Bases de Données relationnelles et leurs systèmes de Gestion

RAPPELS

- **Contraintes d'intégrité sous Oracle**
- **Notion de vue**

III.1- Définition de schémas

1. Typage des attributs
2. Contrainte d'intégrité
   1. Intra-relation
   2. Inter-relations (CI Référentielle)
3. Conduite à tenir si violation

Syntaxe de CREATE TABLE (1/2)

```
CREATE TABLE nomTable
      (définitionDattribut
      [,définitionDattribut]…
      [,définitionDeContrainte]…));
```

définitionDattribut

```
nomAttribut  [typeDeDonnées|domaine]
      [DEFAULT valeur][NULL|NOT NULL]
      [UNIQUE|PRIMARY KEY]
      [REFERENCES nomTableBis(nomAttributBis)]
      [[CONSTRAINT nomContrainte]
                  CHECK (condition)]
```

NULL Autorisé ou pas

Typage (domaine)

Contraintes

Syntaxe de CREATE TABLE (2/2)

Clé Primaire

Nommer une contrainte

Inclusion/Référence

```
[CONSTRAINT nomContrainte]
{PRIMARY KEY listeAttributs |
 FOREIGN KEY listeAttributs REFERENCES
    nomTableBis[listeAttributs]
       [ON DELETE {NO ACTION|CASCADE|SET NULL|
                                SET DEFAULT}]
     [ON UPDATE {NO ACTION|CASCADE|SET NULL|
                                SET DEFAULT}] |
 CHECK (condition)
 }
```

Que faire si…

Autres contraintes

III.1.2- Contraintes d'intégrité (CI)

- Plusieurs types de contraintes statiques déclarables avec **CREATE TABLE**
  - Contraintes sur le **domaine** d'un attribut
  - Contraintes de **clés** (primaire, candidate)
  - Contraintes **d'intégrité référentielle** (contrainte d'**inclusion** et **clé étrangère**)
- D'autres contraintes peuvent être programmées et/ou porter sur des changements d'états de la BD, sur plusieurs tables…
  - **CREATE TRIGGER** (non traité ici)

CI sur le domaine d'un attribut (1/5)

- Type de données (number, char…)
- Contrainte **NOT NULL**
  - Valeur obligatoire pour l'attribut (si pas de DEFAULT)

```
CREATE TABLE Client
 (noClient      INTEGER        NOT NULL,
  nom           VARCHAR(50)    NOT NULL,
  ddn           VARCHAR(15)    NULL,
  téléphone     VARCHAR(15) )
```

- Par défaut : NULL
  - Insertion de la valeur NULL si absence de valeur et de default

CI sur le domaine d'un attribut (2/5)

- Contrainte CHECK sur un attribut
- CHECK (Expression logique)
- Exemple: numéro de client ∈ [0..1000]

```
CREATE TABLE Client
 (noClient        INTEGER          NOT NULL
   CHECK (noClient>0 AND noClient<1000),
  nom    VARCHAR(50)      NOT NULL,
  ddn    VARCHAR(15)      NULL )
```


CI sur le domaine d'un attribut (3/5)

- Contrainte CHECK sur plusieurs attributs du même tuple (Contrainte intra-relation)
- *Les produits dont le numéro est supérieur à 100 ont un prix supérieur à 15 €.*

```
CREATE TABLE Produit
 (noProduit       INTEGER  NOT NULL,
  libellé         VARCHAR(50),
  prixUnitaire    NUMBER(10,2) NOT NULL,
  CHECK (noProduit<=100 OR prixUnitaire>15) )
```


CI sur le domaine d'un attribut (4/5)

- Création d'un domaine spécifique (CREATE DOMAIN)
- *Un employé/client est un homme ou une femme.*

```
CREATE DOMAIN domaineSexe AS
                CHAR(1) CHECK (VALUE IN ('M','F'))

CREATE TABLE employé
        (numEmp          INTEGER,
        genre    domaineSexe, … );
CREATE TABLE client
        (noClientINTEGER,
        genre    domaineSexe, … )
```

N.B. **Les domaines ne sont pas supportés dans Oracle.**

CI sur le domaine d'un attribut (5/5)

- Valeur d'un attribut par défaut (DEFAULT)
  - Si pas de valeur lors d'une insertion (NULL)
  - Ou attribut mis à NULL lors d'une modification
- Exemple: *numéro de téléphone : présence obligatoire et valeur "confidentiel" par défaut*

```
CREATE TABLE employé
 (numEmp        INTEGER,
  numTel        VARCHAR(15)   NOT NULL
        DEFAULT 'confidentiel', … )
```


III.1.2- Contraintes d'intégrité (CI)

- Plusieurs types de contraintes statiques déclarables avec **CREATE TABLE**
  - Contraintes sur le domaine d'un attribut
  - Contraintes de clé
    - Primaire
    - Candidate
  - Contraintes d'intégrité référentielle (contrainte d'inclusion et clé étrangère)

Contrainte de clé primaire

- Deux tuples ne peuvent pas avoir la même valeur de la clé
- Le(s) attribut(s) clé(s) ne peuvent être NULL
- *noClient: clé primaire de la table Client* (clé atomique)

```
CREATE TABLE Client
 (noClient  INTEGER     PRIMARY KEY,
  nom       VARCHAR(50) NOT NULL,
  ddn       VARCHAR(15) NULL );


CREATE TABLE Client
 (noClient  INTEGER ,
  nom       VARCHAR(50) NOT NULL,
  ddn       VARCHAR(15) NULL,
  PRIMARY KEY (noClient))
```

Contrainte de clé : clé composée

- *noClient x noProduit x dateCde = clé primaire de la table Commande*

```
CREATE TABLE Commande
        (noClientINTEGER,
        noProduit          INTEGER,
        dateCommande DATE,
        quantité INTEGER,
        PRIMARY KEY     (noClient, noProduit, dateCommande)
        )
```

- Lorsque la clé est composée de plusieurs attributs, définir la contrainte PRIMARY KEY au niveau de la table

Clé Candidate et Unicité

- Une seule clé primaire par table mais d'autres attributs peuvent avoir des valeurs uniques pour chaque tuple (exemple: clés candidates)
- Clause **UNIQUE**

```
CREATE TABLE Citoyen
        (numSécu        INTEGER PRIMARY KEY ,
        nom             VARCHAR(50),
        prénom          VARCHAR(50),
        ddn             DATE  NULL,    --date de naissance
        noPassport      INTEGER NULL  UNIQUE )
```


Contrainte d'intégrité référentielle (FOREIGN KEY / REFERENCES)

Contrainte inter-relations

- *Émetteur d'une Commande doit être un Client connu*

```
CREATE TABLE Commande
        (noCommande     INTEGER PRIMARY KEY,
        noClient INTEGER,          noProduit  INTEGER,
        dateCommande DATE,         quantité   INTEGER,


        );
```

- Tables ***Client*** et ***Produit*** déjà définies : pas de référence en avant
- Cible d'une référence : PRIMARY KEY ou UNIQUE
- Ajout de commandes autorisée : si le client et le produit existent

III.1- Définition de schémas

1. Typage des attributs
2. Contrainte d'intégrité
   1. Intra-relation
   2. Inter-relations (CI Référentielle)
3. Conduite à tenir si violation de CI référentielle

III.1.3- CI référentielle : conduite à tenir en cas de mise à jour

```
[CONSTRAINT nomContrainte]
FOREIGN KEY listeAttributs REFERENCES
      nomTableBis(listeAttributs)
[ON DELETE {NO ACTION|CASCADE|SET NULL| SET DEFAULT}]
[ON UPDATE {NO ACTION|CASCADE|SET NULL| SET DEFAULT}]
```


CI référentielle : conduite à tenir en cas de mise à jour (DELETE)

Tentative de suppression de la clé primaire

```
CREATE TABLE Commande
(noCommande INTEGER PRIMARY KEY, noClient INTEGER,
                    …
 FOREIGN KEY (noClient) REFERENCES Client(noClient))

DELETE FROM Client WHERE noClient=45
```

- **Que deviennent les commandes du client numéro 45 ?**
- 4 options :
  - NO ACTION
  - CASCADE
  - SET NULL
  - SET DEFAULT

CI référentielle : conduite à tenir en cas de suppression de la clé primaire

- Suppression du client 45 : que faire de ses éventuelles commandes ?
- NO ACTION : suppression du client 45 refusée
- CASCADE :
  1.
  2.
- SET NULL :
  1.
  2.
- SET DEFAULT :
  1.
  2.

CI référentielle : conduite à tenir en mise à jour (UPDATE)

- Tentative de modification de la clé primaire

```
CREATE TABLE Commande
(noCommande INTEGER PRIMARY KEY,
 noClient   INTEGER,  …
 FOREIGN KEY (noClient) REFERENCES Client(noClient))

 UPDATE Client SET noClient=100 WHERE noClient=45
```

- ***Que deviennent les commandes du client numéro 45 ?***

CI référentielle : conduite à tenir en mise à jour (UPDATE)

- NO ACTION :
- CASCADE :
  1.
  2.
- SET NULL :
  1.
  2.
- SET DEFAULT :
  1. noClient = valeur par défaut si celle-ci est définie
  2. Modifier le client

CI référentielle : conduite à tenir en cas de mise à jour

```
CREATE TABLE Commande
(noCommande INTEGER PRIMARY KEY,
 noClient   INTEGER,  …
 FOREIGN KEY (noClient) REFERENCES Client(noClient)
  ON DELETE CASCADE
  ON UPDATE CASCADE )
```

- Sous Oracle
  - par défaut :
    - ON DELETE NO ACTION
    - ON UPDATE NO ACTION
  - seules possibilités dans CREATE TABLE
    - ON DELETE CASCADE
    - ON DELETE SET NULL

III.3. Modification de schémas (1/3)

- Ajouter un attribut (nom, type, défaut, NOT NULL uniquement) ou une contrainte
- Modifier ou supprimer le type ou la valeur par défaut d'un attribut
- Supprimer un attribut ou une contrainte
- Supprimer une contrainte: nommée, clé primaire, unicité
- Impact possible sur les programmes !!!!

III.3. Modification de schémas (1/3)

ALTER TABLE nomTable
ADD (attribut type [DEFAULT valeur] [contrainte]) |
MODIFY (attribut [type] [DEFAULT valeur] [contrainte]) |
DROP COLUMN attribut |
ADD [**CONSTRAINT** nom] contrainte |
DROP {{**PRIMARY KEY** | **UNIQUE**} (attribut) | **CONSTRAINT** nom}

- Renommage de table
  RENAME ancienNomDeTable **TO** nouveauNomDeTable

III.3. Modification de schémas (2/3)

```
ALTER TABLE client ADD (TEL_PORTABLE CHAR(10))

ALTER TABLE client MODIFY (ADR_CLIENT CHAR(70))

ALTER TABLE produit ADD CONSTRAINT PRIX_POSITIF
check (PRIXUNITAIRE >= 0)

ALTER TABLE produit DROP CONSTRAINT PRIX_POSITIF

ALTER TABLE client  DROP COLUMN TEL_PORTABLE

RENAME CLIENT TO ACHETEUR
```


III.3. Modification de schémas (3/3)

- ALTER TABLE et désactivation d'une contrainte nommée

```
ALTER TABLE nom_table
DISABLE CONSTRAINT nom_contrainte;
```

- ALTER TABLE et activation d'une contrainte nommée

```
ALTER TABLE nom_table
ENABLE CONSTRAINT nom_contrainte;
```


III.4. Suppression d'un schéma (DROP TABLE)

DROP TABLE nom-relation
[**CASCADE CONSTRAINTS**]

- Suppression
  1. des **tuples** ET du **schéma** de la relation
  2. des **index**
  3. désactivation des **vues** liées
  4. des contraintes référentielles éventuelles (dans les relations où la clé primaire de la relation à supprimer est référencée)

Le langage SQL (SGBD ORACLE)

IV. Les Vues (1/3)

- Vue : relation calculée
  - ses tuples ne sont pas stockés
  - sa requête de définition est stockée
  - ses tuples sont générés à chaque appel de la vue

```
CREATE [OR REPLACE] VIEW nom-vue[(attribut(s)]
  AS  (commande SELECT de définition)
```

- *Définir, comme une vue* ProduitsChers*, la liste des produits dont le prix dépasse 5000€. Mettre dans la vue les colonnes nomProduit et prixUnitaire*

Les Vues (2/3)

- Une fois créée, une vue est utilisable comme toute autre table
- *Liste des produits chers dont le libellé comporte 'de luxe'*

```
SELECT *     FROM ProduitsChers
WHERE        nomProduit  like '%de luxe%'

SELECT *  FROM  ProduitsChers pc, Stock s
Where ….
```

- Suppression d'une vue

```
DROP VIEW nom-vue
```

Les Vues (3/3): intérêt

- Masquage des opérations de jointure
- Sauvegarde (indirecte) de requêtes complexes
- Support de l'indépendance logique:

  si les tables sont modifiées, les vues doivent être réécrites mais les requêtes utilisant les vues ne subissent pas de changement
- Support de la confidentialité en combinaison avec des privilèges d'accès adéquats:

  Masquer les lignes ou colonnes pour les utilisateurs non autorisés

SQL : Conclusion

- Langage commun aux SGBDR
- Déclaratif, Orienté ensembles
- Fondement : Algèbre + Calcul relationnel
- Définition & Manipulation :
  - Schémas : CREATE, ALTER, DROP
  - Instances : SELECT, INSERT, UPDATE, DELETE
- Extensions de SQL :
  - Fonctions d'agrégation
  - Procédurales (si alors sinon, tant que, etc.)