Bases de données et langage SQL

Cours de spécialité NSI de Terminale

D Pihoué

Lycée Camille Jullian Bordeaux

4 décembre 2023

Capacités attendues

1. Identifier les concepts définissant le modèle relationnel.
2. Savoir distinguer la structure d'une base de données de son contenu.
3. Repérer des anomalies dans le schéma d'une base de données.
4. Identifier les composants d'une requête.
5. Construire des requêtes d'interrogation à l'aide des clauses du langage SQL :

   `SELECT`, `FROM`, `WHERE`, `JOIN`.

6. Construire des requêtes d'insertion et de mise à jour à l'aide de :

   `UPDATE`, `INSERT`, `DELETE`.

Définition

Un **système d'information** est un système technique et humain qui permet de gérer de l'information.

Définition

Un **système d'information** est un système technique et humain qui permet de gérer de l'information.

Exemple

Considérons une version simplifiée d'une médiathèque municipale.

Définition

Un **système d'information** est un système technique et humain qui permet de gérer de l'information.

Exemple

Considérons une version simplifiée d'une médiathèque municipale. Le système d'information contient alors

- une description d'objets, ceux qui peuvent être consultés ou empruntés,
- une description d'usagers, les personnes physiques qui consultent ou empruntent,
- une description de processus, les emprunts et les retours.

Exemple

Le système d'information doit aussi

Exemple

Le système d'information doit aussi

- permettre le stockage des données,
- garantir l'absence d'erreur lors des déroulements des processus,
- proposer différents niveaux d'accès aux données en distinguant l'usage des niveaux d'administration,
- reposer sur une architecture permettant la consultation simultanée depuis différentes machines.

Exemple

Le système d'information doit aussi

- permettre le stockage des données,
- garantir l'absence d'erreur lors des déroulements des processus,
- proposer différents niveaux d'accès aux données en distinguant l'usage des niveaux d'administration,
- reposer sur une architecture permettant la consultation simultanée depuis différentes machines.

Ce système d'information ne contient qu'une approximation de la réalité.

Exemple

Le système d'information doit aussi

- permettre le stockage des données,
- garantir l'absence d'erreur lors des déroulements des processus,
- proposer différents niveaux d'accès aux données en distinguant l'usage des niveaux d'administration,
- reposer sur une architecture permettant la consultation simultanée depuis différentes machines.

Ce système d'information ne contient qu'une approximation de la réalité.
Cette approximation définit une **modélisation**.

L'un des modèles de données les plus utilisés est le **modèle relationnel** qui a été défini en 1970 par l'informaticien britannique Edgar Franck CODD (1923 - 2003), lauréat du prix Turing en 1981.

L'un des modèles de données les plus utilisés est le **modèle relationnel** qui a été défini en 1970 par l'informaticien britannique Edgar Franck CODD (1923 - 2003), lauréat du prix Turing en 1981.

Modèle relationnel

L'un des modèles de données les plus utilisés est le **modèle relationnel** qui a été défini en 1970 par l'informaticien britannique Edgar Franck CODD (1923 - 2003), lauréat du prix Turing en 1981.

Modèle relationnel

- Dans ce modèle, un objet modélisé que l'on désigne par **entité**, est représenté par un n-uplet de valeurs scalaires, désignées comme étant les **attributs**, chacune étant d'un type simple non structuré.

L'un des modèles de données les plus utilisés est le **modèle relationnel** qui a été défini en 1970 par l'informaticien britannique Edgar Franck CODD (1923 - 2003), lauréat du prix Turing en 1981.

Modèle relationnel

- Dans ce modèle, un objet modélisé que l'on désigne par **entité**, est représenté par un n-uplet de valeurs scalaires, désignées comme étant les **attributs**, chacune étant d'un type simple non structuré.
- Une collection d'entités, qui est donc un ensemble de n-uplets, est appelée une **relation**.

L'un des modèles de données les plus utilisés est le **modèle relationnel** qui a été défini en 1970 par l'informaticien britannique Edgar Franck CODD (1923 - 2003), lauréat du prix Turing en 1981.

Modèle relationnel

- Dans ce modèle, un objet modélisé que l'on désigne par **entité**, est représenté par un n-uplet de valeurs scalaires, désignées comme étant les **attributs**, chacune étant d'un type simple non structuré.
- Une collection d'entités, qui est donc un ensemble de n-uplets, est appelée une **relation**.
- Toute relation est conforme à un **schéma**, description ordonnée indiquant pour chaque composante de la relation son **nom** et son **domaine** de valeurs.

Définition

Une base de données est un ensemble de relations et son schéma est l'ensemble des schémas de ces relations.

Définition

Une base de données est un ensemble de relations et son schéma est l'ensemble des schémas de ces relations.

Les domaines de valeurs que l'on peut considérer sont `String`, `Int`, `Boolean`, `Float`, `Date` pour (jour/mois/année) et `Time` pour un instant (heure :minute :seconde).

Définition

Une base de données est un ensemble de relations et son schéma est l'ensemble des schémas de ces relations.

Les domaines de valeurs que l'on peut considérer sont `String`, `Int`, `Boolean`, `Float`, `Date` pour (jour/mois/année) et `Time` pour un instant (heure :minute :seconde).

Exemple

Donnons les schémas des relations *Livre* et *Usager* pour l'exemple de la médiathèque.

Définition

Une base de données est un ensemble de relations et son schéma est l'ensemble des schémas de ces relations.

Les domaines de valeurs que l'on peut considérer sont `String`, `Int`, `Boolean`, `Float`, `Date` pour (jour/mois/année) et `Time` pour un instant (heure :minute :seconde).

Exemple

Donnons les schémas des relations *Livre* et *Usager* pour l'exemple de la médiathèque.

Livre(*titre* `String`, *éditeur* `String`, *année* `Int`, *isbn* `String`)

Définition

Une base de données est un ensemble de relations et son schéma est l'ensemble des schémas de ces relations.

Les domaines de valeurs que l'on peut considérer sont `String`, `Int`, `Boolean`, `Float`, `Date` pour (jour/mois/année) et `Time` pour un instant (heure :minute :seconde).

Exemple

Donnons les schémas des relations *Livre* et *Usager* pour l'exemple de la médiathèque.
Livre(*titre* `String`, *éditeur* `String`, *année* `Int`, *isbn* `String`)
Usager(*nom* `string`, *prénom* `String`, *code_barre* `string`)

Démarche de modélisation

La **modélisation relationnelle des données** se décompose en trois étapes.

Démarche de modélisation

La **modélisation relationnelle des données** se décompose en trois étapes.

1. Déterminer les **entités** (objets, personnes, actions, etc.) que l'on souhaite manipuler.

Démarche de modélisation

La **modélisation relationnelle des données** se décompose en trois étapes.

1. Déterminer les **entités** (objets, personnes, actions, etc.) que l'on souhaite manipuler.
2. Modéliser les ensembles d'entités comme des **relations** en donnant leur **schéma**, en s'attachant particulièrement à choisir le bon **domaine** pour chaque **attribut**.

Démarche de modélisation

La **modélisation relationnelle des données** se décompose en trois étapes.

1. Déterminer les **entités** (objets, personnes, actions, etc.) que l'on souhaite manipuler.
2. Modéliser les ensembles d'entités comme des **relations** en donnant leur **schéma**, en s'attachant particulièrement à choisir le bon **domaine** pour chaque **attribut**.
3. Définir les **contraintes**, c'est-à-dire l'ensemble des propriétés logiques vérifiées à tout instant par les données.

Démarche de modélisation

La **modélisation relationnelle des données** se décompose en trois étapes.

1. Déterminer les **entités** (objets, personnes, actions, etc.) que l'on souhaite manipuler.
2. Modéliser les ensembles d'entités comme des **relations** en donnant leur **schéma**, en s'attachant particulièrement à choisir le bon **domaine** pour chaque **attribut**.
3. Définir les **contraintes**, c'est-à-dire l'ensemble des propriétés logiques vérifiées à tout instant par les données.

Les **contraintes d'intégrité** assurent la cohérence de la base de données.

Quatre contraintes d'intégrité

Quatre contraintes d'intégrité

Contrainte de domaine Il s'agit de toutes les propriétés que le choix du domaine va permettre de garantir.

Quatre contraintes d'intégrité

Contrainte de domaine Il s'agit de toutes les propriétés que le choix du domaine va permettre de garantir.

Contrainte d'entité Elle identifie une entité de manière non ambiguë et assure que chaque élément d'une relation est unique.

Une **clef primaire** est un attribut ou un ensemble d'attributs de la relation qui garantit cette contrainte.

Quatre contraintes d'intégrité

Contrainte de domaine Il s'agit de toutes les propriétés que le choix du domaine va permettre de garantir.

Contrainte d'entité Elle identifie une entité de manière non ambiguë et assure que chaque élément d'une relation est unique.
Une **clef primaire** est un attribut ou un ensemble d'attributs de la relation qui garantit cette contrainte.

Contrainte de référence Une clef primaire d'une relation peut aussi servir de référence dans une autre relation. Elle est alors qualifiée de **clef étrangère**.

Quatre contraintes d'intégrité

Contrainte de domaine Il s'agit de toutes les propriétés que le choix du domaine va permettre de garantir.

Contrainte d'entité Elle identifie une entité de manière non ambiguë et assure que chaque élément d'une relation est unique.
Une **clef primaire** est un attribut ou un ensemble d'attributs de la relation qui garantit cette contrainte.

Contrainte de référence Une clef primaire d'une relation peut aussi servir de référence dans une autre relation. Elle est alors qualifiée de **clef étrangère**.

Contraintes utilisateurs parfois aussi appelées **contraintes métiers** sont toutes celles que l'on ne peut pas exprimer dans l'une des trois catégories précédentes.

Reprenons la version simplifiée d'une médiathèque municipale.

Reprenons la version simplifiée d'une médiathèque municipale.

Exemple

Contrainte de domaine Enrichir la relation *Usager* en ajoutant des attributs pour l'adresse postale et l'adresse mail.

Reprenons la version simplifiée d'une médiathèque municipale.

Exemple

Contrainte de domaine Enrichir la relation *Usager* en ajoutant des attributs pour l'adresse postale et l'adresse mail.

Contrainte d'entité Quelles clefs primaires pour ces deux relations ?

Reprenons la version simplifiée d'une médiathèque municipale.

Exemple

Contrainte de domaine Enrichir la relation *Usager* en ajoutant des attributs pour l'adresse postale et l'adresse mail.

Contrainte d'entité Quelles clefs primaires pour ces deux relations ?

Contrainte de référence Définir une relation *Emprunt* ainsi qu'une relation *Auteur*.

Reprenons la version simplifiée d'une médiathèque municipale.

Exemple

Contrainte de domaine Enrichir la relation *Usager* en ajoutant des attributs pour l'adresse postale et l'adresse mail.

Contrainte d'entité Quelles clefs primaires pour ces deux relations ?

Contrainte de référence Définir une relation *Emprunt* ainsi qu'une relation *Auteur*.

Contraintes utilisateurs Quelles pourraient être ces contraintes ?

Démarche en deux étapes

Démarche en deux étapes

1. Élaboration d'un **modèle conceptuel des données** qui va permettre de représenter la solution envisagée sans tenir compte des choix techniques d'implémentation.

Démarche en deux étapes

1. Élaboration d'un **modèle conceptuel des données** qui va permettre de représenter la solution envisagée sans tenir compte des choix techniques d'implémentation.
2. Déduction d'un **schéma relationnel** de la base de données à partir du modèle précédent.

Démarche en deux étapes

1. Élaboration d'un **modèle conceptuel des données** qui va permettre de représenter la solution envisagée sans tenir compte des choix techniques d'implémentation.
2. Déduction d'un **schéma relationnel** de la base de données à partir du modèle précédent.

On peut recourir à un modèle entité - association pour réaliser la première étape.

Plus concrètement, pour réaliser cette première étape, il faut :

Plus concrètement, pour réaliser cette première étape, il faut :

1. identifier les objets à représenter avec leurs attributs, ce sont les **entités** et déterminer pour chaque entité un **identifiant** qui pourra jouer le rôle de clef primaire dans le schéma relationnel ;

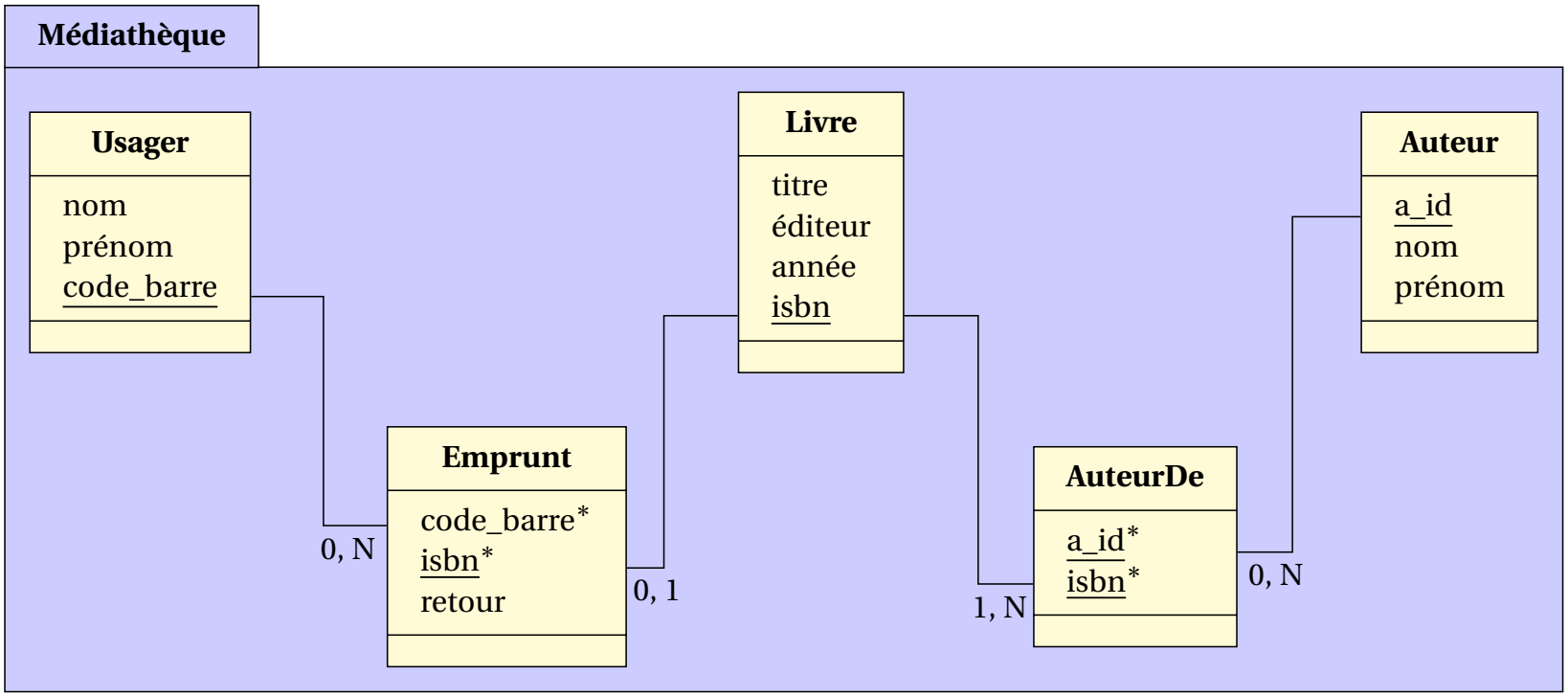
Plus concrètement, pour réaliser cette première étape, il faut :

1. identifier les objets à représenter avec leurs attributs, ce sont les **entités** et déterminer pour chaque entité un **identifiant** qui pourra jouer le rôle de clef primaire dans le schéma relationnel ;
2. exprimer les liens qui peuvent exister entre ces entités, ce sont les **associations** puis enfin déterminer les **cardinalités** de ces associations.

Plus concrètement, pour réaliser cette première étape, il faut :

1. identifier les objets à représenter avec leurs attributs, ce sont les **entités** et déterminer pour chaque entité un **identifiant** qui pourra jouer le rôle de clef primaire dans le schéma relationnel ;
2. exprimer les liens qui peuvent exister entre ces entités, ce sont les **associations** puis enfin déterminer les **cardinalités** de ces associations.

Pour l'exemple de la médiathèque, cette première démarche conduirait au schéma suivant.


Médiathèque
Usager
nom
prénom
code_barre
Livre
titre
éditeur
année
isbn
Auteur
a_id
nom
prénom
Emprunt
code_barre*
isbn*
retour
AuteurDe
a_id*
isbn*
0, N
0, 1
1, N
0, N

Pour la deuxième étape, on déduit le modèle relationnel des données en appliquant au modèle conceptuel les trois règles qui suivent.

Pour la deuxième étape, on déduit le modèle relationnel des données en appliquant au modèle conceptuel les trois règles qui suivent.

Règles de transposition

Pour la deuxième étape, on déduit le modèle relationnel des données en appliquant au modèle conceptuel les trois règles qui suivent.

Règles de transposition

1. Chaque entité du modèle conceptuel devient une relation du modèle relationnel. L'identifiant constitue la clef primaire.

Pour la deuxième étape, on déduit le modèle relationnel des données en appliquant au modèle conceptuel les trois règles qui suivent.

Règles de transposition

1. Chaque entité du modèle conceptuel devient une relation du modèle relationnel. L'identifiant constitue la clef primaire.
2. Si la cardinalité d'une association est au plus égale à 1 alors l'identifiant de l'entité origine constitue une clef étrangère de la relation destination.

Pour la deuxième étape, on déduit le modèle relationnel des données en appliquant au modèle conceptuel les trois règles qui suivent.

Règles de transposition

1. Chaque entité du modèle conceptuel devient une relation du modèle relationnel. L'identifiant constitue la clef primaire.
2. Si la cardinalité d'une association est au plus égale à 1 alors l'identifiant de l'entité origine constitue une clef étrangère de la relation destination.
3. Si la cardinalité d'une association a un maximum égal à N alors une nouvelle relation est créée et porte le nom de l'association. Des clefs étrangères sont alors créées pour toutes les clefs primaires des relations liées. La clef primaire de cette relation est composée de toutes ces clefs étrangères.

Le langage SQL qui permet d'envoyer des ordres au système :

- soit de **mise à jour** comme la création de relations, l'ajout d'entités dans des relations ou leur modification et leur suppression ;
- soit de **requêtes** pour récupérer des données sous certains critères particuliers.

Le langage SQL qui permet d'envoyer des ordres au système :

- soit de **mise à jour** comme la création de relations, l'ajout d'entités dans des relations ou leur modification et leur suppression ;
- soit de **requêtes** pour récupérer des données sous certains critères particuliers.

Ce langage a été normalisé plusieurs fois de 1986 à 2011, d'abord par l'ANSI, pour *American National Standards Institute* puis par l'ISO, pour *International Organization for Standization* dès 1987 comme norme internationale sous la référence ISO/IEC 9075 dont la dernière version date de 2016.

Le langage SQL qui permet d'envoyer des ordres au système :

- soit de **mise à jour** comme la création de relations, l'ajout d'entités dans des relations ou leur modification et leur suppression ;
- soit de **requêtes** pour récupérer des données sous certains critères particuliers.

Ce langage a été normalisé plusieurs fois de 1986 à 2011, d'abord par l'ANSI, pour *American National Standards Institute* puis par l'ISO, pour *International Organization for Standization* dès 1987 comme norme internationale sous la référence ISO/IEC 9075 dont la dernière version date de 2016.
Par convention, les **ordres** sont écrits en lettres majuscules, les **noms** des relations et des entités le sont en lettres minuscules, au singulier et sans espace.

Le langage permet de créer des relations, on dit des **tables**, en spécifiant leur nom, leurs attributs, les domaines de valeurs, on dit les **types**, et les contraintes associées à la table.

Le langage permet de créer des relations, on dit des **tables**, en spécifiant leur nom, leurs attributs, les domaines de valeurs, on dit les **types**, et les contraintes associées à la table.

Exemple

```
CREATE TABLE usager (nom VARCHAR(90),
  prenom VARCHAR(90),
  adresse VARCHAR(300), cp VARCHAR(5),
  ville VARCHAR(60), email VARCHAR(60),
  code_barre CHAR(15) PRIMARY KEY);

CREATE TABLE livre (titre VARCHAR(300),
  editeur VARCHAR(90), annee INT,
  isbn CHAR(14) PRIMARY KEY);
```

Exemple

```
CREATE TABLE auteur (a_id INT PRIMARY KEY,
  nom VARCHAR(90),
  prenom VARCHAR(90));

CREATE TABLE auteur_de (
  a_id INT REFERENCES auteur(a_id),
  isbn CHAR(14) REFERENCES livre(isbn),
  PRIMARY KEY (a_id, isbn));
```

Exemple

```
CREATE TABLE emprunt (
  code_barre CHAR(15)
    REFERENCES usager(code_barre),
  isbn CHAR(14) PRIMARY KEY
    REFERENCES livre(isbn),
  retour DATE);
```

Exemple

```
CREATE TABLE emprunt (
  code_barre CHAR(15)
    REFERENCES usager(code_barre),
  isbn CHAR(14) PRIMARY KEY
    REFERENCES livre(isbn),
  retour DATE);
```

Un ordre se termine par la ponctuation ;.

La syntaxe générale de création d'une table est donc

Création d'une table

```
CREATE TABLE <nom_table> (
  <attribut 1> <domaine 1> <contrainte 1>,
  <attribut 2> <domaine 2> <contrainte 2>,
  ...,
  <contrainte globale 1>,
  ...,
  <contrainte globale n>);
```

Les types SQL permettant de spécifier les domaines de valeurs figurent dans un document disponible dans le cours en ligne.

Les types SQL permettant de spécifier les domaines de valeurs figurent dans un document disponible dans le cours en ligne.

Insertion dans une table

```
INSERT INTO <nom_table> VALUES (valeurs 1),
  (valeurs 2),
  ...,
  (valeurs n);
```

Les types SQL permettant de spécifier les domaines de valeurs figurent dans un document disponible dans le cours en ligne.

Insertion dans une table

```
INSERT INTO <nom_table> VALUES (valeurs 1),
  (valeurs 2),
  ...,
  (valeurs n);
```

Chaque n-uplet écrit entre parenthèses représente une nouvelle ligne dans la table et doit donc respecter le schéma de la relation.

La sélection de données consiste en l'écriture de **requêtes SQL** permettant de trouver toutes les données de la base qui vérifient certaines conditions.

La sélection de données consiste en l'écriture de **requêtes SQL** permettant de trouver toutes les données de la base qui vérifient certaines conditions.

Sélection et projection

```
SELECT <nom_attribut_1>,
  <nom_attribut_2>,
  ...,
  <nom_attribut_n>
  FROM <nom_table>
  WHERE <conditions>
;
```

Des mots clefs, expliqués en annexe, permettent d'affiner la requête.

Des mots clefs, expliqués en annexe, permettent d'affiner la requête.

Sélection et projection

```
SELECT DISTINCT
  <nom_attribut_1> AS <alias_1>,
  <nom_attribut_2>
  FROM <nom_table>
  WHERE <conditions>
;
```

Des mots clefs, expliqués en annexe, permettent d'affiner la requête.

Sélection et projection

```
SELECT DISTINCT
  <nom_attribut_1> AS <alias_1>,
  <nom_attribut_2>
  FROM <nom_table>
  WHERE <conditions>
;
```

pour éviter les doublons quand ils existent.

On peut aussi trier les résultats renvoyés et en limiter le nombre.

On peut aussi trier les résultats renvoyés et en limiter le nombre.

Sélection et projection

```
SELECT *
  FROM <nom_table>
  ORDER BY <nom_attribut_1> ASC,
    <nom_attribut_2> DESC
  LIMIT 20
;
```

Les fonctions d'agrégation permettent d'appliquer une fonction à l'ensemble des valeurs d'un champ et de renvoyer le résultat comme une table d'un seul enregistrement et d'un seul champ.

Les fonctions d'agrégation permettent d'appliquer une fonction à l'ensemble des valeurs d'un champ et de renvoyer le résultat comme une table d'un seul enregistrement et d'un seul champ.

Exemple

- `COUNT` permet d'obtenir le nombre de résultats renvoyés
- `AVG`, pour *average*, permet de calculer la moyenne des valeurs d'un champ
- `SUM` permet de calculer la somme des valeurs d'un champ
- `MIN` et `MAX` permettent de trouver le minimum et le maximum parmi les valeurs d'un champ.

Un exemple de syntaxe

Un exemple de syntaxe

Fonctions d'agrégation

```
SELECT SUM(<attribut_1>) AS total
  FROM <nom_table>
  WHERE <conditions>
;
```

Où les valeurs du champ <attribut_1> sont d'un type numérique compatible avec la fonction SUM.

Définition

Étant données deux tables `<tableA>` et `<tableB>`, l'opération de jointure consiste à créer toutes les combinaisons d'enregistrements de `<tableA>` et de `<tableB>` ayant un champ de même valeur.

Définition

Étant données deux tables `<tableA>` et `<tableB>`, l'opération de jointure consiste à créer toutes les combinaisons d'enregistrements de `<tableA>` et de `<tableB>` ayant un champ de même valeur. Cet attribut est généralement une clé primaire ou une clef étrangère.

Jointure

```
SELECT <attribut_1>, <attribut_2>
  FROM <tableA>
  JOIN <tableB>
    ON <tableA>.<cléA> = <tableB>.<cléB>
  WHERE <conditions>
;
```

Jointure

```
SELECT <attribut_1>, <attribut_2>
  FROM <tableA>
  JOIN <tableB>
    ON <tableA>.<cléA> = <tableB>.<cléB>
  WHERE <conditions>
;
```

Les attributs peuvent être issus d'une table ou de l'autre.

Jointure

```
SELECT <attribut_1>, <attribut_2>
  FROM <tableA>
  JOIN <tableB>
    ON <tableA>.<cléA> = <tableB>.<cléB>
  WHERE <conditions>
;
```

Les attributs peuvent être issus d'une table ou de l'autre.
Il convient d'attribuer un alias à la table avec le mot clef `AS` pour les distinguer.

Jointure avec alias

```
SELECT <tA>.<attribut_1>, <tB>.<attribut_2>
  FROM <tableA> AS tA
  JOIN <tableB> AS tB
    ON <tA>.<cléA> = <tB>.<cléB>
  WHERE <conditions>
;
```

Jointure avec alias

```
SELECT <tA>.<attribut_1>, <tB>.<attribut_2>
  FROM <tableA> AS tA
  JOIN <tableB> AS tB
    ON <tA>.<cléA> = <tB>.<cléB>
  WHERE <conditions>
;
```

Il est possible d'ajouter une deuxième clause `JOIN` après la première pour réaliser une jointure avec une autre table.

On peut ajouter des données dans une table à partir du résultat d'une requête de sélection.

On peut ajouter des données dans une table à partir du résultat d'une requête de sélection.

Ajout de données après sélection

```
INSERT INTO <tableB>
  (<attribut_B1>, <attribut_B2>, ...)
  SELECT <attribut_A1>, <attribut_A2>, ...
  FROM <tableA>
  WHERE <conditions>
;
```

On peut ajouter des données dans une table à partir du résultat d'une requête de sélection.

Ajout de données après sélection

```
INSERT INTO <tableB>
  (<attribut_B1>, <attribut_B2>, ...)
  SELECT <attribut_A1>, <attribut_A2>, ...
  FROM <tableA>
  WHERE <conditions>
;
```

Cet ordre ne fait que copier les données sans transmettre les contraintes d'intégrité.

Les deux ordres SQL qui permettent la copie intégrale d'une table suivent la syntaxe suivante

Les deux ordres SQL qui permettent la copie intégrale d'une table suivent la syntaxe suivante

Copie d'une table avec les contraintes d'intégrité

```
CREATE TABLE <tableSecours>
  (LIKE <tableInitiale> INCLUDING ALL)
;
INSERT INTO <tableSecours>
  (SELECT * FROM <tableInitiale>)
;
```

La syntaxe générale

La syntaxe générale

Suppression d'enregistrements

```
DELETE FROM <nom_table>
  WHERE <conditions>
;
```

La syntaxe générale

Suppression d'enregistrements

```
DELETE FROM <nom_table>
  WHERE <conditions>
;
```

permet de supprimer tous les enregistrements de la table `<nom_table>` qui vérifient les conditions énoncées après la clause `WHERE` si aucune contrainte d'intégrité n'est violée par cet ordre sinon la table reste en l'état.

La mise à jour consiste à remplacer les valeurs de certains attributs par d'autres valeurs.

La mise à jour consiste à remplacer les valeurs de certains attributs par d'autres valeurs.

Mise à jour de valeurs

```
UPDATE <nom_table>
  SET <attribut_1> = <valeur_1>,
  SET <attribut_2> = <valeur_2>,
  ...,
  SET <attribut_n> = <valeur_n>
  WHERE <conditions>
;
```

La mise à jour consiste à remplacer les valeurs de certains attributs par d'autres valeurs.

Mise à jour de valeurs

```
UPDATE <nom_table>
  SET <attribut_1> = <valeur_1>,
  SET <attribut_2> = <valeur_2>,
  ...,
  SET <attribut_n> = <valeur_n>
  WHERE <conditions>
;
```

Les expressions <valeur_i> de mise à jour peuvent mentionner des noms d'attributs.

La requête renvoyant le résultat intermédiaire est désignée comme **sous-requête** et elle est exprimée entre parenthèses.

La requête renvoyant le résultat intermédiaire est désignée comme **sous-requête** et elle est exprimée entre parenthèses.
La requête exploitant le résultat est la **requête principale**.

La requête renvoyant le résultat intermédiaire est désignée comme **sous-requête** et elle est exprimée entre parenthèses.
La requête exploitant le résultat est la **requête principale**.
L'exécution s'effectue depuis la sous-requête la plus profonde dans l'imbrication vers la requête principale, chacune se fondant sur les résultats renvoyés par la précédente.

Placée après la clause `SELECT`, une sous-requête peut renvoyer un attribut calculé.

Placée après la clause `SELECT`, une sous-requête peut renvoyer un attribut calculé.

Requêtes imbriquée premier cas

```
SELECT ...,
  (SELECT ... FROM ... WHERE ...) AS ...
  FROM ...
  WHERE ...
;
```

Placée après `FROM` une sous-requête permet de pré-sélectionner les enregistrements sur lesquels opérera la requête principale.

Placée après `FROM` une sous-requête permet de pré-sélectionner les enregistrements sur lesquels opérera la requête principale.

Requêtes imbriquée deuxième cas

```
SELECT ... FROM
  (SELECT ... FROM ... WHERE ...) AS ...
  WHERE ...
;
```

Placée après la clause `WHERE` une sous-requête permet d'établir une ou des valeurs possibles sur lesquelles opérera la condition.

Placée après la clause `WHERE` une sous-requête permet d'établir une ou des valeurs possibles sur lesquelles opérera la condition.

Requêtes imbriquée troisième cas

```
SELECT ...
  FROM ...
  WHERE ... IN[IS]
  (SELECT ... FROM ... WHERE ...)
;
```