Exemple introductif

Terminologie

Mise en pratique : le langage SQL

Bases de données

26 juin 2022

Table of Contents

Exemple introductif

Terminologie

Mise en pratique : le langage SQL

On souhaite encoder des listes d'élèves de CPGE, avec certaines de leurs caractéristiques.

```
Eleves = [
 [ "MPI", ["BAUDELAIRE", "Charles", "E"],
 ["DESCARTES", "René", "A"], ...],
 [ "MPSI", (...), (...), ...],
 [ "PCSI", (...), (...), ...],
 [ "PSI", (...), (...), ...],
 [ "MP", (...), (...), ...]]
```

Si on cherche à accéder à une partie précise de ces informations, le coût dépendra de la position de l'information et de la structure de données utilisée pour les représenter.

Si on cherche à accéder à une partie précise de ces informations, le coût dépendra de la position de l'information et de la structure de données utilisée pour les représenter.
On souhaite connaître :

- tous les élèves d'une classe :

Si on cherche à accéder à une partie précise de ces informations, le coût dépendra de la position de l'information et de la structure de données utilisée pour les représenter.
On souhaite connaître :

- tous les élèves d'une classe : on parcourt les éléments de `Eleves`, une fois le nom de la classe cherchée trouvée, on en récupère la liste

Si on cherche à accéder à une partie précise de ces informations, le coût dépendra de la position de l'information et de la structure de données utilisée pour les représenter.
On souhaite connaître :

- tous les élèves d'une classe : on parcourt les éléments de `Eleves`, une fois le nom de la classe cherchée trouvée, on en récupère la liste
- tous les élèves dont la LV2 est l'espagnol :

Si on cherche à accéder à une partie précise de ces informations, le coût dépendra de la position de l'information et de la structure de données utilisée pour les représenter.
On souhaite connaître :

- tous les élèves d'une classe : on parcourt les éléments de `Eleves`, une fois le nom de la classe cherchée trouvée, on en récupère la liste
- tous les élèves dont la LV2 est l'espagnol : il faut tout parcourir.

Accès à une base de données

La quasi-totalité systèmes de gestion des bases de données (SGBD) fonctionnent avec SQL (Structured Query Langage). Les requêtes écrites dans ce langage peuvent par exemple s'intégrer au sein d'un programme en PHP . L'utilisateur ne voit pas les requêtes telles qu'elles sont formulées, mais en obtient le résultat.

Accès à une base de données

La quasi-totalité systèmes de gestion des bases de données (SGBD) fonctionnent avec SQL (Structured Query Langage). Les requêtes écrites dans ce langage peuvent par exemple s'intégrer au sein d'un programme en PHP . L'utilisateur ne voit pas les requêtes telles qu'elles sont formulées, mais en obtient le résultat.
En SQL, on peut accéder à des données de la base ; en créer ou en supprimer ; en modifier le contenu ou l'accès.

Table of Contents

Exemple introductif

Terminologie

Mise en pratique : le langage SQL

Terminologie

Définition

[Cadre des bases de données] Soit A un ensemble fini, dont on appelle les éléments des **attributs** (ou colonnes). Soit D un ensemble et soit dom une application de A dans les sous-ensembles de D.
Le **domaine** d'un attribut $A \in \mathrm{A}$ est l'ensemble $dom(A) \subset D$.
Un **schéma relationnel** (ou schéma de table) est un n-uplet de la forme $S = (A_1, \ldots, A_n) \in \mathrm{A}^n$, où les A_i sont deux à deux distincts. On notera plus souvent

$$S = ((A_1, dom(A_1)), (A_2, dom(A_2)), \ldots (A_n, dom(A_n)))$$

pour rappeler les domaines des attributs.
Enfin, on appelle **relation** (ou table) associée à un schéma relationnel $(A_1, \ldots, A_n)$ un sous-ensemble de $dom(A_1) \times dom(A_2) \times \cdots \times dom(A_n)$, et **enregistrement** (ou ligne) un élément d'une relation.

Exemple introductif
Terminologie
Mise en pratique : le langage SQL

Terminologie - exemple

Exemple

Dans le cas de l'exemple introductif on a :

- Attributs : A $=\{$ `Classe, Nom, Prénom, LV2` $\}$.
- Domaines :
  - `dom(Classe)={ "PCSI", ..., "MP" }`
  - `dom(Nom)={ Noms des élèves }`
  - `dom(Prénom)={ Prénoms des élèves }`
  - `dom(LV2)={ "Allemand", "Espagnol" }`.
- Schéma relationnel : `(Classe, Nom, Prénom, LV2)`. Remarquons que, si seuls les noms et prénoms nous intéressent, on peut prendre comme schéma relationnel `(Nom, Prénom)`.

Remarquons qu'on peut envisager un ensemble d'attributs plus étoffé : A=$\{$ `Classe, Nom, Prénom, Moyenne, LV2, Interne` $\}$, avec par exemple,
`dom(Moyenne)=[0,20]` et `dom(Interne)={ "Interne", "Externe" }`.

Exemple introductif
Terminologie
Mise en pratique : le langage SQL

Terminologie - exemple

Exemple

Dans le cadre de l'exemple introductif, la relation serait le sous-ensemble :
`{ ("PCSI", "BAUDELAIRE", "Charles", "Espagnol"), ("PCSI", "DESCARTES", "René", "Allemand", ... }` du produit cartésien
`{"PCSI", ..., "MP" } x { tous les noms } x { tous les prénoms } x { "Espagnol", "Allemand" }` .
En général, on note la relation sous forme de table, plus lisible :

Classe	Nom	Prénom	LV2
PCSI	BAUDELAIRE	Charles	Espagnol
PCSI	DESCARTES	René	Allemand
⋮	⋮	⋮	⋮

Clef primaire

Définition

Une **clef primaire** est un attribut (ou un n-uplet d'attributs) permettant d'identifier chaque enregistrement d'une table de manière unique.

Schéma de la base de données

Une base de données peut contenir plusieurs tables non indépendantes entre elles. Ainsi, un attribut dans une des tables peut rediriger vers un attribut dans une seconde table ou vers une clef primaire, permettant d'identifier un enregistrement. Pour pouvoir interroger une base de données complexe, il nous faut connaître la structure de ces tables et la manière dont elles sont liées entre elles. C'est ce qu'apporte le **schéma de la base de donnée**.

Pour aller plus loin

AlbumId	Title	ArtistId
1	ForThose AboutTo Rock We Salute You	1

Pour aller plus loin

AlbumId	Title	ArtistId
1	ForThose AboutTo Rock We Salute You	1

1. Les enregistrements de la table `album` possèdent les attributs :
2. Les types des attributs sont :
3. Les attributs pouvant être nuls sont :
4. Quel(s) attribut(s) ne prennent jamais deux fois la même valeur ?
5. Quel attribut est une clef primaire ?

Pour aller plus loin

AlbumId	Title	ArtistId
1	ForThose AboutTo Rock We Salute You	1

1. Les enregistrements de la table `album` possèdent les attributs :
2. Les types des attributs sont :
3. Les attributs pouvant être nuls sont :
4. Quel(s) attribut(s) ne prennent jamais deux fois la même valeur ?
5. Quel attribut est une clef primaire ?

Un attributs possède un nom, un type, peut ou non prendre pour valeur `Null`, est une clef primaire ou non. Ces caractéristiques sont fixées lors de la création d'une table.

Modèle entité-association

Correspondance

- relation/association
- propriété/attribut
- classe d'entité/type d'entité
- identifiant/clé

Modèle entité-association

Correspondance

- relation/association
- propriété/attribut
- classe d'entité/type d'entité
- identifiant/clé

Définition

Une **entité** est un objet (concret ou abstrait), possédant des caractéristiques et que l'on veut enregistrer. Un **type d'entités** identifie un ensemble d'entités possédant des caractéristiques communes.

Modèle entité-association

Exemple introductif

Terminologie

Mise en pratique : le langage SQL

Correspondance

- relation/association
- propriété/attribut
- classe d'entité/type d'entité
- identifiant/clé

Définition

Une **entité** est un objet (concret ou abstrait), possédant des caractéristiques et que l'on veut enregistrer. Un **type d'entités** identifie un ensemble d'entités possédant des caractéristiques communes.

Élève
Nom : String
Prénom : String
Classe : Int
Date de naissance : Date

Modèle entité-association

Exemple introductif

Terminologie

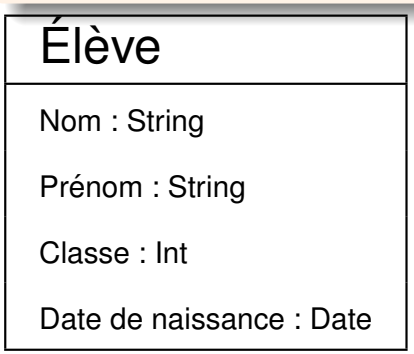
Mise en pratique : le langage SQL

Correspondance

- relation/association
- propriété/attribut
- classe d'entité/type d'entité
- identifiant/clé

Définition

Une **entité** est un objet (concret ou abstrait), possédant des caractéristiques et que l'on veut enregistrer. Un **type d'entités** identifie un ensemble d'entités possédant des caractéristiques communes.



Chaque entité associe une valeur précise à chaque attribut.

Modèle entité-association

Définition

Une **association** relie des entités, dans le respect du l'utilisation prévue des données. Le **type d'associations** est fonction des entités reliées entre elles. La **cardinalité** d'un type d'associations indique le nombre d'occurrences de chaque type d'entités pouvant être intégrés à une association. La cardinalité $x : y$ proche d'un type d'entités E_1, relié à E_2 par une association A, indique qu'une entité de type E_1 est en relation par A avec au minimum x et au maximum y entités de type E_2.

Exemple introductif

Terminologie

Mise en pratique : le langage SQL

Modèle entité-association

Définition

Une **association** relie des entités, dans le respect du l'utilisation prévue des données. Le **type d'associations** est fonction des entités reliées entre elles. La **cardinalité** d'un type d'associations indique le nombre d'occurrences de chaque type d'entités pouvant être intégrés à une association. La cardinalité $x : y$ proche d'un type d'entités E_1, relié à E_2 par une association A, indique qu'une entité de type E_1 est en relation par A avec au minimum x et au maximum y entités de type E_2.

Une association dans laquelle le maximum vaut 1 d'un côté et N de l'autre est une association hiérarchique (ou 1-*) ; lorsque le maximum des deux côtés est N, elle est non-hierarchiques, notée *-*. De la même façon, on note 1-1 le type des associations où le maximum vaut 1 de chaque côté.

Exemple introductif

Terminologie

Mise en pratique : le langage SQL

Exemple introductif

Terminologie

Mise en pratique : le langage SQL

Modèle entité-association

Définition

Une **association** relie des entités, dans le respect du l'utilisation prévue des données. Le **type d'associations** est fonction des entités reliées entre elles. La **cardinalité** d'un type d'associations indique le nombre d'occurrences de chaque type d'entités pouvant être intégrés à une association. La cardinalité $x : y$ proche d'un type d'entités E_1, relié à E_2 par une association A, indique qu'une entité de type E_1 est en relation par A avec au minimum x et au maximum y entités de type E_2.

Une association dans laquelle le maximum vaut 1 d'un côté et N de l'autre est une association hiérarchique (ou 1-*) ; lorsque le maximum des deux côtés est N, elle est non-hierarchiques, notée *-*. De la même façon, on note 1-1 le type des associations où le maximum vaut 1 de chaque côté.

Définition

Une **clef étrangère** est un attribut (ou n-uplet d'attribut) d'une relation renvoyant de manière non équivoque à un (ou plusieurs) attribut(s) d'une autre relation, inclus dans la clef primaire de cette seconde relation.

Modèle entité-association

Pour passer du modèle entité-association au modèle relationnel, on observe les règles suivantes :

- Tout type d'entité est traduit par une relation contenant les mêmes attributs et la clé primaire et la clé du type d'entité.

Modèle entité-association

Pour passer du modèle entité-association au modèle relationnel, on observe les règles suivantes :

- Tout type d'entité est traduit par une relation contenant les mêmes attributs et la clé primaire et la clé du type d'entité.
- Une association 1-1 ou 1-* se traduit par l'ajout de la clef de la seconde entité avecle statut de clef étrangère et l'ajout de tout attribut de l'association comme attribut de la relation issue de la première entité. La clé étrangère se réfère à la clé primaire issue de la seconde entité.

Exemple introductif

Terminologie

Mise en pratique : le langage SQL

Modèle entité-association

Pour passer du modèle entité-association au modèle relationnel, on observe les règles suivantes :

- Tout type d'entité est traduit par une relation contenant les mêmes attributs et la clé primaire et la clé du type d'entité.
- Une association 1-1 ou 1-* se traduit par l'ajout de la clef de la seconde entité avecle statut de clef étrangère et l'ajout de tout attribut de l'association comme attribut de la relation issue de la première entité. La clé étrangère se réfère à la clé primaire issue de la seconde entité.
- Une association *-* peut être traduite par la création d'une relation supplémentaire avec pour attributs les clés des entités associées (et des éventuels attributs de l'association). La clé de l'a relation créée est l'ensemble des attributs représentants les clés des entités associées.

Table of Contents

Exemple introductif

Terminologie

Mise en pratique : le langage SQL

SQL

L'outil de base est l'opération `SELECT`, qui sélectionne une partie d'une table ou relation.
Dans ce cadre, le caractère $*$ signifiera "tous les éléments".

- Union : `SELECT * FROM R1 UNION SELECT * FROM R2`.
- Intersection : `SELECT * FROM R1 INTERSECT SELECT * FROM R2`.
- Différence : `SELECT * FROM R1 EXCEPT SELECT * FROM R2`.
- Projection $\Pi_{\texttt{A1,...,An}}$: `SELECT A1,...,An FROM R1`.
- Sélection $\sigma_{\texttt{A=a}}$: `SELECT * FROM R1 WHERE A=a`.
- Sélection et renommage $\mathrm{p}_{\texttt{A1,...,Am}\leftarrow\texttt{B1,...,BM}}$:
  `SELECT A1 AS B1, A2 AS B2, ..., An AS Bn FROM R1`
- Jointure `R1 [A=B] R2` : `SELECT * FROM R1 JOIN R2 ON A=B`.
- Groupement $\gamma_{(A_1,\ldots,A_m)}(R)$:
  `SELECT * FROM R GROUP BY A1,...,AM`.
- Agrégation $\gamma_{(A_1,\ldots,A_n),f_1(B_1),\ldots f_m(B_m)}$:
  `SELECT A1,...,An,f1(B1),...,fm(Bm) FROM R GROUP BY A1,...,AM`,
  où on peut utiliser les fonctions `COUNT` (pour compter le nombre de lignes), `MAX` (le plus grand), `MIN` (le plus petit), `SUM` (la somme) et `AVG`(la moyenne).

Exercices

- **Sous-requêtes**
  Chercher le titre de l'avant-dernier album de la table album.
  Obtenir le nom de l'artiste concerné (en une requête).

Exercices

- **Sous-requêtes**

  Chercher le titre de l'avant-dernier album de la table album.
  Obtenir le nom de l'artiste concerné (en une requête).

```
SELECT Title FROM
    (SELECT * FROM album
    ORDER BY AlbumId DESC)
LIMIT 1
OFFSET 1
```

Exercices

- **Sous-requêtes**

  Chercher le titre de l'avant-dernier album de la table album. Obtenir le nom de l'artiste concerné (en une requête).

```
SELECT Title FROM
    (SELECT * FROM album
    ORDER BY AlbumId DESC)
LIMIT 1
OFFSET 1
```

```
SELECT name FROM artist
WHERE ArtistId=
    (SELECT ArtistId FROM
        (SELECT * FROM album
        ORDER BY AlbumId DESC)
    LIMIT 13
    OFFSET 1)
```

Exercices

- **Jointures**Obtenir les noms des albums avec le nom des groupes ayant produit ces albums, puis les noms des albums avec le nom de leur genre.

Exercices

- **Jointures**Obtenir les noms des albums avec le nom des groupes ayant produit ces albums, puis les noms des albums avec le nom de leur genre.

```
SELECT Title, Name
FROM album
JOIN Artist ON album.ArtistId=Artist.ArtistId
```

Exercices

- **Jointures**Obtenir les noms des albums avec le nom des groupes ayant produit ces albums, puis les noms des albums avec le nom de leur genre.

```
SELECT Title, Name
FROM album
JOIN Artist ON album.ArtistId=Artist.ArtistId
```

```
SELECT Track.Name, Genre.Name
FROM 'Track'
JOIN 'Genre' ON Track.GenreId = Genre.GenreId
```

Exercices

- Obtenir le nombre d'album par identifiant de groupe.

Exercices

- Obtenir le nombre d'album par identifiant de groupe.

```
SELECT COUNT(*),ArtistId
FROM album
GROUP BY ArtistId
```

SQL

Exercice

On dispose d'une relation `TRIANGLE` définie ainsi : `TRIANGLE (idt:int, ab:int, bc:int, ac:int)` avec les éléments suivants :

idt	ab	bc	ac
1	3	4	5
2	13	5	12
3	15	112	113
4	10	8	6
5	12	16	20
6	21	20	29

1. 1.1 Que renvoie `SELECT COUNT(*) FROM triangles` ?
   1.2 Que renvoie `SELECT * FROM triangles WHERE ab+ac+bc=48` ?
   1.3 Que renvoie
   `SELECT ab*ac*bc FROM triangles WHERE ab+ac+bc>=50` ?
2. Déterminer une expression SQL permettant d'obtenir :
   2.1 la plus petite valeur des produits $AB \times AC \times BC$ pour les triangles ABC de périmètre supérieur ou égal à 40 ;
   2.2 tous les triangles rectangles en A ;
   2.3 le nombre de tels triangles ;
   2.4 le maximum des périmètres des triangles rectangles.