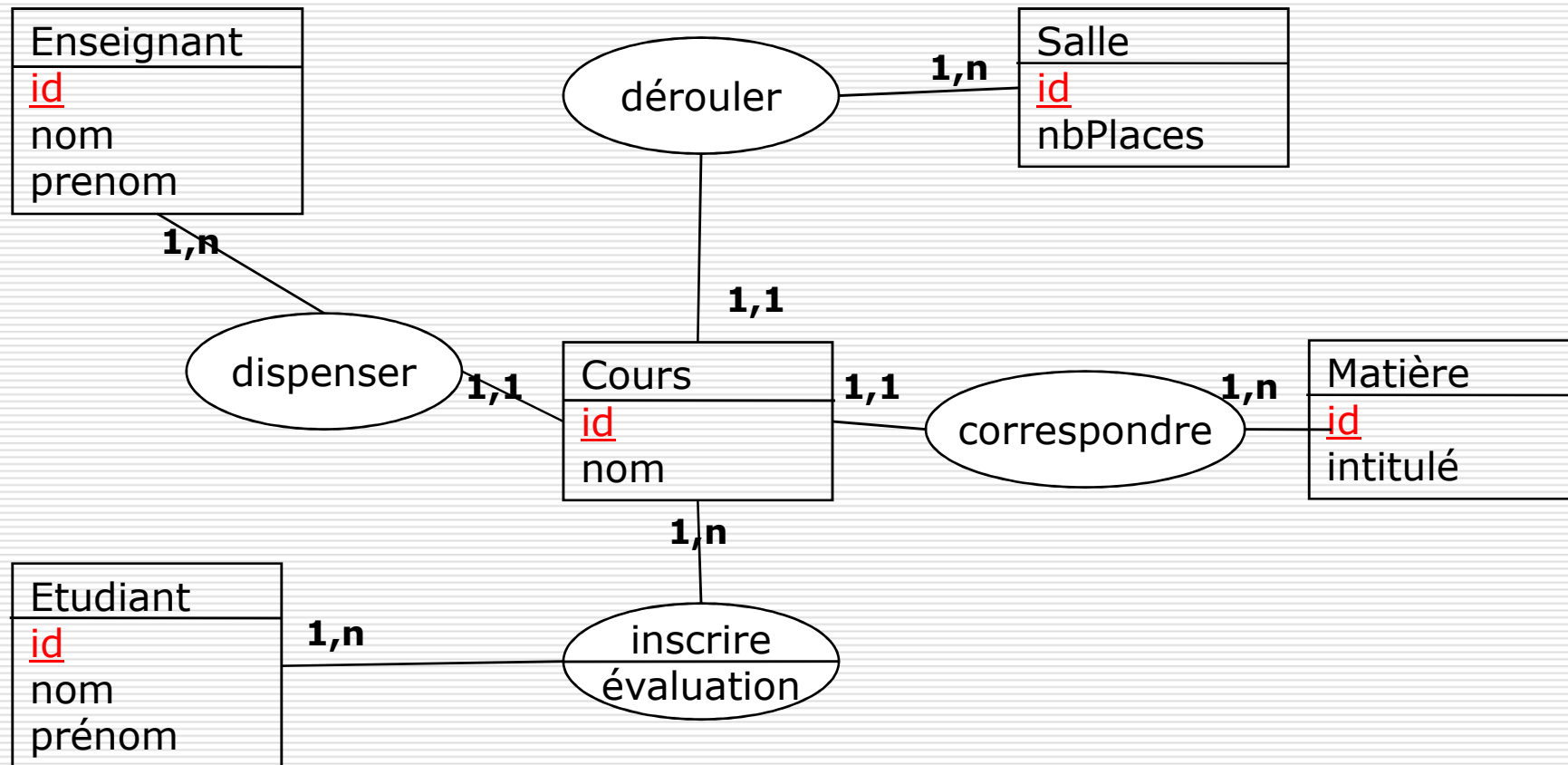
```
DROP TABLE <nom_table>
```

- Exemple :

```
DROP TABLE personne ;
```

- Problèmes liés à la destruction d'une table :
 - Certaines contraintes peuvent interdire la destruction d'une table
-

Exemple : MCD Cours



MLD !

- Enseignant(id, nom, prénom)
 - Etudiant(id, nom, prénom)
 - Salle(id, nbPlaces)
 - Matière(id, intitulé)
 - Cours(id, nom, *#idEnseignant*, *#idMatière*, *#idSalle*)
 - Inscription(*#idEtudiant*, *#idCours*, évaluation)
-

BDD

```
CREATE TABLE enseignant(  
    id NUMBER CONSTRAINT pk_enseignant PRIMARY KEY,  
    nom VARCHAR2(30),  
    prenom VARCHAR2(30));
```

```
CREATE TABLE etudiant(  
    id NUMBER CONSTRAINT pk_etudiant PRIMARY KEY,  
    nom VARCHAR2(30),  
    prenom VARCHAR2(30));
```

```
CREATE TABLE salle(  
    id NUMBER CONSTRAINT pk_salle PRIMARY KEY,  
    nbPlaces NUMBER CHECK (nbPlaces > 0));
```

BDD

```
CREATE TABLE matiere(  
  id NUMBER CONSTRAINT pk_matiere PRIMARY KEY,  
  intitule VARCHAR2(30));
```

```
CREATE TABLE cours(  
  id NUMBER CONSTRAINT pk_cours PRIMARY KEY,  
  nom VARCHAR2(30),  
  idEnseignant CONSTRAINT fk_cours1 REFERENCES enseignant(id),  
  idMatiere CONSTRAINT fk_cours2 REFERENCES matiere(id),  
  idSalle CONSTRAINT fk_cours3 REFERENCES salle(id));
```

```
CREATE TABLE inscription(  
  idEtudiant CONSTRAINT fk_inscription1 REFERENCES  
    etudiant(id),  
  idCours CONSTRAINT fk_inscription2 REFERENCES cours(id),  
  evaluation NUMBER(2,1) CONSTRAINT eval NOT NULL,  
  CONSTRAINT pk_inscription PRIMARY KEY (idEtudiant,idCours));
```
