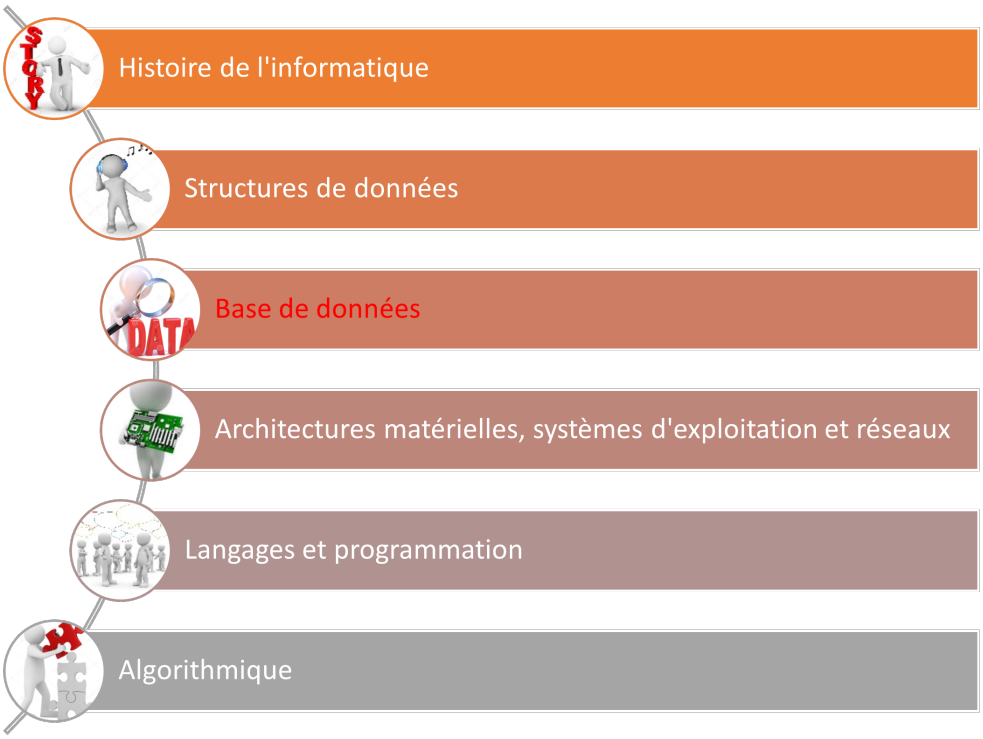

Table des matières

1 Introduction

Directement inspiré du modèle relationnel introduit par E. Codd, le langage SQL permet la définition de relations ou tables dans une base de données relationnelle. Ce langage est standardisé par ISO, sous la référence ISO/IEC 9075. La dernière version du standard date de 2016.

2 Sélection de données

Le langage SQL permet de créer des tables en spécifiant leur nom, leurs attributs, les types de ces derniers et les contraintes associées à la table. Par Exemple pour créer les tables correspondant à la modélisation d'une médiathèque, on peut penser à la modélisation relationnelle de données :

Usager (*nom* : **String**, *prenom* : **String**, *adresse* : **String**, *cp* : **String**, *ville* : **String**, *email* : **String**, *codebarre* : **String**)
Livre (*titre* : **String**, *editeur* : **String**, *annee* : **Int**, *isbn* : **String**)
Auteur (*a_id* : **Int**, *nom* : **String**, *prenom* : **String**)
Auteur_de(*#a_id* : **Int**, *#isbn* : **String**)
Emprunt (*#codebarre* : **String**, *# isbn* : **String**, *retour* : **Date**)

Voici les ordres pour créer ces tables en SQL :

```
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);

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));

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

2.1 Les requêtes

2.1.1 Les clauses `SELECT` et `WHERE`

On souhaite trouver les titres de tous les livres publiés après 1990 dans la base de données de la médiathèque. Une telle requête peut s'écrire en SQL :

```
SELECT titre
FROM livre
WHERE annee>=1990;
```

- `SELECT` : liste des attributs (colonne), si nous indiquons :
  - `*` renvoie "toutes les colonnes"
  - Le mot-clé `DISTINCT` se place juste après `SELECT` et permet d'éliminer les doublons.
- `FROM` : indique la table sur laquelle porte la requête
- `WHERE` : suivi d'une expression booléenne dans laquelle nous pouvons utiliser : <, <=, >, >=, =, <>, +, −, /, ∗, %, `AND`, `OR`, `NOT`, `IN`

Remarque : *La clause* ***`SELECT`*** *permet de restreindre les colonnes de la table renvoyée alors que* ***`WHERE`*** *restreint les lignes.*

Exemples :

- si nous souhaitons afficher les titres de tous les livres publiés par Dargaud entre 1970 et 1980 :

```
SELECT titre
FROM livre
WHERE annee>=1970 AND annee <=1980 AND editeur='Dargaud';
```

- si nous souhaitons afficher les titres de tous les livres publiés par Dargaud, Castermann, Futuropolis :

```
SELECT titre
FROM livre
WHERE editeur IN ('Dargaud', 'Casterman', 'Futuropolis');
```

2.1.2 Trier les données

`ORDER BY champ1, champ2, ... [DESC | ASC]` : permet de trier les données suivant différents champs par ordre croissant `ASC` ou décroissant `DESC`.

Exemples :

- les livres depuis 1990 affichés par ordre alphabétique :

```
SELECT titre
FROM livre
WHERE annee>=1990
ORDER BY titre ASC;
```

- pour renvoyer les plus récents livres en premier et par ordre alphabétique :

```
SELECT titre
FROM livre
WHERE annee>=1990
ORDER BY annee DESC titre ASC;
```

- pour toute requête, il peut arriver que SQL donne plusieurs fois le même résultat car certaines informations sont présentes plusieurs fois dans la table. Le mot-clé `DISTINCT` permet d'éliminer les doublons :

```
SELECT DISTINCT titre
FROM livre
WHERE annee>=1990 ORDER BY titre ASC;
```

2.1.3 L'opérateur `LIKE`

L'opérateur `LIKE` permet de faire des recherches en utilisant des "jokers", c'est-à-dire des caractères qui représentent n'importe quel caractère. Deux jokers existent :

- '%' : qui représente n'importe quelle chaîne de caractères, quelle que soit sa longueur ;
- '_' : qui représente un seul caractère (ou aucun).

Exemples :

- pour chercher les livres dont le titre commence par 'b' :

```
SELECT titre
FROM livre
WHERE titre LIKE 'b%' ;
```

- si nous voulions les titres des livres qui contiennent le mot "Tintin" dans le titre :

```
SELECT titre
FROM livre
WHERE titre LIKE '%Tintin%';
```

- pour chercher les livres dont le titre contient une ou deux lettres dont la première est 'b' (NB : `LIKE` n'est pas sensible à la casse) :

```
SELECT titre
FROM livre
WHERE titre LIKE 'B_' ;
```

2.1.4 Les fonctions d'agrégat

Ces fonctions SQL font des opérations sur plusieurs entrées pour retourner une seule valeur. Voici quelques fonctions d'agrégat représentatives :

- `COUNT` permet d'obtenir le nombre de résultats
- `AVG` permet de calculer la moyenne d'une colonne
- `SUM` permet de calculer la somme de la colonne
- `MIN` et `MAX` renvoie la valeur minimum ou maximum

Exemples :

- pour compter tous les livres de la médiathèque comportant le mot Tintin dans le titre :

```
SELECT COUNT(titre) AS total
FROM livre
WHERE titre LIKE '%Tintin%';
```

  Notons que nous avons choisi de renommer la "colonne" renvoyée, appelée alias, qui n'existe que le temps de la requête en utilisant pour cela le mot-clé `AS`.

- Pour renvoyer la date du plus vieil ouvrage de la médiathèque :

```
SELECT MIN(annee) AS ancien
FROM livre ;
```

  Notez qu'il n'y a pas de clause `WHERE` : toutes les lignes sont évaluées.

2.2 Les jointures

Les requêtes que nous avons vues nous permettent assez facilement de déterminer les livres qui ont été empruntés. Ces derniers sont simplement ceux dont l'ISBN est présent dans la table emprunt.

```
SELECT * FROM emprunt;
```

Cette réponse n'est cependant pas très satisfaisante. En effet, il serait plus naturel de pouvoir afficher les titres de ces livres plutôt que leur ISBN. Le problème est que les titres des livres sont présents uniquement dans la table `livre`. Pour répondre à ce problème nous devons faire une jointure entre les tables. Étant données deux tables A et B, la jointure consiste à créer toutes combinaisons de lignes de A et de B ayant un attribut de même valeur. Ici, nous souhaitons obtenir une "grande table" dont les colonnes sont celles de la table `emprunt` et celle de la table `livre`, en réunissant les lignes ayant le même `isbn`. Cela peut être fait au moyen de la directive `JOIN`.

livre	titre	editeur	annee	isbn
	...	...	...	...

emprunt	codebarre	isbn	date
	...	...	...

Jointure livre-emprunt	titre	editeur	annee	isbn	codebarre	date
	...	...	...	...	...	...

```
SELECT *
FROM emprunt
JOIN livre ON emprunt.isbn=livre.isbn;
```

Nous pouvons également combiner avec les clauses `SELECT` et `WHERE`. Par exemple, si on souhaite afficher uniquement les titres et les dates des livres empruntés qui sont à rendre avant le 1 février 2022, on peut écrire la requête suivante :

```
SELECT livre.titre, emprunt.retour
FROM emprunt
JOIN livre ON emprunt.isbn=livre.isbn
WHERE emprunt.retour<'2022-02-01';
```

Cependant avec les nouveaux SGBD il est possible de faire cette même requête de la manière suivante :

```
SELECT livre.titre, emprunt.retour
FROM emprunt, livre
WHERE emprunt.isbn=livre.isbn AND emprunt.retour<'2022-02-01';
```

De plus nous ne sommes pas limités à une seule jointure. Si on souhaite afficher les noms et prénoms des utilisateurs ayant emprunté ces livres, il suffit de joindre la table `usager`, en rajoutant une nouvelle clause `JOIN ON`, cette fois sur le `code_barre` de l'usager.

```
SELECT livre.titre, emprunt.retour, usager.nom, usager.prenom
FROM emprunt
JOIN livre ON emprunt.isbn=livre.isbn
JOIN usager ON usager.code_barre = emprunt.code_barre
WHERE emprunt.retour<'2022-02-01';
```

3 Modification de données

3.1 Suppression de lignes

L'ordre `DELETE FROM t WHERE c` permet de supprimer de la table `t` toutes les lignes vérifiant la condition `c`.

Dans l'exemple de notre médiathèque, supposons que l'utilisateur Sébastien Petit, dont le `code_barre` est '934701281931582', ait rendu ses livres. Il faut supprimer de la table `emprunt` toutes les lignes pour lesquelles le `code_barre` vaut '934701281931582', ce qui donne l'ordre suivant :

```
DELETE FROM emprunt
WHERE code_barre='934701281931582';
```

3.2 Mise à jour

Elle consiste à remplacer certains attributs d'un ensemble de lignes par de nouvelles valeurs.

```
UPDATE t SET a1=e1,...,SET an=en WHERE c;
```

Cette dernière signifie "sélectionne dans la table `t` toutes les lignes vérifiant la condition `c` et, pour chacune de ces lignes, remplace la valeur courante de l'attribut a_i par la valeur de l'expression e_i". Par exemple, si l'utilisateur Sébastien Petit souhaite mettre à jour son adresse `email`, on écrit ceci :

```
UPDATE utilisateur
SET email='spetit@hmail.com'
WHERE code_barre='934701281931582';
```

Les expressions de mise à jour peuvent mentionner des noms d'attributs. Ces dernier sont alors remplacés par la valeur courante (avant mise à jour) de ces attributs. Supposons par exemple que la médiathèque soit fermée au mois d'avril. On souhaite que tous les emprunts dont la date de rendu était en avril soient prolongés de 30 jours :

```
UPDATE emprunt
SET retour=retour+30
WHERE retour>='2022-04-01';
```

3.3 Copie de table

Toute modification qui ne viole pas de contrainte est définitive (risque de perte de données). Les bonnes pratiques recommandent l'utilisation de plusieurs "copies" de la BDD. Une copie de test sera utilisée pour le développement. Comme nous l'avons vu l'ordre `SELECT...FROM` renvoie une table comme résultat. Il est possible de nommer cette dernière grâce au mot clé `INTO` (attention cette instruction ne copie pas les contraintes) :

```
SELECT * INTO usager_paris FROM usager WHERE cp LIKE '75%';
```

Cet ordre créé une nouvelle table nommée `usager_paris` contenant le résultat de la requête. Cette dernière a le même schéma que la table `usager` car toutes les colonnes ont été copiées. En revanche, elle ne contiendra que les lignes pour lesquelles le code postal commence par 75. La table `usager_paris` est ensuite utilisable comme n'importe quelle autre table. La clause `WHERE` permet de choisir les lignes à conserver.
Créer une copie conforme d'une table peut se faire en utilisant deux variations sur des opérations que nous connaissons bien. La première est l'opération `CREATE`, utilisée comme ceci :

```
CREATE TABLE usager3 (LIKE usager INCLUDING ALL);
INSERT INTO usager3 (SELECT * FROM usager);
```

La première commande permet de recréer toute la table avec les contraintes.
La deuxième commande `INSERT` à la table toutes les lignes de la requête mise entre parenthèses.

3.4 Requêtes imbriquées

Il est possible de créer une table de manière temporaire et d'exécuter une requête sur cette table en imbriquant la première requête dans la clause `FROM` de la seconde ou dans une clause `JOIN...ON`.

```
SELECT * FROM (SELECT * FROM livre WHERE annee >=1990) AS tmp WHERE tmp.annee<=2000;
```

La requête ci-dessus calcule d'abord une table intermédiaire nommée `tmp` qui liste les livres publiés après 1990. Suite à quoi, table `tmp` est refiltrée pour ne garder que les livres pour lesquels l'année est inférieure à 2000.

4 Exercices

Exercice n° 1. (requêtes simples, sans jointure ni imbrication)

Soit la base de données de la médiathèque. Donner le code SQL de chacune des requêtes ci-dessous :

1. Tous les titres de livre.
2. Tous les noms d'usagers.
3. Tous les noms d'usager en retirant les doublons.
4. Les titres des livres publiés avant 1980.
5. Les titres des livres dont le titre contient la lettre "A".
6. Les ISBN des livres à rendre pour le 01/02/2022.
7. Les noms d'auteurs triés par ordre alphabétique.
8. Les noms d'usagers vivant dans le 12^e ou 13^e arrondissement de Paris (codes postaux 75012 et 75013).
9. Les noms et adresses des usagers n'habitant pas dans une rue. La chaine "Rue" ne doit pas apparaitre dans l'adresse).
10. Les années et titres des livres publiés lors d'une année bissextile. On rappelle que ce sont les années divisibles par 4, mais pas celles divisibles par 100 sauf si elles sont divisibles par 400.

Exercice n° 2. (requêtes avancées avec jointure ou imbrication)

Soit la base de données de la médiathèque. Donner le code SQL de chacune des requêtes ci-dessous :

1. Le titre des livres empruntés.
2. Le titre des livres empruntés à rendre avant le 31/03/2022.
3. Le nom et le prénom de l'auteur du livre '1984'.
4. Le nom et le prénom des usagers ayant emprunté des livres, sans doublons.
5. Même requête que précédemment, avec les noms triés par ordre alphabétique.
6. Les titres des livres publiés strictement avant 'Dune'.
7. Les noms et prénoms des auteurs des livres trouvés à la question précédente.
8. Comme la question précédente, en retirant les doublons.
9. Le nombre de résultats trouvés à la question précédente.

Exercice n° 3.

Soit la base de données de la médiathèque. Formuler simplement en français les requêtes SQL suivantes :

```sql
SELECT * FROM livre WHERE titre LIKE '%Robot%';

SELECT nom,prenom FROM usager WHERE ville='Guingamp';

SELECT u.nom, u.prenom
FROM usager AS u
JOIN emprunt AS e ON u.code_barre=e.code_barre
WHERE retour < '2022-04-02';

SELECT l.titre
FROM livre AS l
WHERE l.isbn IN (SELECT isbn FROM livre WHERE annee >1990);
```

Exercice n° 4.

On considère les trois tables suivantes :

```
CREATE TABLE x (a INT PRIMARY KEY, b INT, CHECK (b>=0));
CREATE TABLE y (c INT PRIMARY KEY, d INT, CHECK (d<=30));
CREATE TABLE z (a INT REFERENCES X(a), c INT REFERENCES Y(c), e INT, UNIQUE (a,c));
```

x :

a	b
1	1
2	2
3	2
4	2
5	1
6	9
7	1

y :

c	d
9	9
10	10
11	9
12	20
13	20
14	9
15	1
16	10
17	10

z :

a	c	e
1	11	30
2	14	9
5	15	1
7	17	3
1	10	50
2	9	8
2	15	15
3	17	19
4	16	12
5	10	20
2	11	30
7	14	9
7	9	12

Pour chacune des requêtes SQL ci-dessous, calculer son résultat (à la main)

```
SELECT * FROM x WHERE b>3;

SELECT DISTINCT e FROM z WHERE e>10 AND e<50;

SELECT * FROM y WHERE c%2=0 ORDER BY d ASC;

SELECT x.a,x.b FROM x JOIN z ON z.a=x.a WHERE z.e<9;

SELECT DISTINCT x.b,y.d FROM x JOIN z ON z.a=x.a JOIN y ON y.c=z.c;
```

Exercice n° 5.

On considère les trois tables décrites de l'exercice précédent. Pour chacune des modification ci-dessous, indiquer si elle réussit ou si elle échoue. Si elle réussit, indiquer comment la table est modifiée. Si elle échoue, expliquer pourquoi. Les questions sont indépendantes, c'est à dire que chacune repart des tables de base entre chaque question.

```
UPDATE x SET b=b+a;

UPDATE x SET b=b-2;

INSERT INTO z VALUES (1,17,1);

INSERT INTO z VALUES (1,18,7);

INSERT INTO z VALUES (1,10,7);

DELETE FROM y WHERE c>=12 AND c<=13;

DELETE FROM y WHERE c>=12 AND c<=14;

INSERT INTO y VALUES (40,20);

INSERT INTO y VALUES (20,40);

DELETE FROM z WHERE a%2=0 OR c%2=0 OR e%2=0;
```

Exercice n° 6.

Dans cet exercice, nous allons utiliser une base de données tournoi, contenant deux tables Joueur et Partie, définie comme ceci :

joueur
idJoueur
nomJoueur
prenomJoueur

partie
idPartie
idJoueur1
idJoueur2
score1
score2
idVainqueur

Les clés `idJoueur1`, `idJoueur2` et `idVainqueur` sont des clés étrangères référençant `idJoueur` de la table Joueur.
Avec les tables :

joueur	idJoueur	nomJoueur	prenomJoueur
	1	Lagaffe	Gaston
	2	Talon	Achille
	3	Bioskop	Jill
	4	Blanc-sec	Adele
	5	Maltese	Corto

partie	idPartie	# idJoueur1	# idJoueur2	score1	score2	# idVainqueur
	1	1	2	54	45	2
	2	1	3	45	85	3
	3	4	5	96	35	4
	4	2	5	54	8	2
	5	2	3	9	12	3
	6	2	4	56	84	2
	7	1	4	9	23	4
	8	1	5	15	35	5
	9	3	5	6	12	5
	10	3	4	8	7	3

Vous remarquerez qu'il y a des erreurs d'informations concernant les vainqueurs. Nous allons voir à la question 3 comment modifier ces erreurs.

1. (a) Je cherche à connaître les identifiants des parties gagnées par Jill Bioskop (`idJoueur = 3`). Quelle requête SQL faut-il taper ?
   (b) Modifier votre requête de façon à ce que soit affiché le nom du gagnant et non pas son identifiant.
2. (a) Que produit le requête suivante ?

```sql
SELECT score2-score1 FROM partie
```

   (b) Quelle requête doit-on taper pour obtenir :

idPartie	?
1	9
2	40
3	61
4	46
5	3
6	28
7	14
8	20
9	6
10	1

   (c) Quelle requête doit-on taper pour obtenir le nombre de parties gagnées par Achille Talon ?
3. Proposer une modification de la table pour corriger ces erreurs.
4. Insérer le nouveau joueur avec un `idJoueur = 7` de nom 'Bros' et de prénom 'Mario'.
5. Vérifier la présence par une requête de votre nouveau joueur.
6. Supprimer ce nouveau joueur.

Exercice n° 7.

Un club de handball souhaite regrouper efficacement toutes ses informations. Il utilise pour cela des bases de données relationnelles afin d'avoir accès aux informations classiques sur les licenciés du club ainsi que sur les matchs du championnat. Le langage SQL a été retenu.
On suppose dans l'exercice que tous les joueurs d'une équipe jouent à chaque match de l'équipe.
La structure de la base de données est composée des deux tables (ou relations) suivantes :

Table `licencies`	
Attributs	**Types**
`id_licencie`	INT
`prenom`	VARCHAR
`nom`	VARCHAR
`annee_naissance`	INT
`equipe`	VARCHAR

Table `matchs`	
Attributs	**Types**
`id_matchs`	INT
`equipe`	VARCHAR
`adversaire`	VARCHAR
`lieu`	VARCHAR
`date`	DATE

Ci-dessous un exemple de ce que l'on peut trouver dans la base de données :
Exemple **non exhaustif** d'entrées de la table `licencies`

id_licencie	prenom	nom	annee_naissance	equipe
63	Jean-Pierre	Masclef	1965	Vétérans
102	Eva	Cujon	1992	Femmes 1
125	Emile	Alinio	2000	Hommes 2
247	Ulysse	Trentain	2008	-12 ans

Exemple **non exhaustif** d'entrées de la table `matchs`

id_match	equipe	adversaire	lieu	date
746	-16 ans	PHC	Domicile	2021-06-19
780	Vétérans	PHC	Exterieur	2021-06-26
936	Hommes 3	LSC	Exterieur	2021-06-20
1032	-19 ans	LOH	Exterieur	2021-05-22
1485	Femmes 2	CHM	Domicile	2021-05-02
1512	Vétérans	ATC	Domicile	2021-04-12

1. (a) L'attribut `nom` de la table `licencies` pourrait-il servir de clé primaire ? Justifier.
   (b) Citer un autre attribut de cette table qui pourrait servir de clé primaire.
2. (a) Expliquer ce que renvoie la requête SQL suivante :
   `SELECT prenom, nom FROM licencies WHERE equipe = "-12ans"`
   (b) Que renvoie la requête précédente si `prenom, nom` est remplacé par une étoile (`*`) ?
   (c) Écrire la requête qui permet l'affichage des dates de tous les matchs joués à domicile de l'équipe `Vétérans`.
3. Écrire la requête qui permet d'inscrire dans la table `licencies`, `Jean Lavenu` né en 2001 de l'équipe `Hommes 2` et qui aura comme numéro de licence 287 dans ce club.
4. On souhaite mettre à jour les données de la table `licencies` du joueur `Joseph Cuviller`, déjà inscrit. Il était en équipe `Hommes 2` et il est maintenant en équipe `Vétérans`. Afin de modifier la table dans ce sens, proposer la requête adéquate.
5. Pour obtenir le nom de tous les licenciés qui jouent contre le `LSC` le 19 juin 2021, recopier et compléter la requête suivante :

```
SELECT nom FROM licencies
JOIN Matchs ON licencies.equipe = matchs.equipe
WHERE ..................................... ;
```