SGBD
2ème partie

Chapitre 1 : Définition des données

Chapitre 2 : Maintien de la cohérence

Chapitre 3 : Gestion des utilisateurs

Chapitre 4 : Gestion de la confidentialité

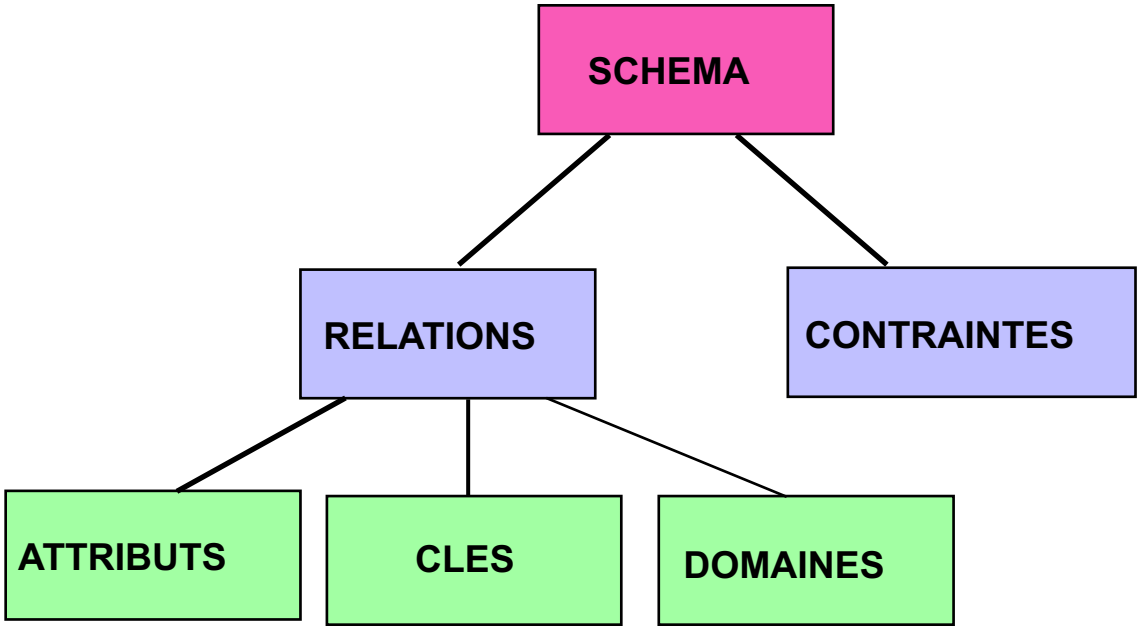
Chapitre 1 : DEFINITION DES DONNEES

- Catalogue d'une base de données (Métabase)

- Langage de définition des données : compléments

Chapitre 1 : Structuration d'une BD (SQL 92)

Un schéma est associé à un utilisateur ou une application. Il regroupe toutes les relations et les éléments associés (contraintes d'intégrité, vues, index …).



Un schéma est identifié par un nom et l'identifiant du propriétaire du schéma.

Chapitre 1 : Catalogue (Métabase) (1)

• Un SGBD doit connaître les informations sur les différents objets et structures qu'il contient : structure des relations, type des attributs, contraintes d'intégrité …

• Ces informations constituent le CATALOGUE du SGBD

• Elles sont stockées dans des relations particulières qui sont manipulées par le SGBD lui-même. L'ensemble de ces relations, appelé METABASE, est regroupé dans un schéma particulier : INFORMATION_SCHEMA

• SQL normalise les tables de la métabase

Chapitre 1 : Catalogue (Métabase) (2)

- **STRUCTURE TYPIQUE DU CATALOGUE (NOMS STANDARD SQL92)**

  USERS : liste des identifiants des utilisateurs

  SCHEMATA : liste des schémas avec l'identifiant du propriétaire

  TABLES: liste des relations

  COLUMNS: liste des attributs des relations

  DOMAINS : liste des domaines des attributs

  TABLE_CONSTRAINTS : liste des clés primaires sur les relations

  REFERENTIAL_CONSTRAINTS : liste des contraintes référentielles

  KEY_COLUMNS_USAGE : liste des clés primaires et étrangères sur les relations

  CHECK_CONSTRAINTS : liste des contraintes générales

  …

Exemples :

TABLES(Table_Catalog, Table_Schema, Table_Name, Table_Type)

COLUMNS(Table_Catalog, Table_Schema, Table_Name, Column_Name, Ordinal_Position, Domain_Catalog, Domain_Schema, Domain_Name, Column_Default, IS_Nullable)

Chapitre 1 : Catalogue (Métabase) (3)

Commande (ncom, ncli, npro, qtecom, datecom)

Client (ncli, raisonSociale, NbCom)

Table_Catalog	Table_Schema	Table_Name	Table_Type
IUT	laleau	Commande	Base
IUT	laleau	Client	Base

T_C	T_S	T_N	C_N	O_P	D_C	D_S	D_N	C_D	IS_N
IUT	laleau	Commande	ncom	1	IUT	laleau	ident	NULL	False
IUT	laleau	Commande	ncli	2	IUT	laleau	ident	NULL	False
IUT	laleau	Commande	npro	3	IUT	laleau	ident	NULL	False
IUT	laleau	Commande	qtecom	4	NULL	NULL	NULL	NULL	False
IUT	laleau	Commande	datecom	5	IUT	laleau	DateCom	NULL	False
IUT	laleau	Client	ncli	1	IUT	laleau	ident	NULL	False
IUT	laleau	Client	raisonSociale	2	NULL	NULL	NULL	NULL	True
IUT	laleau	Client	NbCom	3	NULL	NULL	NULL	NULL	False

Chapitre 1 : Métabase(3)

LE CONTENU DE LA METABASE EST :

- MANIPULÉ par Le Langage de Définition de Données

- CONSULTABLE PAR DES COMMANDES AD HOC OU EN SQL

Exemple : **DESCRIBE Commande**

```
SELECT Column_Name, IS_Nullable
FROM Columns
WHERE Table_Name = 'Commande';
```

Chapitre 1 : LANGAGE DE DEFINITION DE DONNEES (syntaxe SQL 92)

TROIS TYPES DE COMMANDES

CREATE : CRÉATION D'OBJET

ALTER : MODIFICATION D'OBJET

DROP : SUPPRESSION D'OBJET

POUR LA MANIPULATION DU SCHEMA DES RELATIONS, L'INTÉGRITÉ, LES VUES, LES DROITS D'ACCÈS, LES STRUCTURES PHYSIQUES

CRÉATION DE RELATION : DÉFINIT LE SCHÉMA D'UNE RELATION

CREATE TABLE <nom_relation>

(<élément de relation> < , élément de relation>*

< , contrainte de relation>*

)

<élément de relation> ::= <nom_attribut> <type de données>

[<contrainte d 'attribut>*]

<type de données> ::= VARCHAR <longueur> | INT | REAL | DATE …

Chapitre 1 : Exemples de création de relations

```
CREATE TABLE commande
        ( numCom INT Primary Key,
          numCli INT references Client,
          numPro INT references Produit,
          dateCom DATE
        )
```

Cette commande génère des insertions de tuples dans les relations suivantes:

TABLES(Table_Catalog, Table_Schema, Table_Name, Table_Type)

```
INSERT INTO TABLES values ('IUT', 'laleau', 'commande', base);
```

COLUMNS(Table_Catalog, Table_Schema, Table_Name, Column_Name, Ordinal_Position, Domain_Catalog, Domain_Schema, Domain_Name, Column_Default, IS_Nullable)

```
INSERT INTO COLUMNS values ('IUT', 'laleau', 'commande', 'numCom', 1, 'IUT', 'laleau',
        ident, NULL, FALSE);
INSERT INTO COLUMNS values ('IUT', 'laleau', 'commande', 'numCli', 2, 'IUT', 'laleau',
        ident, NULL, FALSE);
```

SQL - LDD : Modification du schéma d'une relation

Un schéma de relation peut être modifié (ajout ou suppression d'attributs, de contraintes, …)

Syntaxe SQL 92

```
ALTER TABLE nom_relation
        ADD nom_attribut  type_attribut
        | ADD CONSTRAINT Contrainte sur relation
        | ALTER  nom_attribut
                (DROP DEFAULT |  SET DEFAULT value)
        | DROP COLUMN nom_attribut
        | DROP CONSTRAINT nom_contrainte
```

SQL - LDD : Modification du schéma d'une relation

EX :

AJOUT/ MISE A JOUR / SUPPRESSION D'UN ATTRIBUT

```
ALTER TABLE vin ALTER millésime SET DEFAULT 1991;

ALTER TABLE vin ADD prix INTEGER (2) ;

ALTER TABLE commande DROP COLUMN dateCom;
```

Au niveau de la métabase, cette dernière requête génère:

```
DELETE FROM COLUMNS
WHERE Catalog_Name = 'IUT'
        and Schema_Name = 'laleau'
        and Table_Name = 'commande'
        and Column_Name = 'dateCom';
```

SQL - LDD : Suppression de Relations

DROP TABLE nom_relation

supprime la relation et les tuples

EX : **DROP TABLE** Client;

Au niveau de la métabase:

```
DELETE FROM COLUMNS
        WHERE Catalog_Name = 'IUT'
                and Schema_Name = 'laleau'
                and Table_Name = 'Client';

DELETE FROM TABLES
        WHERE Catalog_Name = 'IUT'
                and Schema_Name = 'laleau'
                and Table_Name = 'Client';
```

Chapitre 2 : Maintien de la Cohérence

PERTE POSSIBLE DE LA COHÉRENCE DANS TROIS SITUATIONS

- **MISES À JOUR INTRODUISANT DES DONNÉES FAUSSES, INCOHÉRENTES PAR RAPPORT À LA DESCRIPTION DE LA BASE, OU INCOHÉRENTES PAR RAPPORT AUX DONNÉES DÉJÀ STOCKÉES**

===> CONTRÔLE DE L'INTÉGRITÉ

- **ACCÈS SIMULTANÉS, EN CONSULTATION ET EN MÀJ, PAR DES UTILISATEURS DIFFÉRENTS, AUX MÊMES DONNÉES**

===> CONTRÔLE DES ACCÈS SIMULTANÉS (vu en 2ème année)

- **PANNES (MACHINE, SYSTÈME, ERREUR DE MANIPULATION HUMAINE) ENTRAINANT LA PERTE DE LA MC OU DE LA MS**

===> PROCÉDURES DE REPRISE (vu en 2ème année)

Chapitre 2 : NOTIONS COMMUNES

- **CONTRAINTES D'INTÉGRITÉ**

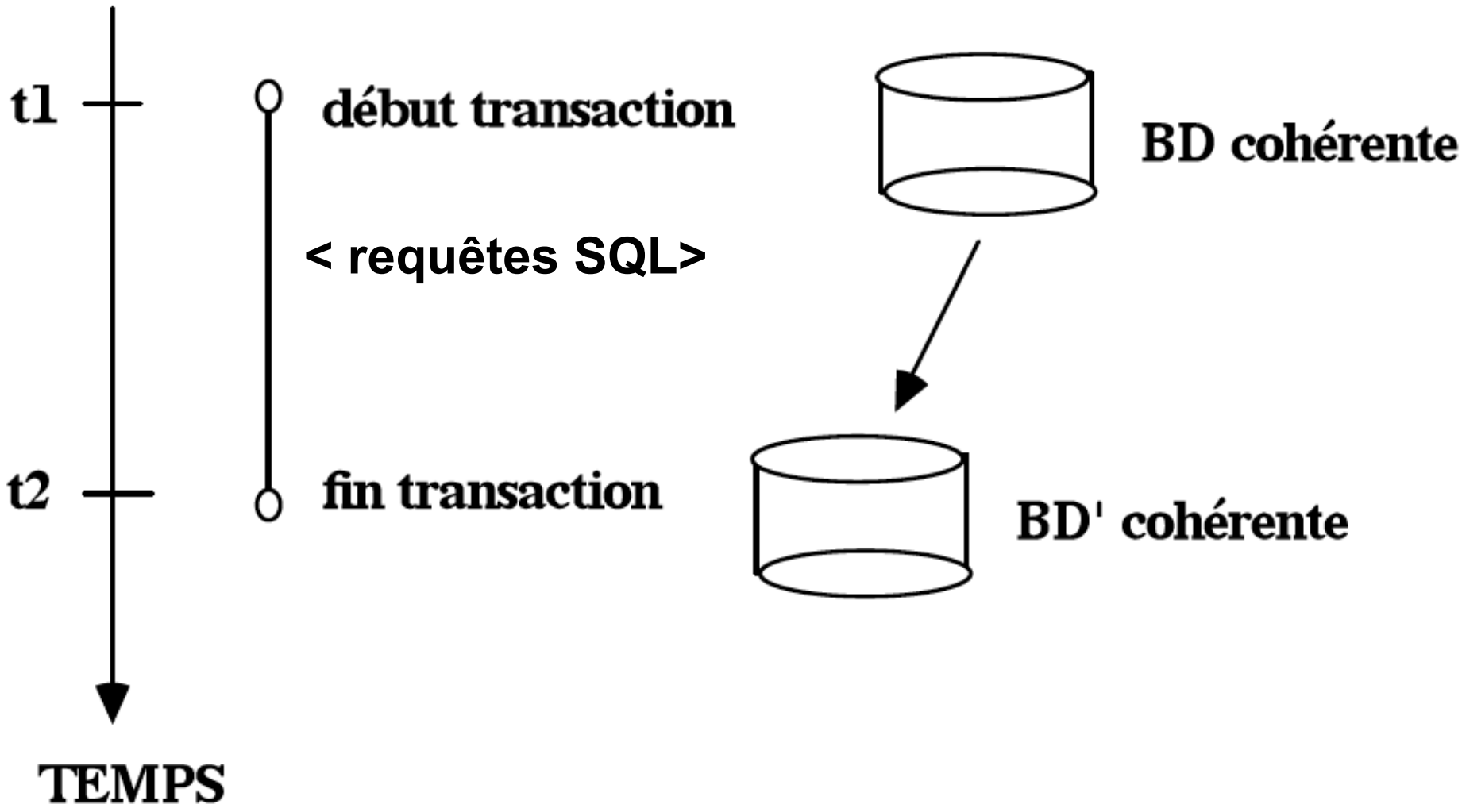
PROPRIÉTÉS QUE DOIVENT RESPECTER LES DONNÉES POUR ÊTRE CONFORMES AU MONDE RÉEL QU'ELLES REPRÉSENTENT

- **BASE DE DONNÉES COHÉRENTE**

BASE DE DONNÉES INACTIVE DONT TOUTES LES DONNÉES VÉRIFIENT LES CONTRAINTES D'INTÉGRITÉ

- **TRANSACTION**

UNITÉ COHÉRENTE DE TRAITEMENT


t1
début transaction
BD cohérente
< requêtes SQL>
t2
fin transaction
BD' cohérente
TEMPS

Chapitre 2 : DÉROULEMENT D'UNE TRANSACTION

- **DÉBUT**
  - **INSTRUCTION**
    - **SET TRANSACTION READ ONLY**
    - **SET TRANSACTION READ WRITE**
  - **PREMIÈRE REQUÊTE SQL**

  (ou après la fin de la transaction précédente)

- **TRAITEMENT (CONTENU DE LA TRANSACTION)**
  - **INTERROGATIONS (LECTURE: SELECT)**
  - **MISES-À-JOUR (ÉCRITURES: UPDATE/INSERT/ DELETE)**

- **FIN**
  - **FIN NORMALE ---> COMMIT (validation des màj)**
  - **FIN ANORMALE ---> ROLLBACK (annulation des màj)**

Chapitre 2 : ATOMICITÉ D'UNE TRANSACTION

• SOIT TOUTES LES MISES-À-JOUR SONT ENREGISTRÉES DANS LA BASE DE DONNÉES

COMMIT (VALIDATION)

• SOIT AUCUNE MISE-À-JOUR N'EST ENREGISTRÉE DANS LA BASE DE DONNÉES

ROLLBACK (ABANDON, ANNULATION)

Chapitre 2 : EXEMPLE DE TRANSACTION

Commande (ncom, ncli, npro, qtecom, datecom)

Client (ncli, raisonSociale, NbCom)

```
Set Transaction Read Write ;
INSERT INTO Commande values (10, 100, 45, 15, '17-Octobre-2000');
UPDATE Client SET NbCom = NbCom + 1 WHERE ncli = 100;
COMMIT;
```

Que se passe-t-il si la date de la commande est fausse (mal saisie)?

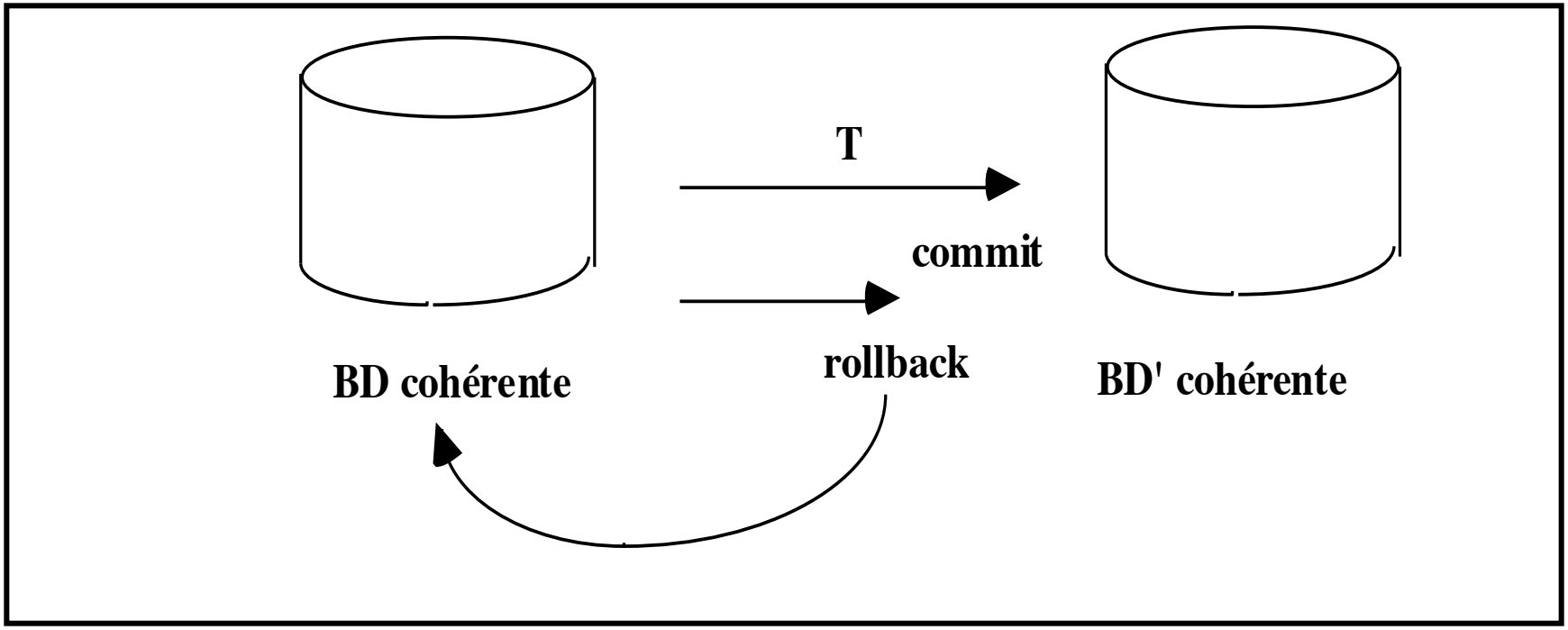
Ou si le client n° 100 n'existe pas?

==> Il ne faut pas faire les modifications dans la base.

Il faut abandonner la transaction

Chapitre 2 : PROPRIETES D'UNE TRANSACTION

UNITÉ DE COHÉRENCE AU NIVEAU DES TRAITEMENTS



PROPRIÉTÉS "ACID"

- ATOMICITÉ
- COHÉRENCE
- ISOLATION
- DURABILITÉ

Chapitre 2 : Intégrité sémantique

• Définition plus fine et plus précise de la sémantique des données

→ Prise en compte des règles de gestion (Business Rules)

• Contraintes d'intégrité : propriétés intrinsèques des données dans une application

• Les contraintes d'intégrité sont connues par le SGBD

Rôle du SGBD :

- Garder les bases de données cohérentes
- Vérifier les données lors des chargements
- Vérifier les données lors des mises à jour
- Répercuter certaines mises à jour entre relations
- Gérer les références inter-relations

C'est essentiel : Une BD non cohérente est peu utile !

TYPES DE CONTRAINTES(1)

- **DEFINITION DE DOMAINE**

  TypeCouleur = {Rouge, Vert, Bleu, Blanc, Jaune}

- **TYPE DE DONNÉES**

  LIBELLE_PRODUIT : TEXTE

- **PLAGE DE VALEUR**

  1 ≤ N° PRODUIT ≤ 1000

- **ÉNUMÉRATION DE VALEURS**

  VILLE ∈ (PARIS, LYON, BORDEAUX, ÉPINAL)

- **CONTRAINTE DE NON NULLITE :** Spécifie que la valeur d'un attribut ne peut être nulle

- **CONTRAINTE ENTRE ATTRIBUTS D'UN MÊME TUPLE**

  PRIX_VENTE > PRIX _ACHAT

  DATE_LIV > DATE_COM

TYPES DE CONTRAINTES(2)

• **CONTRAINTE TEMPORELLE :** permet de comparer l'ancienne valeur d'un attribut à la nouvelle après mise à jour

Ex : le salaire d'un employé ne peut pas décroître

• **UNICITÉ :** si la valeur d'un attribut est non nulle dans la relation, alors elle est unique

CHAQUE N° PRODUIT EST UNIQUE

• **CLÉ**

On peut définir plusieurs clés dans une relation

DANS COMMANDE:

N° COMMANDE

(N° CLIENT, N° PRODUIT, DATE_COM)

• **CONTRAINTE D'AGRÉGATS**

- MOYENNE DES SALAIRES > 6000
- UNE USINE FABRIQUE MOINS DE 10 PRODUITS DIFFÉRENTS

TYPES DE CONTRAINTES(3)

- **CONTRAINTES RÉFÉRENTIELLES**

  PRODUIT.N° USINE ⊆ USINE.N° USINE

  CLÉ ÉTRANGÈRE CLÉ

- **DÉPENDANCES D'INCLUSION :** généralise les contraintes référentielles
  - **SOUS-TYPE**

    ENSEIGNANT. N° EMPL ⊆ PERSONNEL_UNIV.N° EMPL

    ADMINISTRATIF. N° EMPL ⊆ PERSONNEL_UNIV.N° EMPL

  - **INCLUSION ENTRE ATTRIBUTS NON CLÉ**

    VILLE_USER ⊆ VILLE_CLIENT

Expression des contraintes en SQL (1)

• **CREATION DE DOMAINE**

CREATE DOMAIN <nom_domaine> [AS] <type de données>
[DEFAULT <valeur>]
<contrainte de domaine> < , contrainte de domaine>* ;

<contrainte de domaine> =
[CONSTRAINT nom-contrainte] CHECK (condition de sélection SQL) [vérification]

([vérification] est vu plus loin)

Exemples : **CREATE DOMAIN NOM AS varchar(20)**
CONSTRAINT domain-nom
CHECK (VALUE IS NOT NULL);
CREATE DOMAIN COULEUR AS varchar(20)
CONSTRAINT domain-couleur
CHECK (VALUE IN ('Rouge', 'Vert', 'Bleu', 'Jaune'));

Expression des contraintes en SQL (2)

- **DEFINITION DE CONTRAINTES LORS DE LA CRÉATION DE LA RELATION**

```
CREATE TABLE <nom_relation>
( <élément de relation> < , élément de relation>*
  < , contrainte de relation>*
);
<élément de relation> ::= <nom_attribut> <type de données>
                          [DEFAULT valeur]
                          [ <contrainte d 'attribut>* ]
```

CONTRAINTE D'ATTRIBUT : ne porte que sur un seul attribut

CONTRAINTE DE RELATION : peut porter sur plusieurs attributs

Expression des contraintes en SQL (3)

- **TYPES DE CONTRAINTES**
  - **CLÉ :**
    - **NOT NULL + UNIQUE**
    - **PRIMARY KEY (une seule par relation)**
  - **ATTRIBUT OBLIGATOIRE : NOT NULL**
  - **CONTRAINTE RÉFERENTIELLE**
    - **REFERENCES nom-relation-référencée [(nom-d'attribut)]**
    - **FOREIGN KEY (nom_attribut [, nom_attribut]*) REFERENCES nom-relation [(nom_attribut [, nom_attribut]*)]**
  - **AUTRES CONTRAINTES**
    - **CHECK (condition de sélection SQL)**

- **FORMAT GÉNÉRAL**
  - **[CONSTRAINT nom-contrainte] expression-contrainte [vérification]**

Expression des contraintes en SQL (4)

Exemple de contrainte de clé avec plusieurs attributs

- **CREATE TABLE personne (**
  nom VARCHAR (20), prenom VARCHAR (20),
  PRIMARY KEY (nom, prenom),
  adresse VARCHAR (80) not null);

- **Exemple de contrainte référentielle avec plusieurs attributs**
  CREATE TABLE commande (
  numcom INTEGER PRIMARY KEY,
  nom VARCHAR (20), prenom VARCHAR (20),
  FOREIGN KEY (nom, prenom) REFERENCES personne,
  n° pro INTEGER not null REFERENCES produit,
  qte_com INTEGER);

Expression des contraintes en SQL (5)

Exemples de contrainte avec CHECK

- **Plage de Valeur**

```
CREATE TABLE produit (
        n°  produit INTEGER
                CONSTRAINT dom_n°  produit
                        CHECK (n°  produit BETWEEN 1 AND 1000)
                CONSTRAINT clé1-pro PRIMARY KEY,
        ... );
```

- **Enumération de valeurs**

```
CREATE TABLE client (
        …
        ville_cli VARCHAR (8)
                CONSTRAINT dom_ville_cli
                        CHECK (ville_cli IN ('paris',  'macon', 'lyon')),
        …);
```

Expression des contraintes en SQL (6)

• **AUTRES CONTRAINTES (entre attributs, d'agrégat,…)**

```
CREATE TABLE produit (
        n° produit INTEGER … ,
        poids_prod INTEGER,
        prix_fabric INTEGER,
        prix_ht INTEGER,
        …
        CONSTRAINT prixvente1 CHECK (prix_ht > prix_fabric ),
        …
        CONSTRAINT prod_valide CHECK ( n° produit NOT IN
                        (SELECT n° produit FROM produit_abandonné)),
        …);
```

Expression des contraintes en SQL (7)

- **DEFINITION DE CONTRAINTES PAR LA CLAUSE ASSERTION**

La contrainte n'est pas associée à une relation.

```
CREATE ASSERTION  <nom de contrainte>
        CHECK (condition de sélection SQL)
        [vérification] ;
```

Exemple : le poids total d'une commande ne dépasse pas 3 kg

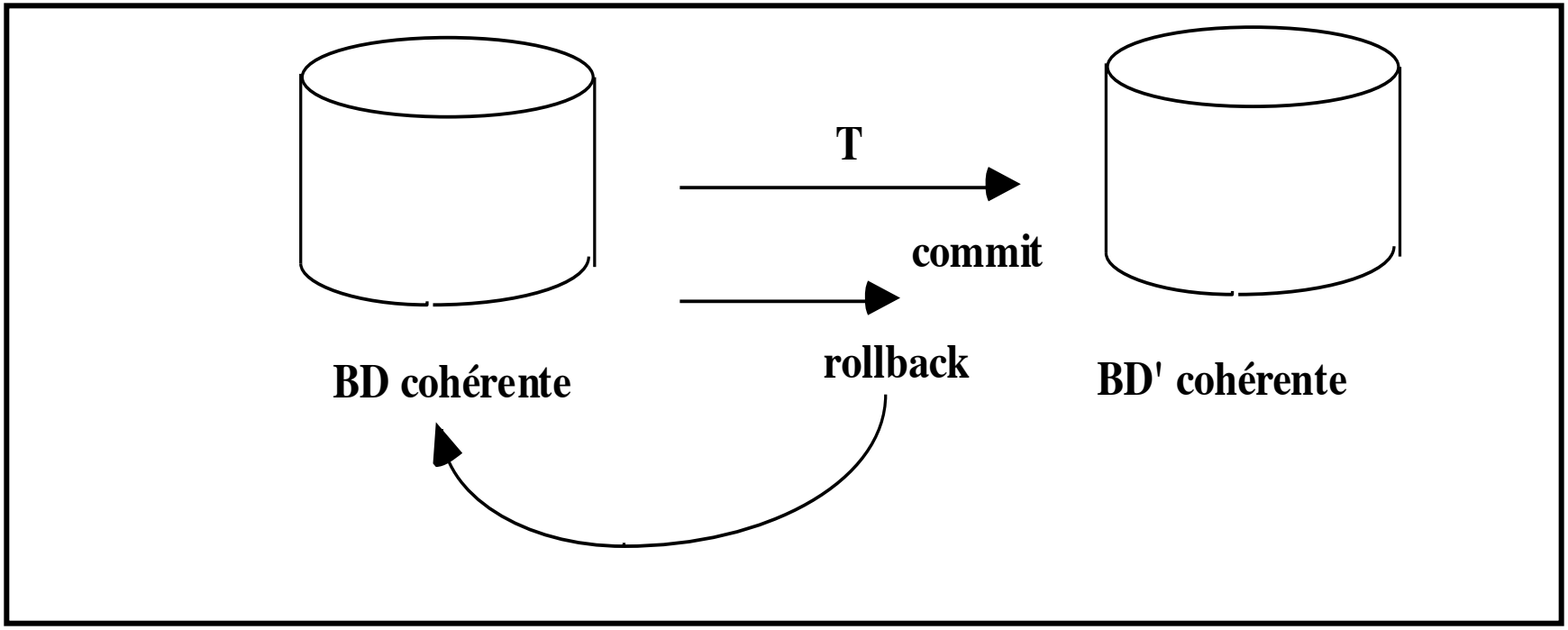
```
CREATE ASSERTION PdsCom
CHECK (NOT EXISTS
        (SELECT * FROM commande NJ produit
        WHERE qté_com * poids_prod > 3));
```

Suppression d'une assertion :

```
DROP ASSERTION <nom de contrainte>
```

RAPPEL : TRANSACTION

UNITÉ DE COHÉRENCE AU NIVEAU DES TRAITEMENTS



PROPRIÉTÉS "ACID"

- ATOMICITÉ
- COHÉRENCE
- ISOLATION
- DURABILITÉ

VERIFICATION DES CONTRAINTES (1)

COÛT DE VÉRIFICATION DES CONTRAINTES

- **CONTRAINTES INDIVIDUELLES**
  - **VÉRIFIABLES À CHAQUE M-À-J**
  - **COÛT PEU ÉLEVÉ**

- **CONTRAINTES ENSEMBLISTES**
  - **COÛT ÉLEVÉ**
  - **CERTAINES VÉRIFIABLES SEULEMENT EN FIN DE TRANSACTION**

- **CONTRAINTES AVEC AGRÉGAT**
  - **TECHNIQUES SPÉCIFIQUES NÉCESSAIRES**

VERIFICATION DES CONTRAINTES (2)

QUAND SONT VÉRIFIÉES LES CONTRAINTES D'INTÉGRITÉ ?

-NOT DEFERRABLE (par défaut en SQL 92)

CONTRAINTE À VÉRIFICATION IMMÉDIATE (À LA FIN DE CHAQUE INSTRUCTION DE MISE-À-JOUR)

-DEFERRABLE [INITIALLY DEFERRED | INITIALLY IMMEDIATE]

CONTRAINTE À VÉRIFICATION :

- SOIT DIFFÉRÉE AU COMMIT (**INITIALLY DEFERRED**),
- SOIT À L'EXÉCUTION DE CHAQUE REQUETE SQL DE MISE-À-JOUR (**INITIALLY IMMEDIATE**).

Pour les contraintes DEFERRABLE, le mode de vérification peut être changé :

SET CONSTRAINT { **ALL** / **liste noms C.I** } { **IMMEDIATE** / **DEFERRED** }

VERIFICATION DES CONTRAINTES (3)

```
CREATE TABLE projet (
        num_projet INTEGER PRIMARY KEY NOT DEFERRABLE,
        date_deb DATE
                CONSTRAINT date_début_projet
                CHECK (date_deb > ' 01-jan-1996') NOT DEFERRABLE,
        chef_proj INTEGER
                CONSTRAINT chef_projet
                CHECK (chef_proj IN (SELECT num_empl FROM
                        employé WHERE fonction = 'analyste'))
                                DEFERRABLE INITIALLY DEFERRED
);
```

Exemple du TP5 (S1) en SQL 92

Insérez le vin de numéro 10, de cru ‘MEURSAULT', de millésime 1985, de région 'BOURGOGNE', produit par le viticulteur de numéro 14, de nom 'DUPONT', de prénom 'JEAN' et de la ville de 'POMMARD'

```
SET TRANSACTION READ WRITE;

INSERT into viticulteur VALUES (14, 'DUPONT', 'JEAN', 'POMMARD');

INSERT into vin VALUES (10, 'MEURSAULT',1985, 'BOURGOGNE',14);

Commit;
```

Remarque : même si on définit une transaction avec "SET TRANSACTION READ WRITE;" les 2 requêtes doivent rester dans cet ordre car les contraintes référentielles sont par défaut en NOT DEFERRABLE.

Exemple du TP3 (S2) en SQL 92 (1)

Insérer la cave coopérative de numéro 12 qui se trouve à STRASBOURG et dont le responsable est le viticulteur numéro 40 de nom WEISS et de prénom JACQUES et qui habite à COLMAR.

Les contraintes référentielles étant par défaut en NOT DEFERRABLE, elles sont vérifiées à chaque requête. Donc il faut exécuter dans l'ordre les requêtes suivantes:

```
SET TRANSACTION READ WRITE;
INSERT into viticulteur
VALUES (40, ' WEISS', 'JACQUES', 'COLMAR', NULL);

INSERT into Cave_Cooperative
VALUES (12, 'STRASBOURG',40);

UPDATE viticulteur set numcav = 12 where numvitic = 40;

Commit;
```

Exemple du TP3 (S2) en SQL 92 (2)

2 solutions :

- définir les contraintes en DEFERRABLE INITIALLY DEFERRED : elles seront vérifiées en fin de transaction :

```
SET TRANSACTION READ WRITE;
INSERT into viticulteur
VALUES (40, ' WEISS', 'JACQUES', 'COLMAR', 12);
INSERT into Cave_Cooperative
VALUES (12, 'STRASBOURG',40);
Commit;
```

- définir les contraintes en DEFERRABLE INITIALLY IMMEDIATE et forcer leur vérification en fin de transaction par la commande SET CONSTRAINT

```
SET TRANSACTION READ WRITE;
SET CONSTRAINT DEFERRED;
INSERT into viticulteur
VALUES (40, ' WEISS', 'JACQUES', 'COLMAR', 12);
INSERT into Cave_Cooperative
VALUES (12, 'STRASBOURG',40);
Commit;
```

Remarque : dans les 2 solutions, l'ordre des 2 insert n'a pas d'importance

Chapitre 2 : LES TRIGGERS RÉFÉRENTIELS - Introduction

- **ACTIONS À DÉCLENCHER AUTOMATIQUEMENT POUR SATISFAIRE LES CONTRAINTES RÉFÉRENTIELLES**

- **ASSOCIÉS À LA DÉFINITION DES CONTRAINTES RÉFÉRENTIELLES**

RAPPELS

CREATE TABLE département (num_dep INTEGER PRIMARY KEY);

CREATE TABLE projet (num_projet INTEGER PRIMARY KEY,

num_dep INTEGER REFERENCES département);

Si on supprime un département, et s'il existe des projets dans ce département, alors la suppression sera refusée par le SGBD.

Ex : département = {1,3,5} projet = {(23,1), (65,3), (13,1)}

Begin Transaction

Delete From département Where num_dep = 5;

Commit;

Pas de projet associé au département 5, donc la transaction est exécutée correctement

Begin Transaction

Delete From département Where num_dep = 1;

Commit;

2 projets sont associés au département 1, donc la transaction est refusée

Les triggers référentiels – Définition SQL2

REFERENCES nom-relation-référencée [(nom-d'attribut)]

ou

FOREIGN KEY (nom_attribut [, nom_attribut]*) REFERENCES
nom-relation [(nom_attribut [, nom_attribut]*)]

[ON DELETE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]

[ON UPDATE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]

On peut préciser l'action que le SGBD va exécuter si la contrainte référentielle est violée lors d'une maj ou d'une suppression dans la relation référencée.

(par défaut c'est NO ACTION et la requête SQL échouera)

CASCADE : propagation de la suppression ou de la mise à jour dans la relation qui référence (et récursivement)

SET DEFAULT : la clé étrangère prend la valeur par défaut (si elle existe)

SET NULL : la clé étrangère prend la valeur NULL (si c'est possible)

Les triggers référentiels – SQL2 – Exemple (1)

```
CREATE TABLE département ( num_dep INTEGER PRIMARY KEY);

CREATE TABLE projet (    num_projet INTEGER PRIMARY KEY,
                         num_dep INTEGER REFERENCES département
                         ON UPDATE CASCADE ON DELETE CASCADE );
```

ON UPDATE CASCADE : si on modifie le num_dep dans département alors la modif est répercutée sur les tuples de projet concernés

Ex : département = {1,3,5} projet = {(23,1), (65,3), (13,1)}

Update département set num_dep = 10 where num_dep = 3;

département = {1,10,5} projet = {(23,1), (65,10), (13,1)}

ON DELETE CASCADE : si on supprime un département, les tuples de projet concernés sont aussi supprimés.

Delete From département where num_dep = 1;

département = {10,5} projet = {(65,10)}

Les triggers référentiels – SQL2 – Exemple (2)

CREATE TABLE employé (num_employé VARCHAR(20) PRIMARY KEY,

num_projet INTEGER REFERENCES projet

ON UPDATE SET NULL ON DELETE SET NULL);

Si on modifie l'attribut *num_projet* de *projet* ou si on supprime un *projet*, l'attribut *num_projet* de *employé* est mis à NULL pour les tuples de *employé* concernés.

Ex : projet = {(23,1), (65,3), (13,1)} employé = {(e98,13), (f54,65), (j76,23)}

Update projet set num_projet = 17 where num_projet = 13;

projet = {(23,1), (65,3), (17,1)} employé = {(e98,NULL), (f54,65) , (j76,23)}

Propagation des mises à jour : si une action (delete ou update) est réalisée sur *département* alors elle est répercutée sur *projet* et sur *employé*

département = {1,10,5}

Delete From département where num_dep = 1;

département = {10,5} projet = {(65,3)}

employé = {(e98,NULL), (f54,65), (j76,NULL)}

Intégrité – Conclusion (1)

- "BIEN CONCEVOIR" LE SCHÉMA
  - CHOISIR LES C.I ET/OU LES TRIGGERS À INCLURE
  - COHÉRENCE DES CONTRAINTES

- "BIEN DÉFINIR " LES TRANSACTIONS
  - ACTIONS CORRECTRICES
  - VÉRIFICATIONS
  - ABANDON

- POUVOIR ACTIVER / DESACTIVER LES CONTRAINTES
  - CHANGEMENTS MASSIFS
  - IMPORTATION DE DONNÉES
  - CORRECTIONS PAR LE DBA

Intégrité – Conclusion (2)

CONFLITS ENTRE CONTRAINTES

EXEMPLE :

E.SALAIRE > 4000

E.DIRECTEUR = 3500

===> PAS DE SOLUTION GÉNÈRALE IMPLANTÉE

PRATIQUE :

SI ON A PU INTRODUIRE AU MOINS UN TUPLE DANS CHAQUE RELATION DE LA BASE, ALORS L'ENSEMBLE DES CONTRAINTES EST COHÉRENT

Chapitre 3 : Gestion des utilisateurs

La notion d'utilisateur est fondamentale pour accéder aux données d'une base de données.

A chaque utilisateur est associé un schéma qui contient les objets gérés par l'utilisateur :

- **Relations,**
- **Vues,**
- **Contraintes,**
- **Procédures,**
- **…**

Le nom de ce schéma est le nom de l'utilisateur.
Les objets du schéma sont préfixés par ce nom de schéma.

Exemple : laleau.commande

Requête SQL : CREATE USER (ORACLE)

CREATE USER <nom_user>

IDENTIFIED BY { <mot_de_passe> | EXTERNALLY}

[DEFAULT TABLESPACE <nom_tablespaceD>]

[TEMPORARY TABLESPACE <nom_tablespaceT>]

[QUOTA {<nombre> [K|M] | UNLIMITED}]

[PROFILE <nom_profil>]

[PASSWORD EXPIRE]

Suppression d'un utilisateur :

DROP USER <nom_user> [CASCADE]

CASCADE : supprime tous les objets du schéma de l'utilisateur avant de supprimer l'utilisateur.

Requête SQL : CREATE USER (ORACLE)

nom_user : Un nom utilisateur unique. Les noms utilisateurs ne peuvent pas dépasser 30 caractères, ne doivent pas contenir de caractères spéciaux et doivent commencer par une lettre

Méthode d'authentification (IDENTIFIED BY) : soit par un mot de passe propre à Oracle, soit par le mot de passe récupéré du compte utilisateur du système d'exploitation

DEFAULT TABLESPACE : Espace disque de travail pour créer des objets propres à l'utilisateur (relations, vues, …)

TEMPORARY TABLESPACE : Espace disque de travail temporaire pour faire des tris par exemple

QUOTA : taille de l'espace de travail (K = kilooctet, M =Megaoctet)

PROFILE : définit un profil utilisateur : caractéristiques d'utilisation des ressources du serveur affectées à l'utilisateur et contraintes sur les mots de passe. Un profil par défaut est affecté à chaque utilisateur lors de sa création.

PASSWORD EXPIRE : il faut changer le mot de passe à la 1ère connexion.

Requête SQL : CREATE USER (ORACLE) - EXEMPLE

```
CREATE USER fin_user1
IDENTIFIED BY toto
DEFAULT TABLESPACE data1
TEMPORARY TABLESPACE temp
QUOTA 100M ON data1
PASSWORD EXPIRE;
```

Chapitre 4 : Gestion de la Confidentialité

Confidentialité : permet de protéger les données gérées contre tout accès (malveillant ou accidentel) non autorisé.

2 commandes SQL pour octroyer ou retirer des droits d'accès sur une relation ou une vue : GRANT et REVOKE

Les droits d'accès donnent la possibilité d'exécuter des requêtes SQL.

Le propriétaire (créateur) d'une relation à tous les droits sur cette relation et on ne peut lui les supprimer.

Contrôle d'accès dans les SGBD

3 concepts : *sujets, objets, privilèges*

Sujet : utilisateur avec un identificateur SQL d'autorisation d'accès.

- **Accompagné en général d'un mot de passe.**
- **Géré dans une relation spéciale de la BD.**
- **Peut être une personne ou une application.**

Les objets à protéger sont de deux types:

- **Les tuples des relations et des vues.**
- **Les éléments de la métabase (relation, index, trigger, ...)**

Un privilège est le droit d'exécuter un type d'instruction SQL spécifique.

- **le droit de sélectionner des lignes dans une table,**
- **le droit de créer une table.**

SQL - privilèges

- **Privilèges objets : droits limités à un objet de la base,**

  select, insert, update … (requêtes du Langage de Manipulation de Données)

- **Privilèges systèmes: droits d'exécuter des actions portant sur tout le système:**

  create table, create view, alter table, … (requêtes du Langage de Définition de Données)

Commande Grant (privilèges systèmes) (Oracle)

GRANT {privilège {,privilège…}| ALL PRIVILEGES}
TO {user-name {, user-name…} | PUBLIC}
[WITH ADMIN OPTION]

Privilege: CREATE, ALTER, DROP, …

WITH ADMIN OPTION : permet de distribuer ce privilège à d'autres utilisateurs

- **grant create table to Pierre ;**

=> permet à Pierre de créer une table dans son schéma

- **grant create any table to Pierre with admin option;**

=> permet à Pierre de créer une table dans n'importe quel schéma. La clause 'admin option' permet à Pierre de distribuer ce privilège à d'autres utilisateurs

Commande Grant (privilège objets)

GRANT {ALL PRIVILEGES | privilège {,privilège…}}

on [TABLE] nom-relation | nom-vue

TO {PUBLIC | user-name {, user-name…}}

[WITH GRANT OPTION]

privilège : **SELECT, DELETE, INSERT,**

s'appliquent sur les relations (ou vues) *entières*.

UPDATE [(nom-attribut {, nom-attribut})],

REFERENCES [(nom-attribut {, nom-attribut})]

donne le droit de référencer un attribut dans une clé étrangére

EXECUTE donne le droit d'exécuter des pgs PL/SQL

Tout créateur d'une relation a tous les droits sur celle-ci

Grant - exemple

Bob: CREATE TABLE Employé
(Num INTEGER Primary Key,
Nom varchar(30),
...);

Bob: GRANT select, insert, update(Nom) ON Employé TO Ann;
Bob: GRANT select ON Employé TO Jim;

Ann: SELECT * FROM Employé ;
Requête acceptée
Ann: DELETE FROM Employé WHERE Nom = 'Dupond';
Requête rejetée, Ann n'a pas le droit de suppression (delete) sur Employé

Ann: GRANT select, insert ON Employé TO Jim;
Requête rejetée, Ann n'est pas propriétaire de la relation Employé

Délégation des privilèges en SQL92

- **La délégation de privilèges se fait par *grant option* : si un privilège est accordé avec *grant option*, celui qui le recevra pourra non seulement utiliser son privilège mais aussi l'accorder à d'autres utilisateurs.**

- **Un utilisateur peut seulement accorder un privilège sur une relation s'il est propriétaire de la relation ou s'il a reçu ce privilège avec l'option *grant option***

Grant avec délégation - exemple

Bob: GRANT select, insert ON Employé TO Ann
WITH GRANT OPTION;
Bob: GRANT select ON Employé TO Jim
WITH GRANT OPTION;
Ann: GRANT select, insert ON Employé TO Jim;

- Sur **Employé, Jim** a le privilege **select** (reçu à la fois de **Bob** et **Ann**) et le privilege **insert** (reçu de **Ann**)

- **Jim** peut accorder à d'autres le privilege **select** (parce qu'il l'a reçu avec grant option), mais pas le privilege **insert**

Exécution d'un Grant

- **Le SGBD garde, pour chaque utilisateur, les privilèges qu'il possède et ceux qu'il peut transmettre.**

- **A chaque fois qu'un utilisateur exécute une commande `Grant`, le système fait l'intersection entre les privilèges qu'il peut accorder et l'ensemble des privilèges spécifiés dans la commande. Si l'intersection est vide, la commande n'est pas exécutée.**

Exécution d'un Grant – exemple (1)

Bob: GRANT select, insert ON Employé TO Jim WITH GRANT OPTION;

Bob: GRANT select ON Employé TO Ann WITH GRANT OPTION;

Bob: GRANT insert ON Employé TO Ann;

Jim: GRANT update ON Employé TO Tim
WITH GRANT OPTION;

Ann: GRANT select, insert ON Employé TO Tim;

Exécution d'un Grant – exemple (2)

Les 3 premiers **GRANT** sont exécutés (**Bob** est le propriétaire de la relation)

Le 4ème **GRANT** n'est pas exécuté car **Jim** n'a pas de privilège **update** sur la relation.

Le 5ème **GRANT** est partiellement exécuté: **Ann** a les privilèges **select** and **insert** mais pas le **grant option** pour le **insert**.

---> **Tim** reçoit seulement le privilège **select**

Commande Revoke (suppression des droits)

REVOKE {ALL PRIVILEGES | priv {, priv…} }

ON nom-relation | nom-vue

FROM {PUBLIC | user {, user…} }

;

- **Un utilisateur peut seulement retirer les privilèges qu'il a reçus. Ce n'est pas possible de retirer uniquement le grant option.**

- **A l'exécution d'un revoke, l'utilisateur à qui les privilèges ont été retirés perd ces privilèges sauf s'il les a aussi reçus d'une autre source indépendante de celle qui a exécuté le revoke.**

Commande Revoke - exemple

Bob: GRANT select ON Employé TO Jim WITH GRANT OPTION;

Bob: GRANT select ON Employé TO Ann WITH GRANT OPTION;

Jim: GRANT select ON Employé TO Tim;

Ann: GRANT select ON Employé TO Tim;

Jim: REVOKE select ON Employé FROM Tim;

Tim continue de posséder le privilège select sur Employé après la commande car il l'a reçu aussi de Ann indépendamment.

Propagation du Revoke

Retrait récursif: quand un utilisateur U retire un privilège sur une relation à un autre utilisateur V, tous les privilèges que V avait accordés sont aussi retirés.

Le processus est appliqué itérativement à tous les sujets qui ont reçu le privilège de V.

Vues et confidentialité

Les vues sont utilisées pour gérer le contrôle d'accès basé sur le contenu:

- Définir une vue V contenant les tuples qui peuvent être lus par un sujet S
- Accorder à S le privilège "select" sur la vue V, et pas sur les relations de base.

Vues et confidentialité : exemple (1)

- **Attribution du privilège Select à certains attributs d'une relation :**

  Employee (Num, Name, Address, Salary)

```
CREATE VIEW Vemp (Name, Address) AS
SELECT Name, Address FROM Employee ;

GRANT Select ON Vemp TO Ann;
```

Vues et confidentialité : exemple (2)

- **Attribution du privilège Select à certains tuples d'une relation :**

  Supposons qu'on veuille autoriser l'utilisateur Ann à accéder seulement aux employés dont le salaire est inférieur à 20000 euros :

  – **CREATE VIEW Vemp AS**
  SELECT * FROM Employee
  WHERE Salary < 20000;

  - GRANT Select ON Vemp TO Ann;

Vues et confidentialité : exemple (3)

- **Attribution du privilège Select pour des statistiques:**

Employee (Num, Name, Address, NumDepart, Salary)

Department (NumDepart, Head, Location)

```
CREATE VIEW AvgSal (NumDepart, AvgSalary) AS SELECT
NumDepart, Avg(all Salary)
    FROM Department natural join Employee
    GROUP BY NumDepart;

GRANT Select ON AvgSal TO Ann;
```

Privilèges du créateur d'une vue (1)

- Soit U l'utilisateur qui crée une vue, les privilèges que U a sur la vue dépendent :
  - de la sémantique de la vue: certaines opérations ne peuvent pas être exécutées.

    Ex : sur la vue AvgSal, l'attribut AvgSalary ne peut pas être mis à jour.
  - des privilèges que U a sur les relations de la base
- Exemple:
  - Bob crée la relation Employee (Num, Name, Salary)
  - Bob: GRANT Select, Insert, Update ON Employee TO Tim;
  - Tim: CREATE VIEW V1 AS SELECT Num, Salary FROM Employee
  - Tim: CREATE VIEW V2 (Num, Annual_Sal) AS

    SELECT Num, Salary * 12 FROM Employee;

  Sur V1, Tim a les mêmes privilèges que sur Employee

  Sur V2, Tim a les privilèges Select et Update(Num)

Privilèges du créateur d'une vue (2)

- **Soit U l'utilisateur qui crée une vue, les privilèges que U peut accorder sur la vue sont ceux qu'il reçoit avec *grant option* sur les relations de base.**

- **Exemple:**
  - **Sur V1 et V2, Tim ne peut pas accorder de privilèges**
  - **Bob: GRANT Select ON Employee TO Tim with Grant Option;**
  - **Bob: GRANT Insert, Update ON Employee TO Tim;**
  - **Tim: CREATE VIEW V4 AS SELECT * FROM Employee WHERE salary < 20000;**
  - **Tim: GRANT Select ON V4 TO Ann;**