Le langage SQL

Langage et sécurité des bases de données S5
L3 MIAGE Classique

Thibault Anani Agondja
Inspiré du cours de Sonia Guehis

Nanterre Université
Année 2021-2022

Table des matières

Table des matières

SQL (Structured Query language)

Définition

Langage de définition, de manipulation et de contrôle d'accès aux données développé par IBM

Langage de définiton des données [LDD]

- Description de la structure des données
- Définition du schéma de la base
- Typage des colonnes, contraintes

Langage de manipulation des données [LMD]

- Permet la manipulation des données
- Lecture et mise à jour
- Ajout, modification, suppression

Système de gestion des bases de données

Définition

Logiciel de haut niveau d'abstraction qui permet de manipuler les informations présentent dans une base de données

Interpréteur SQL

Chaque SGBD va posséder son propre interpréteur SQL différent propre, plus ou moins proche de la norme SQL

- MySQL
- PostgreSQL
- SQLite
- Microsoft SQL Server
- Oracle

Existence de SGBD NoSQL

Redis, Cassandra, MongoDB, ...

Les types d'opérations

4 types d'opérations

- création (ou insertion)
- modification (ou mise à jour)
- destruction (ou suppression)
- recherche (ou requêtes)

Ces opérations correspondent à des commandes du LMD et du LDD. La plus complexe est la recherche en raison de la variété des critères

Exemples d'opérations : création

Insérer des informations concernant un employé nommé Jean

Nom	Prénom	Date de Naissance	Adresse	Département	Statut	Salaire
Didier	Denis	21/01/1993	60 rue Chopin	Logistique	Cadre	56 000
Beauchamp	Patrice	07/04/1961	54 rue Ernest	Comptabilité	Employé	40 000
Cantina	Clémentine	12/08/1976	51 rue l'Epeule	Comptabilité	Cadre	49 000
Delarue	Bruno	25/07/1978	9 rue Orange	Marketing	Employé	39 000

Employé

⇓

Nom	Prénom	Date de Naissance	Adresse	Département	Statut	Salaire
Didier	Denis	21/01/1993	60 rue Chopin	Logistique	Cadre	56 000
Beauchamp	Patrice	07/04/1961	54 rue Ernest	Comptabilité	Employé	40 000
Cantina	Clémentine	12/08/1976	51 rue l'Epeule	Comptabilité	Cadre	49 000
Delarue	Bruno	25/07/1978	9 rue Orange	Marketing	Employé	39 000
Beaudoin	Jean	12/11/1985	43 rue Michel	Logistique	Cadre	56 000

Employé

Exemples d'opérations : modification

Augmenter le salaire de Jean de 10%

Nom	Prénom	Date de Naissance	Adresse	Département	Statut	Salaire
Didier	Denis	21/01/1993	60 rue Chopin	Logistique	Cadre	56 000
Beauchamp	Patrice	07/04/1961	54 rue Ernest	Comptabilité	Employé	40 000
Cantina	Clémentine	12/08/1976	51 rue l'Epeule	Comptabilité	Cadre	49 000
Delarue	Bruno	25/07/1978	9 rue Orange	Marketing	Employé	39 000
Beaudoin	Jean	12/11/1985	43 rue Michel	Logistique	Cadre	56 000

Employé

Nom	Prénom	Date de Naissance	Adresse	Département	Statut	Salaire
Didier	Denis	21/01/1993	60 rue Chopin	Logistique	Cadre	56 000
Beauchamp	Patrice	07/04/1961	54 rue Ernest	Comptabilité	Employé	40 000
Cantina	Clémentine	12/08/1976	51 rue l'Epeule	Comptabilité	Cadre	49 000
Delarue	Bruno	25/07/1978	9 rue Orange	Marketing	Employé	39 000
Beaudoin	Jean	12/11/1985	43 rue Michel	Logistique	Cadre	61 600

Employé

Exemples d'opérations : destruction

Retirer les informations concernant Jean

Nom	Prénom	Date de Naissance	Adresse	Département	Statut	Salaire
Didier	Denis	21/01/1993	60 rue Chopin	Logistique	Cadre	56 000
Beauchamp	Patrice	07/04/1961	54 rue Ernest	Comptabilité	Employé	40 000
Cantina	Clémentine	12/08/1976	51 rue l'Epeule	Comptabilité	Cadre	49 000
Delarue	Bruno	25/07/1978	9 rue Orange	Marketing	Employé	39 000
Beaudoin	Jean	12/11/1985	43 rue Michel	Logistique	Cadre	61 600

Employé

Nom	Prénom	Date de Naissance	Adresse	Département	Statut	Salaire
Didier	Denis	21/01/1993	60 rue Chopin	Logistique	Cadre	56 000
Beauchamp	Patrice	07/04/1961	54 rue Ernest	Comptabilité	Employé	40 000
Cantina	Clémentine	12/08/1976	51 rue l'Epeule	Comptabilité	Cadre	49 000
Delarue	Bruno	25/07/1978	9 rue Orange	Marketing	Employé	39 000

Employé

Exemples d'opérations : recherche

Chercher les employés cadres

Nom	Prénom	Date de Naissance	Adresse	Département	Statut	Salaire
Didier	Denis	21/01/1993	60 rue Chopin	Logistique	Cadre	56 000
Beauchamp	Patrice	07/04/1961	54 rue Ernest	Comptabilité	Employé	40 000
Cantina	Clémentine	12/08/1976	51 rue l'Epeule	Comptabilité	Cadre	49 000
Delarue	Bruno	25/07/1978	9 rue Orange	Marketing	Employé	39 000

Employé

Nom	Prénom	Date de Naissance	Adresse	Département	Statut	Salaire
Didier	Denis	21/01/1993	60 rue Chopin	Logistique	Cadre	56 000
Cantina	Clémentine	12/08/1976	51 rue l'Epeule	Comptabilité	Cadre	49 000

Employé

Langages de requêtes relationnelles

Pouvoir d'expression

Représente ce qu'il est possible de calculer et les opérations qu'il est possible de faire

Algèbre relationnelle

- Langage procédural : langage qui décrit explicitement comment trouver le résultat en une suite d'instructions
- Langage de bas niveau difficile à manipuler proche des langages de programmation
- Notation algébrique

Calcul relationnel

- Langage déclaratif : langage qui décrit les propriétés que devra avoir le résultat plutôt que les procédures
- Langage de haut niveau facile d'accès proche du langage naturel
- Notation logique

Les deux langages possèdent le même pouvoir d'expression

Langages de requêtes relationnelles

SQL

- Langage déclaratif
- Langage de haut niveau proche du langage naturel
- Notation logique
- Inspiré à la fois de l'algèbre relationnelle et du calcul relationnel

Le langage SQL possède un pouvoir d'expression plus important que l'algèbre relationnelle et le calcul relationnel

Table des matières

Schéma et instance : exemples

Parenté(Parent, Enfant)
Descriptif(Parent, Age, Sexe, Ville)
Scolarité(Enfant, Ecole)

Parent	**Enfant**
Pascal	Marie
Pascal	Leo
Raymond	Zoe
Clara	Zoe
Marcel	Raymond

Parenté

Personne	**Age**	**Sexe**	**Ville**
Pascal	40	M	Paris
Marie	20	F	Paris
Leo	18	M	Paris
Zoe	2	F	Nice
Clara	27	F	Nice
Marcel	60	M	Marseille
Raymond	40	M	Nice
Johnny	65	M	Lyon

Descriptif

Enfant	**Ecole**
Zoe	A
Marie	B
Leo	A

Scolarité

La clause SELECT

Définition

La clause SELECT permet d'établir une projection qui va définir une relation restreinte à un sous-ensemble des attributs de *Exp* en extrayant les valeurs des attributs spécifiés

$$SELECT\ Attr_1, Attr_2, Attr_3, \ldots\ FROM\ Exp$$

Où $Attr_1, Attr_2, Attr_3$ sont des attributs de l'expression relationnelle de *Exp*

Le résultat de $SELECT\ Attr_1, Attr_2, Attr_3\ FROM\ Exp$ contient les mêmes n-uplets que *Exp*, tronqués des attributs ne figurant pas dans la liste de projection

La clause SELECT

Requête

Liste des parents de la base

SQL

SELECT parent FROM Parenté

Parent
Pascal
Pascal
Raymond
Clara
Marcel

Remarque

Les doublons ne sont pas supprimés automatiquement

La clause SELECT

La clause DISTINCT

Enlève les doublons des résultats

SQL

SELECT DISTINCT parent FROM Parenté

Parent
Pascal
Raymond
Clara
Marcel

Remarque

Le doublon du n-uplet Pascal a été supprimé

La clause SELECT

Requête

Les informations sur les personnes de la base

SQL

*SELECT * FROM Descriptif*

Personne	**Age**	**Sexe**	**Ville**
Pascal	40	M	Paris
Marie	20	F	Paris
Leo	18	M	Paris
Zoe	2	F	Nice
Clara	27	F	Nice
Marcel	60	M	Marseille
Raymond	40	M	Nice
Johnny	65	M	Lyon

Remarque

Possibilité de sélectionner tous les attributs avec l'opérateur $*$

La clause WHERE

Définition

La clause WHERE permet d'établir une sélection qui va définir une relation qui ne contient que les n-uplets de *Exp* qui vérifient la condition spécifiée ou aussi appelé prédicat

*SELECT * FROM Exp WHERE F*

Où F est une formule logique de premier ordre formée de :

- Constantes
- Attributs figurant dans *Exp*
- Comparateurs : $=, <, >, !=, \leq, \geq$
- Connecteurs logiques : *OR* (ou), *AND* (et), *NOT* (non)

Le résultat de *SELECT * FROM Exp WHERE F* contient tous les n-uplets de *Exp* tels que F est vraie

La clause WHERE

Requête

Liste des personnes de sexe féminin

SQL

*SELECT * FROM Descriptif WHERE sexe = "F"*

Personne	**Age**	**Sexe**	**Ville**
Marie	20	F	Paris
Zoe	2	F	Nice
Clara	27	F	Nice

La clause WHERE

Requête

Les personnes de plus de 40 ans

SQL

*SELECT * FROM Descriptif WHERE age > 40*

Personne	**Age**	**Sexe**	**Ville**
Marcel	60	M	Marseille
Johnny	65	M	Lyon

La clause WHERE

L'opérateur BETWEEN

Sélectionner un intervalle de n-uplets dans une requête utilisant WHERE. Les données peuvent être des nombres, des chaînes de caractères ou des dates.

Requête

Les personnes qui ont entre 18 et 40 ans

SQL

SELECT * *FROM Descriptif WHERE age BETWEEN* 18 *AND* 40

Personne	**Age**	**Sexe**	**Ville**
Pascal	40	M	Paris
Marie	20	F	Paris
Leo	18	M	Paris
Clara	27	F	Nice
Raymond	40	M	Nice

La clause WHERE

L'opérateur IS

Permet de filtrer les n-uplets qui contiennent des valeurs manquantes (NULL) qui ne peuvent pas être filtrés avec les autres opérateurs ($<$, $>$, $!=$, ...)

Personne	Age	Sexe	Ville
NULL	40	M	Paris
Marie	20	F	Paris
NULL	18	M	Paris
Zoe	2	F	Nice

Descriptif

IS NULL

*SELECT * FROM Descriptif WHERE personne IS NULL*

Donne tous les n-uplets qui ont aucune valeur dans l'attribut personne

IS NOT NULL

*SELECT * FROM Descriptif WHERE personne IS NOT NULL*

Donne tous les n-uplets qui ont une valeur dans l'attribut personne

La clause WHERE

L'opérateur IN

Vérifie si une colonne est égale à une des valeurs comprise dans un ensemble de valeurs déterminés.

Requête

Les informations sur Pascal, Marie, Leo et Zoe

SQL

*SELECT * FROM Descriptif WHERE personne IN ("Pascal", "Marie", "Leo", "Zoe")*

Personne	**Age**	**Sexe**	**Ville**
Pascal	40	M	Paris
Marie	20	F	Paris
Leo	18	M	Paris
Zoe	2	F	Nice

La clause WHERE

L'opérateur LIKE

Recherche les n-uplets d'une colonne qui commence par un caractère ou une chaîne de caractères spécifiques

- *a%* recherche les chaîne de caractères qui commencent par 'a'
- *%bob* recherche les chaîne de caractères qui se terminent par 'bob'
- *%c%* recherche les chaîne de caractères qui contiennent la lettre 'c'
- *_d* recherche les chaîne de caractères qui contiennent comme deuxième lettre la letttre 'd'

Requête

Les personnes dont l'avant-dernière lettre du nom est un 'n'

SQL

SELECT personne FROM Descriptif WHERE personne LIKE "%n_"

Personne
Raymond
Johnny

La clause EXISTS

Défintion

Vérifie s'il existe oui ou non au moins un n-uplet dans une sous-requête. La requête principale s'exécutera uniquement si la sous-requête retourne au moins un n-uplet

Requête

Les parents des enfants scolarisés

SQL

SELECT DISTINCT p.Parent FROM Parenté p WHERE EXISTS(SELECT s.Enfant FROM Scolarité s WHERE s.Enfant = p.Enfant)

Parent
Pascal
Raymond
Clara

Exercices : Sélection et Projection

Comment écrire ces requêtes en SQL ?

1. Les personnes qui habitent à Paris ou Nice
2. Les personnes qui n'habitent pas à Nice
3. La ville où habite Raymond
4. L'âge de Marcel
5. Les personnes qui habitent à Paris et qui ont plus de 18 ans

Exercices : Sélection et Projection

Les personnes qui habitent à Paris ou Nice

*SELECT * FROM Descriptif WHERE ville = "Paris" OR ville = "Nice"*

Les personnes qui n'habitent pas à Nice

*SELECT * FROM Descriptif WHERE ville != "Nice"*

La ville où habite Raymond

SELECT ville FROM Descriptif WHERE personne = "Raymond"

L'âge de Marcel

SELECT age FROM Descriptif WHERE personne = "Marcel"

Les personnes qui habitent à Paris et qui ont plus de 18 ans

*SELECT * FROM Descriptif WHERE ville = "Paris" AND age = 18*

Les opérateurs ensemblistes

L'Union

*SELECT * FROM Exp_1 UNION SELECT * FROM Exp_2*

Permet d'obtenir les n-uplets à la fois soit dans Exp_1 soit dans Exp_2 soit les deux

La Différence

*SELECT * FROM Exp_1 MINUS SELECT * FROM Exp_2*

Permet d'obtenir les n-uplets qui existent dans la relation Exp_1 et non dans la relation Exp_2

L'Intersection

*SELECT * FROM Exp_1 INTERSECT SELECT * FROM Exp_2*

Permet d'obtenir les n-uplets qui existent dans la relation Exp_1 et à la fois dans la relation Exp_2

L'Union

Requête

Les enfants de Raymond ou de Pascal

SQL

SELECT enfant FROM Parenté WHERE parent = "Raymond"
UNION
SELECT enfant FROM Parenté WHERE parent = "Pascal"

Enfant
Zoe

$\cup$

Enfant
Marie
Leo

$\Rightarrow$

Enfant
Zoe
Marie
Leo

La Différence

Requête

Les enfants non scolarisés

SQL

SELECT enfant FROM Parenté
MINUS
SELECT enfant FROM Scolarité

Enfant
Marie
Leo
Zoe
Raymond

$-$

Enfant
Zoe
Marie
Leo

$\Rightarrow$

Enfant
Raymond

L'Intersection

Requête

Les enfants de Raymond et Clara

SQL

SELECT enfant FROM Parenté WHERE parent = "Raymond"
INTERSECT
SELECT enfant FROM Parenté WHERE parent = "Clara"

Enfant
Zoe

$\cap$

Enfant
Zoe

$\Rightarrow$

Enfant
Zoe

L'Intersection

Requête

Les personnes de sexe féminin de plus de 20 ans

SQL

SELECT personne FROM Dscriptif WHERE age $>$ 20
INTERSECT
SELECT personne FROM Descriptif WHERE sexe $=$ *"F"*

Personne
Pascal
Clara
Marcel
Raymond
Johnny

$\cap$

Personne
Marie
Zoe
Clara

$\Rightarrow$

Personne
Clara

Exercices : Les opérateurs ensemblistes

Comment écrire ces requêtes en SQL avec les opérateurs ensemblistes ?

1. Les parents de Marie ou de Raymond
2. Les personnes qui ne sont ni parisiens ni marseillais
3. Les personnes de sexe masculin qui habitent à Paris
4. Les personnes de sexe féminin qui habitent Nice qui ont 20 ans ou plus
5. Les Parisiens de moins de 40 ans

Exercices : Les opérateurs ensemblistes

Les parents de Marie ou de Raymond

SELECT parent FROM Parenté WHERE enfant = "Marie"
UNION
SELECT parent FROM Parenté WHERE enfant = "Raymond"

Les personnes qui ne sont ni parisiens ni marseillais

*SELECT * FROM Descriptif*
MINUS
*SELECT * FROM Descriptif WHERE ville = "Paris" OR ville = "Marseille"*

Les personnes de sexe masculin qui habitent à Paris

*SELECT * FROM Descriptif WHERE sexe = "M"*
INTERSECT
*SELECT * FROM Descriptif WHERE ville = "Paris"*

Exercices : Les opérateurs ensemblistes

Les personnes de sexe féminin qui habitent Nice qui ont 20 ans ou plus

*SELECT * FROM Descriptif WHERE sexe = "F"*
INTERSECT
*SELECT * FROM Descriptif WHERE ville = "Nice"*
INTERSECT
*SELECT * FROM Descriptif WHERE age*20

Les Parisiens de moins de 40 ans

*SELECT * FROM Descriptif WHERE ville = "Paris"*
INTERSECT
*SELECT * FROM Descriptif WHERE age* < 40

La Jointure

Définition

Définit une relation qui contient les n-uplets qui vérifient le prédicat F du produit cartésien de Exp_1 et Exp_2. Elle permet de combiner une paire de n-uplets de deux relations différentes en un seul n-uplet

Pour exprimer une jointure il faut :

- Dans la clause FROM indiquer toutes les tables nécessaires pour la jointure
- Dans la clause WHERE les conditions de jointure

La Jointure

Requête

Liste des parents et de l'école de leurs enfants

SQL

SELECT D.parent, S.ecole FROM Descriptif D, Scolarité S
WHERE D.enfant = S.enfant

Parent	**Enfant**	**Ecole**
Pascal	Marie	B
Pascal	Leo	A
Raymond	Zoe	A
Clara	Zoe	A

$\Rightarrow$

Parent	**Ecole**
Pascal	B
Pascal	A
Clara	A
Raymond	A

$Parenté \bowtie Scolarité$

Exercices : Jointure

Soient les relations suivantes

Personne(<u>CIN</u>, Nom, Prenom, Adresse)
Voiture(NCarteGrise, <u>CIN</u>, Modele)
Moto(NCarteGrise, <u>CIN</u>, Modele)

Comment écrire ces requêtes SQL ?

1. Le modèle des voitures au nom de Cristophe Martin
2. Le nom des personnes qui possèdent une voiture mais pas de moto
3. Le prénom des personnes qui possèdent une voiture et une moto
4. L'adresse des personnes qui ne possèdent ni voiture ni moto

Exercices : Jointure

Le modèle des voitures au nom de Cristophe Martin

SELECT v.model FROM Voiture v Personne p
WHERE v.cin = p.cin and p.nom = "Martin" and p.prenom = "Cristophe"

Le nom des personnes qui possèdent une voiture mais pas de moto

SELECT p.nom FROM Voiture v Personne p Moto m
WHERE v.cin = p.cin and v.cin ! = m.cin

Le prénom des personnes qui possèdent une voiture et une moto

SELECT p.nom FROM Voiture v Personne p Moto m
WHERE v.cin = p.cin and v.cin = m.cin

L'adresse des personnes qui ne possèdent ni voiture ni moto

SELECT p.adresse FROM Voiture v Personne p Moto m
WHERE v.cin ! = p.cin and v.cin ! = m.cin

Le Renommage

Définition

Permet de changer le nom d'un ou plusieurs attributs d'une relation

Requête

Les grands parents de Zoé

SELECT t.parent as GPdeZoé FROM Parenté t Parenté s
WHERE s.enfant = "Zoé" and t.enfant = s.parent

GPdeZoé
Marcel

La Division

Définition

Produit une relation Exp_3 qui comporte les attributs appartenant à Exp_1 mais n'appartenant pas à Exp_2 donnent toujours un n-uplet de Exp_1

Requête

Les parents scolarisant leurs enfants dans toutes les écoles

Il n'existe pas en SQL d'équivalent direct à la division !

La Division

Requête

Les parents scolarisant leurs enfants dans toutes les écoles

$\Leftrightarrow$

Les parents pour lesquels il n'existe pas d'école où un de leurs enfants n'est pas scolarisé

SQL

SELECT DISTINCT p1.parent FROM Parenté p1 WHERE NOT EXISTS(
*SELECT * FROM Scolarité s WHERE NOT EXISTS(*
*SELECT * FROM Parenté p2 WHERE p1.parent = p2.parent AND p2.enfant = s.enfant*

Parent
Pascal

Le tri des n-uplets

L'opérateur ORDER BY

Permet de trier les n-uplets selon un certain critère qui peut comprendre :

- Des noms d'attributs
- Des expressions calculées sur les colonnes
- Des numéros de positions de ces colonnes
- Des expressions dans la clause SELECT

Il est possible de préciser l'ordre du tri à l'aide du qualitatif :

- ASC pour un ordre croissant
- DESC Pour un ordre décroissant

Par défaut, le tri est dans l'ordre croissant

Les valeurs nulles sont classées en fin de liste par ordre croissant et en début de liste en cas de tri décroissant

Le tri des n-uplets

Requête

La liste des personnes triés par âge et nom

SQL

*SELECT * FROM Descriptif ORDER BY age personne*

Personne	**Age**	**Sexe**	**Ville**
Zoe	2	F	Nice
Leo	18	M	Paris
Marie	20	F	Paris
Clara	27	F	Nice
Raymond	40	M	Nice
Marcel	60	M	Marseille
Johnny	65	M	Lyon

Les fonctions de groupe

Définition

Un groupe est un ensemble n-uplets d'une table lesquels la valeur d'un attribut est constante. La fonction de groupe effectuent un calcul sur l'ensemble des valeurs d'un attribut pour un groupe de n-uplets :

Une fonction peut apparaître dans une clause SELECT, elle sera affichée comme un attribut ou une expression mais, en l'absence de clause GROUP BY, on ne peut mettre ensemble des fonctions de groupe et des attributs dans un même SELECT

Les fonctions de groupe

fonction COUNT

Fonction d'agrégation qui retourne le nombre total de n-uplet qui satisfait la condition spécifiée dans la condition WHERE. Si aucune condition n'est spécifiée alors la requête renvoie le nombre total de lignes de la table.

SELECT COUNT(*) FROM Parenté

Retourne le nombre de n-uplets de la table Parenté

SELECT COUNT(DISTINCT parent) FROM Parenté

Retourne le nombre de parents de la table Parenté sans les doublons

fonction MIN/MAX

Fonction qui retourne le la valeur minimale (resp maximale) d'un attribut.

SELECT MAX(age) FROM Descriptif

Retourne l'âge le plus haut de la table Descriptif

Les fonctions de groupe

fonction AVG

Fonction qui retourne la moyenne des valeurs présentes dans un attribut.

SELECT AVG(age) FROM Descriptif

Retourne l'âge moyen de la table Descriptif

fonction SUM

Fonction qui additionne les valeurs présentes dans un attribut.

SELECT SUM(age) FROM Descriptif

Retourne la somme des âges des personnes de la table Descriptif

clause GROUP BY

Permet de grouper plusieurs résultats et utiliser une fonction de totaux sur un groupe de résultat.

SELECT Sexe, AVG(age) FROM Descriptif GROUP BY Sexe

Retourne l'âge moyen pour chaque sexe