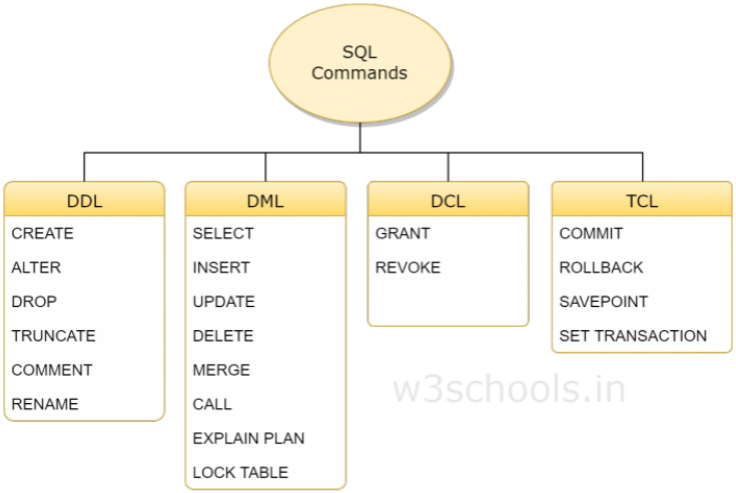
Chapitre 4 - SQL, création et modification de tables

1. SQLDDL-DML
   - Créer des relations (attributs et ses domaines)
   - Contraintes d'intégrité
   - Modification, suppression de relations

Instruction SQL - Ces slides traitent...



- DDL – Data Definition Language (CREATE, ALTER, DROP)
- DML – Data Manipulation Language (INSERT, DELETE, UPDATE)
- DCL – Data Control Language
- TCL - Transaction Control Language

Définition de données en SQL

Langage de Définition de Données de SQL :

- Créer et modifier un schéma de base de données : relations, vues, utilisateurs, ...

Langage de Manipulation de Données de SQL :

- Insérer et modifier des données

Les instructions LDD et LDM sont soumises comme des requêtes (via l'interpréteur SQL ou une application).
Une requête du LDD ou LDM est exécutée par le SGBD uniquement lorsque le nouvel état de la base de données qu'elle produit est cohérent.

Créer des relations (attributs)

Pour créer des relations (vides) :

CREATE TABLE T (
    Liste_de_définitions_d'attributs
    [,Liste_de_définitions_de_contraintes]
) ;

Définition des attributs (contraintes de domaine)

Liste de définitions d'attributs (ORACLE) :

- *A1 NUMBER (3)*,
- *A2 VARCHAR (10)*,
- *A3 NUMBER (3) DEFAULT 1*,
- *A4 DATE*,
- ...

Liste de définitions d'attributs (PostgresQL) :

- *A1 INTEGER*,
- *A2 VARCHAR (10)*,
- *A3 INTEGER DEFAULT 1*,
- *A4 DATE*,
- ...

*L'ensemble des types proposés ainsi que les notations associées sont dépendants du système.

Définition des attributs en SQLite (contraintes de domaine)

*L'ensemble des types proposés ainsi que les notations associées sont dépendants du système.

Liste de définitions d'attributs (SQLite) avec les types de base:

- *A1 INTEGER*,
- *A2 TEXT*,
- *A3 INTEGER DEFAULT 1*,
- *A4 TEXT*,
- ...

Liste de définitions d'attributs (SQLite) sans les types de base (map dynamique)

- *A1 INT*,
- *A2 VARCHAR(10)*,
- *A3 INT DEFAULT 1*,
- *A4 DATE*, – – Date est sauvegardé comme texte ou entier en fonction du INSERT
- ...

Infos sur les types SQLite : `https://sqlite.org/datatype3.html`

Infos sur le nouveau opérateur STRICT : `https://sqlite.org/stricttables.html`

Infos sur les dates : `https://www.sqlite.org/lang_datefunc.html`

Créer des relations (autres contraintes)

Pour créer des relations (vides) :

CREATE TABLE T (
    Liste_de_définitions_d'attributs
    [,**Liste_de_définitions_de_contraintes**]
) ;

Intégrité des données

L'intégrité des données repose sur les *contraintes d'intégrité* qui :

- préviennent l'entrée de données invalides dans les tables de la base de données.
- précisent le sens des informations stockées dans la base de données.

Par exemple :

- Le numéro d'étudiant est un entier sur au plus 9 chiffres.
- Un adhérent de la bibliothèque ne peut pas avoir plus de 4 emprunts en cours.

Types de contraintes

Une relation, un état de la base de données

- Restriction de valeurs :
  Le salaire doit être plus grand que 100
- Obligatoire :
  Le nom ne peut pas être absent
- Unicité :
  Tous les noms sont différents deux à deux
- Identification :
  Chaque employé est identifié par son numéro.

Plusieurs relations, un état de la base de données

- Intégrité référentielle : une valeur (projection sur un attribut ou un ensemble d'attributs) dans une relation doit correspondre à au moins une valeur dans une relation reliée.
  Chaque chef est un employé qui travaille dans le rayon.

Spécifier les contraintes en SQL : avantages

Déléguer au SGBD la responsabilité des contraintes

- **Facilité (requête SQL)**
  Aucun programme supplémentaire n'est nécessaire. Le SGBD garde le contrôle.
- **Centralisation**
  Associer les contraintes aux relations les rend plus faciles à manipuler que si elles étaient dispersées dans les programmes d'application.
- **Performance Supérieure**
  La sémantique des contraintes d'intégrité est clairement définie, et des optimisations sont spécifiquement implémentées pour chaque cas.
- **Flexibilité**
  Les contraintes d'intégrité peuvent être temporairement désactivées pour des besoins spécifiques.
  Attention : quand une contrainte est désactivée, elle l'est pour toutes les applications !

Conclusion :

- La plupart des SGBD supportent seulement une partie de ces contraintes d'intégrité (une relation, un état de la base de données).
- Les contraintes d'intégrité qui ne sont pas supportées par le SGBD doivent être gérées au niveau de l'application (ou avec des déclencheurs (*triggers*) lorsque c'est possible). Exemples:
  - *Le salaire des employés ne peut pas diminuer.*
  - *Pour être nommé chef d'un département, un employé doit avoir été employé au moins 2 ans.*
  - *Chaque employé doit gagner un salaire plus petit que celui de son chef*
- L'interface graphique peut quelques fois vérifier la cohérence des données :
  - Restrictions de type : saisie d'une date dans un format prédéfini.
  - Énumération et contrainte référentielle : choix dans un menu ou une liste

Il faut utiliser au maximum les possibilités du SGBD.

Spécifications de contraintes en SQL

CONSTRAINT ch_nom_contrainte CHECK (P)

/ Plusieurs contraintes de restriction. P est un prédicat qui porte sur les attributs de la relation et/ou sur des constantes. */*

Spécifications de contraintes en SQL

CONSTRAINT ch_nom_contrainte CHECK (P)

/ Plusieurs contraintes de restriction. P est un prédicat qui porte sur les attributs de la relation et/ou sur des constantes. */*

CONSTRAINT pk_nom_contrainte PRIMARY KEY (A1, A2)

/ A1,A2 forment une clef.* Une seule clef primaire par relation. **/*

CONSTRAINT uq_nom_contrainte UNIQUE (A3,A1)

/ A3,A1 forment une autre clef */*

Spécifications de contraintes en SQL

CONSTRAINT ch_nom_contrainte CHECK (P)

/ Plusieurs contraintes de restriction. P est un prédicat qui porte sur les attributs de la relation et/ou sur des constantes. */*

CONSTRAINT pk_nom_contrainte PRIMARY KEY (A1, A2)

/ A1,A2 forment une clef.* Une seule clef primaire par relation. */*

CONSTRAINT uq_nom_contrainte UNIQUE (A3,A1)

/ A3,A1 forment une autre clef */*

CONSTRAINT fk_nom_contrainte FOREIGN KEY (A1,..,An) REFERENCES TargetRel(B1,..,Bn)

/ B1,...,Bn doivent être clef primaire dans la relation existante TargetRel. B1,...,Bn doivent être compatibles avec A1,...,An. */*

Spécifications de contraintes en SQL

CONSTRAINT ch_nom_contrainte CHECK (P)

/ Plusieurs contraintes de restriction. P est un prédicat qui porte sur les attributs de la relation et/ou sur des constantes. */*

CONSTRAINT pk_nom_contrainte PRIMARY KEY (A1, A2)

/ A1,A2 forment une clef.* Une seule clef primaire par relation. **/*

CONSTRAINT uq_nom_contrainte UNIQUE (A3,A1)

/ A3,A1 forment une autre clef */*

CONSTRAINT fk_nom_contrainte FOREIGN KEY (A1,..,An) REFERENCES TargetRel(B1,..,Bn)

/ B1,...,Bn doivent être clef primaire dans la relation existante TargetRel. B1,...,Bn doivent être compatibles avec A1,...,An. */*

A4 DATE CONSTRAINT nn_nom_contrainte NOT NULL

/ La contrainte de valuation obligatoire doit être placée avec la définition de l'attribut. */*

Un premier exemple

Bars (name, addr, openDate)
Drinker(name, addr, phone)
Frequents[drinker] $\subseteq$ Drinker[name]

Frequents(drinker, bar)
Frequents[bar] $\subseteq$ Bars[name]

Un premier exemple

Bars (<u>name</u>, addr, openDate)
Drinker(<u>name</u>, addr, phone)
Frequents[drinker] $\subseteq$ Drinker[name]

Frequents(<u>drinker, bar</u>)
Frequents[bar] $\subseteq$ Bars[name]

```
CREATE TABLE Bars (
    name VARCHAR(30), addr VARCHAR(30), openDate DATE,
    CONSTRAINT pk_bars_c0 PRIMARY KEY (name));
```

Un premier exemple

Bars (name, addr, openDate)
Drinker(name, addr, phone)
Frequents[drinker] $\subseteq$ Drinker[name]

Frequents(drinker, bar)
Frequents[bar] $\subseteq$ Bars[name]

```
CREATE TABLE Bars (
    name VARCHAR(30), addr VARCHAR(30), openDate DATE,
    CONSTRAINT pk_bars_c0 PRIMARY KEY (name));

CREATE TABLE Drinkers (
    name VARCHAR(30), addr VARCHAR(30), phone VARCHAR(30),
    CONSTRAINT pk_drinkers_c0 PRIMARY KEY (name)
);
```

Un premier exemple

Bars (name, addr, openDate)
Drinker(name, addr, phone)
Frequents[drinker] $\subseteq$ Drinker[name]

Frequents(drinker, bar)
Frequents[bar] $\subseteq$ Bars[name]

```
CREATE TABLE Bars (
    name VARCHAR(30), addr VARCHAR(30), openDate DATE,
    CONSTRAINT pk_bars_c0 PRIMARY KEY (name));

CREATE TABLE Drinkers (
    name VARCHAR(30), addr VARCHAR(30), phone VARCHAR(30),
    CONSTRAINT pk_drinkers_c0 PRIMARY KEY (name)
);

CREATE TABLE Frequents (
    drinker VARCHAR(30), bar VARCHAR(30),
    CONSTRAINT pk_frequents_c0 PRIMARY KEY (drinker,bar),
    CONSTRAINT fk_frequents_c1 FOREIGN KEY (bar) REFERENCES Bars(name),
    CONSTRAINT fk_frequents_c2 FOREIGN KEY (drinker) REFERENCES Drinkers(name)
);
```

Etat initial de la base de données

Drinkers

name	**addr**	**phone**
Adam	Randwick	9385-4444
Gernot	Newtown	9415-3378
John	Clovelly	9665-1234
Justin	Mosman	9845-4321
Marie	Rose Bay	9371-2126

Bars

name	**addr**	**openDate**
Coogee Bay Hotel	Coogee	1980-08-31
Lord Nelson	The Rocks	1920-11-11
Australia Hotel	The Rocks	1940-01-12
Regent Hotel	Kingsford	2000-02-29
Marble Bar	Sydney	2001-04-01
Rose Bay Hotel	Randwick	1986-06-26

Frequents

drinker	**bar**
Adam	Coogee Bay Hotel
Gernot	Lord Nelson
John	Coogee Bay Hotel
John	Lord Nelson
John	Australia Hotel
Justin	Regent Hotel
Justin	Marble Bar

Observation de mises à jour

```
SQL> INSERT INTO Bars VALUES ('Le bon coin', 'Grenoble',
       TO_DATE ('29/02/2022','DD/MM/YYYY'));
                    *
```

ERROR at line 1: ORA-01839: date not valid for month specified

Observation de mises à jour

```
SQL> INSERT INTO Bars VALUES ('Le bon coin', 'Grenoble',
       TO_DATE ('29/02/2022','DD/MM/YYYY'));
                    *
```

ERROR at line 1: ORA-01839: date not valid for month specified

```
SQL> INSERT INTO Drinkers VALUES ('Pierre', 'Grenoble');
               *
```

ERROR at line 1: ORA-00947: not enough values

Observation de mises à jour

```
SQL> INSERT INTO Bars VALUES ('Le bon coin', 'Grenoble',
       TO_DATE ('29/02/2022','DD/MM/YYYY'));
                    *
```

ERROR at line 1: ORA-01839: date not valid for month specified

```
SQL> INSERT INTO Drinkers VALUES ('Pierre', 'Grenoble');
                *
```

ERROR at line 1: ORA-00947: not enough values

```
SQL> INSERT INTO Drinkers VALUES ('Pierre', 'Grenoble', 'rrrrr');
```

1 row created.

Observation de mises à jour

```
SQL> INSERT INTO Bars VALUES ('Le bon coin', 'Grenoble',
       TO_DATE ('29/02/2022','DD/MM/YYYY'));
                    *
```

ERROR at line 1: ORA-01839: date not valid for month specified

```
SQL> INSERT INTO Drinkers VALUES ('Pierre', 'Grenoble');
                *
```

ERROR at line 1: ORA-00947: not enough values

```
SQL> INSERT INTO Drinkers VALUES ('Pierre', 'Grenoble', 'rrrrr');
```

1 row created.

```
SQL> INSERT INTO Drinkers VALUES ('Pierre', 'Voiron', 'aaaaa');
                *
```

ERROR at line 1: ORA-00001: unique constraint (BEERS.pk_drinkers_c0) violated

Observation du comportement de contraintes d'intégrité référentielle

```
SQL> INSERT INTO Frequents VALUES ('Pierre', 'Bar les amis');
INSERT INTO Frequents VALUES ('Pierre', 'Bar les amis')
*
```

ERROR at line 1: ORA-02291: integrity constraint (BEERS.fk_frequents_c1) violated - parent key not found

Observation du comportement de contraintes d'intégrité référentielle

```
SQL> INSERT INTO Frequents VALUES ('Pierre', 'Bar les amis');
INSERT INTO Frequents VALUES ('Pierre', 'Bar les amis')
*
```

ERROR at line 1: ORA-02291: integrity constraint (BEERS.fk_frequents_c1) violated - parent key not found

```
SQL> INSERT INTO Frequents VALUES ('Paul', 'Coogee Bay Hotel');
INSERT INTO Frequents VALUES ('Paul', 'Coogee Bay Hotel')
*
```

ERROR at line 1: ORA-02291: integrity constraint (BEERS.fk_frequents_c2) violated - parent key not found

Observation du comportement de contraintes d'intégrité référentielle (suite)

```
SQL> DELETE FROM Drinkers WHERE name='John';
DELETE FROM Drinkers WHERE name='John'
*
```

ERROR at line 1: ORA-02292: integrity constraint (BEERS.fk_frequents_c2) violated - child record found

Observation du comportement de contraintes d'intégrité référentielle (suite)

```
SQL> DELETE FROM Drinkers WHERE name='John';
DELETE FROM Drinkers WHERE name='John'
*
```

ERROR at line 1: ORA-02292: integrity constraint (BEERS.fk_frequents_c2) violated - child record found

```
SQL> UPDATE Drinkers SET name='Peter' WHERE name='Adam';
UPDATE Drinkers SET name='Peter' WHERE name='Adam'
*
```

ERROR at line 1: ORA-02292: integrity constraint (BEERS.fk_frequents_c2) violated - child record found

Règles associées aux contraintes d'intégrité référentielle

Permettent de spécifier le comportement du SGBD lors de la modification de données concernées par une contrainte d'intégrité.

Règles associées aux contraintes d'intégrité référentielle

Permettent de spécifier le comportement du SGBD lors de la modification de données concernées par une contrainte d'intégrité.

Exemple: Frequents[Drinker] $\subseteq$ Drinkers[Name]

Drinkers

name	**addr**	**phone**
Adam	Randwick	9385-4444
Gernot	Newtown	9415-3378
John	Clovelly	9665-1234
Justin	Mosman	9845-4321
Marie	Rose Bay	9371-2126

Frequents

drinker	**bar**
Adam	Coogee Bay Hotel
Gernot	Lord Nelson
John	Coogee Bay Hotel
John	Lord Nelson
John	Australia Hotel
Justin	Regent Hotel
Justin	Marble Bar

Règles associées aux contraintes d'intégrité référentielle

Permettent de spécifier le comportement du SGBD lors de la modification de données concernées par une contrainte d'intégrité.

Exemple: Frequents[Drinker] $\subseteq$ Drinkers[Name]

Drinkers

name	**addr**	**phone**
Adam	Randwick	9385-4444
Gernot	Newtown	9415-3378
John	Clovelly	9665-1234
Justin	Mosman	9845-4321
Marie	Rose Bay	9371-2126

Frequents

drinker	**bar**
Adam	Coogee Bay Hotel
Gernot	Lord Nelson
John	Coogee Bay Hotel
John	Lord Nelson
John	Australia Hotel
Justin	Regent Hotel
Justin	Marble Bar

- Supprimer <John, Clovelly> dans Drinkers ?
- Modifier "Adam" en "Peter" dans Drinkers ?
- Ajouter <Marie, Regent Hotel> dans Frequents ?

Plusieurs comportements possibles

- **Restrict**: interdit la modification ou la suppression des données référencées (**comportement par défaut**).

Plusieurs comportements possibles

- **Restrict**: interdit la modification ou la suppression des données référencées (**comportement par défaut**).

- **Set to null (resp. DEFAULT)** : lorsque la donnée référencée est modifiée ou supprimée toutes les données qui en dépendaient sont mises à NULL (resp. à la valeur par défaut). Ce qui pose problème lorsque ces dernières participent à une clef.

Plusieurs comportements possibles

- **Restrict**: interdit la modification ou la suppression des données référencées (**comportement par défaut**).
- **Set to null (resp. DEFAULT)** : lorsque la donnée référencée est modifiée ou supprimée toutes les données qui en dépendaient sont mises à NULL (resp. à la valeur par défaut). Ce qui pose problème lorsque ces dernières participent à une clef.
- **Cascade** : lorsque la donnée référencée est modifiée ou supprimée toutes les données associées sont mises à jour en conséquence (modifiées ou supprimées).

Sur l'exemple précédent :

CREATE TABLE Frequents (

....

CONSTRAINT fk_drinker FOREIGN KEY (drinker) REFERENCES Drinkers(name)
ON DELETE CASCADE ON UPDATE CASCADE

...

- Absence de la clause on .. cascade $\Longrightarrow$ restrict
- ON UPDATE CASCADE $\Longrightarrow$ "Adam" est remplacé par "Peter" dans Frequents
- ON DELETE CASCADE $\Longrightarrow$ tous les n-uplets correspondant à "John" sont supprimés dans Frequents

Exemple : une (autre) bibliothèque

Spécification des relations :

Livres (titre, dateEdition)
Exemplaires (numEx, dateAchat, titre)
    titre not null
    Exemplaires[titre] = Livres[titre]
Emprunts (numEx, dateEmp, dateRetour, numAdh)
/ (numEx,dateRetour) est une autre clef */*
    dateEmp $<$ dateRetour
    numAdh not null
    Emprunts[numEx] $\subseteq$ Exemplaires[numEx]
domaine (dateEmp) = domaine (dateRetour) = Date(Heure)
domaine (dateAchat) = domaine (dateEdition) = Date(Jour)

Exemple : une (autre) bibliothèque

Spécification des relations :

Livres (titre, dateEdition)
Exemplaires (numEx, dateAchat, titre)
    titre not null
    Exemplaires[titre] = Livres[titre]
Emprunts (numEx, dateEmp, dateRetour, numAdh)
/ (numEx,dateRetour) est une autre clef */*
    dateEmp < dateRetour
    numAdh not null
    Emprunts[numEx] $\subseteq$ Exemplaires[numEx]
domaine (dateEmp) = domaine (dateRetour) = Date(Heure)
domaine (dateAchat) = domaine (dateEdition) = Date(Jour)
- A chaque instant, chaque exemplaire est emprunté au plus une fois
- Chaque adhérent ne peut pas faire plus de 4 emprunts à la fois

En LDD SQL :

```
CREATE TABLE Livres (
    titre VARCHAR(30) PRIMARY KEY,
    dateEdition DATE);
```

En LDD SQL :

```
CREATE TABLE Livres (
   titre VARCHAR(30) PRIMARY KEY,
   dateEdition DATE);
CREATE TABLE Exemplaires (
   numEx NUMBER(5) PRIMARY KEY,
   dateAchat DATE,
   titre VARCHAR(30) NOT NULL,
   CONSTRAINT fk_exemplaires_c1 FOREIGN KEY (titre) REFERENCES Livres (titre)
      ON DELETE CASCADE ON UPDATE CASCADE );
```

En LDD SQL :

```
CREATE TABLE Livres (
   titre VARCHAR(30) PRIMARY KEY,
   dateEdition DATE);
CREATE TABLE Exemplaires (
   numEx NUMBER(5) PRIMARY KEY,
   dateAchat DATE,
   titre VARCHAR(30) NOT NULL,
   CONSTRAINT fk_exemplaires_c1 FOREIGN KEY (titre) REFERENCES Livres (titre)
      ON DELETE CASCADE ON UPDATE CASCADE );
CREATE TABLE Emprunts (
   numEx NUMBER(5),
   dateEmp DATE,
   dateRetour DATE,
   numAdh VARCHAR(10) NOT NULL,
   CONSTRAINT pk_emprunts_c1 PRIMARY KEY (numEx, dateEmp),
   CONSTRAINT uq_emprunts_c2 UNIQUE (numEx, dateRetour),
   CONSTRAINT fk_emprunts_c3 FOREIGN KEY (numEx)
      REFERENCES Exemplaires (numEx) ON DELETE CASCADE ON UPDATE CASCADE,
   CONSTRAINT ch_emprunts_c4 CHECK (dateEmp < dateRetour) );
```

Quelles sont les contraintes à la charge de l'application ?

- A chaque instant, chaque exemplaire est emprunté au plus une fois
- Chaque adhérent ne peut pas faire plus de 4 emprunts à la fois
- Qu'en est-il de Exemplaires[titre] = Livres[titre] ?

Spécifications des contraintes : bonnes pratiques

Les contraintes doivent être nommées :

- Maintenance du schéma : seules les contraintes nommées peuvent être désactivées/réactivées
- Gestion des exceptions dans les programmes : facilite l'analyse des exceptions retournées par le SGBD.

Attention aux règles de nommage !

En résumé

- Attention à l'ordre de définition des relations !
- Contraintes d'intégrité à la charge des applications :
  - Granularité du temps
    (ex. domaine (dateRetour) = Date(Heure))
  - Restriction de valeur basée sur plus d'une relation
    (ex. Chaque membre ne peut faire plus de 4 emprunts à la fois)
  - Contraintes d'intégrité référentielle définies sur des attributs non clés
  - Contraintes d'intégrité référentielle bi-directionnelles
  - Etc.
- Options "ON DELETE CASCADE", "ON UPDATE CASCADE" ?

Supprimer une relation

Afin de satisfaire les contraintes d'intégrité toute modification de schéma entraîne l'ajustement des données.

DROP TABLE T; supprime la relation T et ses n-uplets

DROP TABLE Emprunts ;

Lorsque T est cible d'une contrainte d'intégrité référentielle, T ne peut pas être supprimée.

Modifier des relations

Il est possible de faire toute modification qui n'implique pas de suppression de données :

- Ajouter un attribut (chaque tuple existant dans la relation a la valeur absente pour le nouvel attribut).
- Ajouter une contrainte d'intégrité
- Redéfinir un attribut (type, taille, valeur par défaut)
- Activer, désactiver ou supprimer une contrainte d'intégrité

Quelques exemples :

```
ALTER TABLE LesEtudiants ADD (anniv DATE, adresse VARCHAR(20)) ;
ALTER TABLE LesEtudiants MODIFY (adresse VARCHAR (30)) ;
ALTER TABLE LesEtudiants DISABLE CONSTRAINT LesEtudiants_c1 ;
```

Insérer des données dans des relations

Insérer des données :

- *INSERT INTO nom_relation VALUES ...*

Deux options :

```
/* un n-uplet à la fois */
   INSERT INTO Batiments VALUES (210, 'John') ;
   INSERT INTO Exemplaires VALUES (12, SYSDATE, 'Dune') ;
/* Pour une relation existante */
   INSERT INTO Batiments
      SELECT A, B FROM R WHERE .....
```

Rappel : une requête est exécutée uniquement lorsqu'elle construit un état de la base de données qui est cohérent.

Modifier des données

Modifier des données :

- *UPDATE nom_relation*
  SET $nom\_attribut_1$ =
  $nouvelle\_val_1$, ...
  $nom\_attribut_N$ =
  $nouvelle\_val_N$
  WHERE condition ; — la clause WHERE est optionnelle

Exemple :

```
UPDATE LesEmployes
   SET salaire = salaire * 1.2
   WHERE salaire < 10000 ;
```

Supprimer des données

Supprimer des données : Soient : Batiments (noB, nom) et LesEtages (noEt, noB, acces)
LesEtages[noB] $\subseteq$ Batiments[noB]

- *DELETE FROM nom_relation WHERE condition ;*

La clause WHERE est optionnelle
Exemple :

DELETE FROM Batiments ;

Supprime tous les n-uplets des deux relations Batiments et LesEtages (à cause de l'option ON DELETE CASCADE)
Les modifications sont faites si et seulement si elles satisfont toutes les contraintes de la base de données définies en SQL.

Le langage de définition et manipulation des données de SQL permet de :

- Créer, ajouter, modifier des relations
- Spécifier certaines contraintes
  D'autres restent sous la responsabilité des programmes applicatifs :
  - Restrictions de domaines spécifiques (le salaire d'un employé ne peut être plus haut que celui de son chef)
  - Contraintes dynamiques (un salaire ne peut diminuer)
- ...

Possibilités des systèmes non étudiées ici : index, user, grant, trigger, views, ...