

Structured Query Language

SQL DML

L2A

Semestre 4
Mehdi Benzine

Introduction

SQL : (Structured Query Language) : langage de définition (DDL), de manipulation (DML) et de contrôle d'accès aux données de bases de données relationnelles (DCL):

- Développé à partir du langage SEQUEL de IBM.
- S'appuie sur les opérateurs de l'algèbre relationnelle.
- Norme ISO en 1986/1987 (SQL₁).
- Norme ISO en 1992 (SQL₂). La plus répandue.
- Norme ISO 1999 (SQL₃). Objet/Relationnel
- Norme 2003, Norme 2008.
- Utilisable en mode interactif, en mode procédural ou intégré dans un langage hôte (Java, C, Delphi, Php ...).
- Le langage de définition des données permet de créer, modifier, supprimer ou renommer les éléments constitutifs d'un schéma de base de données relationnelle.
- Le langage de manipulation de données permet d'interroger, ajouter des lignes dans une table, modifier et supprimer des lignes

Langage de manipulation de données

Le langage de manipulation des données comprend quatre instructions principales:

- L'instruction SELECT pour l'interrogation d'une ou plusieurs tables
- L'instruction INSERT pour l'ajout de lignes dans une table
- L'instruction UPDATE pour la modification de lignes
- L'instruction DELETE pour la suppression de lignes

Instruction SELECT

```
SELECT liste d'attributs  
FROM liste de tables
```

La clause SELECT permet de projeter les attributs souhaités.

La clause FROM permet de choisir les tables utilisées pour répondre à la requête.

Exemple:

```
SELECT Nom, Prénom  
FROM Employé
```

```
SELECT Num_Employé, Nom, Prénom, Date_Naissance, Fonction, Est_Cadre  
FROM Employé
```

```
SELECT *  
FROM Employé
```

* permet de projeter tous les attributs de la table.

Clause WHERE

La clause WHERE permet de sélectionner les tuples vérifiant un prédictat logique. Les connecteurs logiques et (AND), ou (OR), non (NOT) peuvent être utilisés pour construire le prédictat.

Exemple:

```
SELECT *  
FROM Employé  
WHERE Fonction = 'Administrateur'
```

```
SELECT Nom, Prénom, Date_Naissance  
FROM Employé  
WHERE Date_Naissance < '01/01/1988' AND  
Fonction = 'Administrateur'
```

Clause WHERE

Chercher les noms, prénoms et dates de naissance des employés nés avant 1988 ou n'étant pas administrateurs.

```
SELECT Nom, Prénom, Date_Naissance  
FROM Employé  
WHERE Date_Naissance < '01/01/1988' OR  
      Fonction <> 'Administrateur'
```

```
SELECT Nom, Prénom, Date_Naissance  
FROM Employé  
WHERE Date_Naissance < '01/01/1988' OR  
      NOT Fonction = 'Administrateur'
```

Liste de valeurs

Pour tester si la valeur d'un attribut appartient à une liste de valeurs on utilise IN.

Chercher les concepteurs et les administrateurs.

```
SELECT *  
FROM Employé  
WHERE Fonction = 'Concepteur' OR Fonction='Administrateur'
```

```
SELECT *  
FROM Employé  
WHERE Fonction IN ('Concepteur', 'Administrateur')
```

Intervalle de valeurs

Pour tester si la valeur d'un attribut est incluse dans un intervalle de valeurs on utilise BETWEEN.

Chercher les projets dont le budget est de 20000.00 DA à 50000.00DA.

```
SELECT *  
FROM Projet  
WHERE Budget >= 20000.00 AND Budget <= 50000.00
```

```
SELECT *  
FROM Projet  
WHERE Budget BETWEEN 20000.00 AND 50000.00
```

Filtre

On peut comparer une chaîne de caractères à un filtre en utilisant l'opérateur LIKE.

Chercher les employés dont le nom commence par 'B'.

```
SELECT *  
FROM Employé  
WHERE NOM LIKE 'B%'
```

% désigne n'importe quelle chaîne de caractères.

Valeurs nulles (NULL)

Il faut utiliser IS (ou IS NOT) pour tester si un attribut a une valeur ou non.

Chercher les numéros des employés n'ayant pas de supérieur pendant leur affectation à un projet.

```
SELECT Num_Employé  
FROM Affectation  
WHERE supérieur IS NULL
```

Valeurs distinctes

Pour éliminer les doublons dans le résultat d'une requête on utilise le mot clé DISTINCT dans la clause SELECT.

Afficher toutes les fonctions existantes.

```
SELECT Fonction  
FROM Employé
```

```
SELECT DISTINCT Fonction  
FROM Employé
```

Fonction
Concepteur
Chef de projet
Développeur
Analyste
Administrateur
Développeur

Fonction
Concepteur
Chef de projet
Développeur
Analyste
Administrateur

Clause ORDER BY

La clause ORDER BY permet de trier le résultat d'une requête.

Le tri peut se faire par ordre croissant ASC (option par défaut), ou par ordre décroissant (DESC).

Il est possible de trier un résultat sur plusieurs attributs.

Clause ORDER BY

```
SELECT DISTINCT Fonction  
FROM Employé  
ORDER BY Fonction
```

Fonction
Administrateur
Analyste
Chef de projet
Concepteur
Développeur

```
SELECT DISTINCT Fonction  
FROM Employé  
ORDER BY Fonction DESC
```

Fonction
Développeur
Concepteur
Chef de projet
Analyste
Administrateur

Opération ensemblistes

SQL permet de réaliser les 3 opérations ensemblistes de l'algèbre relationnelle.

Intersection: INTERSECT

Union: UNION

Différence: EXCEPT (MINUS sous Oracle)

Opération ensemblistes

SELECT *

FROM Employé

INTERSECT

SELECT *

FROM Retraité

SELECT *

FROM Employé

MINUS

SELECT *

FROM Retraité

SELECT *

FROM Employé

UNION

SELECT *

FROM Retraité

Fonctions d'agrégats

Il existe des fonctions de calcul qui s'appliquent à des groupes de tuples.
Parmi ces fonctions:

COUNT: compte le nombre de tuples appartenant au groupe.

SUM: calcule la somme des valeurs de l'attribut passé en paramètre.

AVG: calcule la moyenne arithmétique des valeurs de l'attribut passé en paramètre.

MIN: calcule le minimum des valeurs de l'attribut passé en paramètre.

MAX: calcule le maximum des valeurs de l'attribut passé en paramètre.

Fonctions d'agrégats

Calculer le nombre totale d'employé

```
SELECT COUNT(*)  
FROM Employé
```

Calculer le nombre de concepteurs

```
SELECT COUNT(*)  
FROM Employé  
WHERE FONCTION = 'Concepteur'
```

Fonctions d'agrégats

Calculer la somme des budgets de tous les projets

```
SELECT SUM(Budget)  
FROM Projet
```

Calculer la moyenne des budgets des projets ayant démarré en 2011

```
SELECT AVG(Budget)  
FROM Projet  
WHERE Date_Debut >= '01/01/2011' AND  
      Date_Debut <= '31/12/2011'
```

Instruction SELECT multi-tables

Chercher pour chaque employé ses dates de début d'affectation à des projets.

```
SELECT Employé.Num_Employé, Nom, Prénom, date_Naissance, Fonction,  
       Est_Cadre, Debut_Affect  
FROM Employé, Affectation
```

Cette requête calcul le produit cartésien !!!!

Pour calculer la jointure naturelle, il faut spécifier le prédicat de jointure dans la clause WHERE.

```
SELECT Employé.Num_Employé, Nom, Prénom, Date_Naissance, Fonction,  
       Est_Cadre, Debut_Affect  
FROM Employé, Affectation  
Where Employé.Num_Employé = Affectation.Num_Employé
```

Jointures

La norme SQL2 apporte une nouvelle syntaxe pour exprimer les jointures:

Jointure naturelle: jointure sur attributs de mêmes noms

```
SELECT *  
FROM Employé NATURAL JOIN Affectation  
WHERE Fonction = 'Concepteur'
```

Jointures

Jointure interne:

```
SELECT *  
FROM Employé INNER JOIN Affectation ON  
Employé.Num_Employé = Affectation.Num_Employé  
WHERE Fonction = 'Concepteur'
```

Jointure externe

Dans une opération de jointure classique (interne, inner join) entre deux relations R et S, les tuples de R (resp. S) qui ne sont reliés à aucun tuple de S (resp. R) ne sont pas présents dans le résultat de la requête.

La jointure externe conserve les tuples de R (resp. S) qui ne sont associés avec aucun tuple de S (resp. R).

La jointure externe peut être gauche (left outer join), qui préservent tous les tuples de la relation de gauche (R), droite (right outer join), qui conservent tous les tuples de la relation de droite (S) ou totale (full outer join), qui conserve tous les tuples des deux relations. Les tuples qui ne sont pas associés à aucun tuple sont reliés à un tuple ayant des valeurs NULL pour l'ensemble de ses attributs.

Num_Employé	Nom	Prénom	Date_Naissance	Fonction	Est_Cadre
1001	Belaid	Toufik	12/05/1965	Concepteur	true
1009	Touati	Rachid	13/09/1941	Chef de projet	true
1023	Kadri	Amine	23/11/1970	Développeur	true
1053	Djabi	Fatiha	04/06/1980	Analyste	false
1026	Bouras	Kamel	19/04/1968	Administrateur	true
1005	Djabi	Fatiha	22/08/1976	Développeur	false

Num_Employé	Num_Projet	Début_Affect	Fin_Affect	Supérieur
1009	122	07/03/2011	13/11/2011	NULL
1001	122	08/03/2011	28/06/2011	1009
1023	122	15/06/2011	04/10/2011	1009
1009	103	12/09/2010	01/11/2010	NULL
1001	208	15/06/2011	12/10/2011	1009
1009	208	15/06/2011	06/03/2012	NULL
1023	208	01/09/2011	17/12/2011	1009
1009	133	06/11/2011	19/02/2012	NULL
1053	208	01/09/2011	06/03/2012	1026
1026	208	19/08/2011	06/03/2012	1009

Jointure externe

```
SELECT Nom, Prénom, Num_Projet  
FROM Employé LEFT OUTER JOIN Affectation ON  
Employé.Num_Employé = Affectation.Num_Employé
```

```
SELECT Nom, Prénom, Num_Projet  
FROM Affectation RIGHT OUTER JOIN Employé ON  
Employé.Num_Employé = Affectation.Num_Employé
```

```
SELECT Nom, Prénom, Num_Projet  
FROM Affectation FULL OUTER JOIN Employé ON  
Employé.Num_Employé = Affectation.Num_Employé
```

Jointure externe

Ces deux notations sont équivalentes:

```
SELECT Nom, Prénom, Num_Projet  
FROM Employé LEFT OUTER JOIN Affectation ON  
Employé.Num_Employé = Affectation.Num_Employé
```

```
SELECT Nom, Prénom, Num_Projet  
FROM Employé,Affectation  
WHERE Employé.Num_Employé = Affectation.Num_Employé(+)
```

Nom	Prénom	Num_Projet
Touati	Rachid	122
Belaid	Toufik	122
Kadri	Amine	122
Touati	Rachid	103
Belaid	Toufik	208
Touati	Rachid	208
Kadri	Amine	208
Touati	Rachid	133
Djabi	Fatiha	208
Bouras	Kamel	208
Djabi	Fatiha	NULL

Nom	Prénom	Num_Projet
Touati	Rachid	122
Belaid	Toufik	122
Kadri	Amine	122
Touati	Rachid	103
Belaid	Toufik	208
Touati	Rachid	208
Kadri	Amine	208
Touati	Rachid	133
Djabi	Fatiha	208
Bouras	Kamel	208

SELECT Nom, Prénom, Num_Projet
 FROM Employé e, Affectation a
 WHERE e.Num_Employé = a.Num_Employé (+)

SELECT Nom, Prénom, Num_Projet
 FROM Employé e, Affectation a
 WHERE e.Num_Employé = a.Num_Employé

Jointures

Syntaxe SQL2 pour les jointures:

SELECT ...

FROM Table1 [NATURAL][LEFT| RIGHT|
FULL][INNER|OUTER] JOIN Table2[ON att11 θ att21
[AND ...]]

...

$\theta \in \{=, <>, >, <, >=, <= \}$

Colonnes renommées

Il est possible de renommer une colonne projetée en utilisant le mot clé AS

```
SELECT Nom AS LName, Prénom AS Fname  
FROM Employé
```

```
SELECT COUNT(*) AS Nombre_Employés  
FROM Employé
```

Utilisation d'alias de tables

Lorsque plusieurs attributs utilisés dans une requête portent le même nom, il est nécessaire de les préfixer par le nom de leur table d'origine pour les distinguer.

Il est possible de préfixer ces attributs par des alias (surnoms).

```
SELECT e.Num_Employé AS Numéro, Nom, Prénom  
FROM Employé e, Affectation a  
WHERE e.NumEmployé = a.Num_Employé AND  
Num_Projet = 122
```

Requêtes imbriquées

Une requête imbriquée (ou sous requête) est une requête SQL incluse dans la clause WHERE d'une autre requête.

Quels sont les employés ayant au moins une affectation

```
SELECT *  
FROM Employé  
WHERE Num_Employé IN (SELECT Num_Employé  
                      FROM Affectation)
```

```
SELECT *  
FROM Employé  
WHERE Num_Employé IN (1009, 1001, 1023, 1009, 1001, 1009, 1023, 1009,  
                      1053, 1026)
```

Requêtes imbriquées

Quels sont les employés ayant au moins une affectation ?

```
SELECT *
FROM Employé a
WHERE EXISTS (SELECT *
               FROM Affectation a
               WHERE e.Num_Employé = a.Num_Employé)
```

Quels sont les employé n'ayant aucune affectation ?

```
SELECT *
FROM Employé a
WHERE NOT EXISTS (SELECT *
                   FROM Affectation a
                   WHERE e.Num_Employé = a.Num_Employé)
```

Requêtes imbriquées

Quels sont les employés n'ayant aucune affectation ?

```
SELECT *  
FROM Employé  
WHERE Num_Employé NOT IN (SELECT Num_Employé  
                           FROM Affectation)
```

Quel est le nom et le prénom du supérieur de l'employé 1023 sur le projet 122 ?

```
SELECT Nom, Prénom  
FROM Employé  
WHERE Num_Employé = (SELECT Supérieur  
                           FROM Affectation  
                           WHERE Num_Employé = 1023 AND  
                                 Num_Projet = 122)
```

Requêtes imbriquées

Quels sont les employés n'ayant aucune affectation ?

```
SELECT *  
FROM Employé  
WHERE Num_Employé <> ALL (SELECT Num_Employé  
                           FROM Affectation)
```

Quels sont les employés ayant au moins une affectation ?

```
SELECT *  
FROM Employé  
WHERE Num_Employé = ANY (SELECT Num_Employé  
                           FROM Affectation)
```

Clause GROUP BY

Les fonctions d'agrégat calcul un résultat agrégé sur un groupe de tuples.

Si la clause GROUP BY n'est pas spécifiée, un seul groupe comprenant tous les tuples est considéré.

```
SELECT COUNT(*)  
FROM Projet
```

```
SELECT SUM(Budget)  
FROM Projet  
WHERE Date_début BETWEEN '01/01/2011' AND '31/12/2011'
```

La clause GROUP BY permet de constituer des groupes de tuples en spécifiant des attributs de groupement .

Clause GROUP BY

Calculer le nombre d'affectations de chaque employé

```
SELECT Num_Employé, COUNT(*) AS nb_affect  
FROM Affectation  
GROUP BY Num_Employé
```

La clause SELECT ne peut contenir que les attributs de groupement (présents dans la clause GROUP BY) et le résultat d'une fonction d'agrégat.

Num_Employé	Num_Projet	Début_Affect	Fin_Affect	Supérieur
1009	122	07/03/2011	13/11/2011	NULL
1001	122	08/03/2011	28/06/2011	1009
1023	122	15/06/2011	04/10/2011	1009
1009	103	12/09/2010	01/11/2010	NULL
1001	208	15/06/2011	12/10/2011	1009
1009	208	15/06/2011	06/03/2012	NULL
1023	208	01/09/2011	17/12/2011	1009
1009	133	06/11/2011	19/02/2012	NULL
1053	208	01/09/2011	06/03/2012	1026
1026	208	19/08/2011	06/03/2012	1009

```

SELECT Num_Employé, COUNT(*) AS nb_affect
FROM Affectation
GROUP BY Num_Employé
    
```

Num_Employé	Nb_Affect
1009	4
1001	2
1023	2
1053	1
1026	1

Clause GROUP BY

Quel est le nombre d'affectations par projet

```
SELECT Num_Projet, COUNT(*) AS nb_affect  
FROM Affectation  
GROUP BY Num_Projet
```

Num_Employé	Num_Projet	Début_Affect	Fin_Affect	Supérieur
1009	122	07/03/2011	13/11/2011	NULL
1001	122	08/03/2011	28/06/2011	1009
1023	122	15/06/2011	04/10/2011	1009
1009	103	12/09/2010	01/11/2010	NULL
1001	208	15/06/2011	12/10/2011	1009
1009	208	15/06/2011	06/03/2012	NULL
1023	208	01/09/2011	17/12/2011	1009
1009	133	06/11/2011	19/02/2012	NULL
1053	208	01/09/2011	06/03/2012	1026
1026	208	19/08/2011	06/03/2012	1009

SELECT Num_Projet, COUNT(*) AS nb_affect
 FROM Affectation
 GROUP BY Num_Projet

Num_Projet	Nb_Affect
122	3
103	1
208	5
133	1

Clause HAVING

La clause WHERE permet de sélectionner les tuples qui vérifient un prédicat logique.

La clause HAVING permet de sélectionner les groupes vérifiant un prédicat logique.

Quels est le nombre d'affectations pour chaque employé ayant au moins 2 affectations.

```
SELECT Num_Employé, COUNT(*) AS nb_affect  
FROM Affectation  
GROUP BY Num_Employé  
HAVING COUNT(*) >= 2
```

Num_Employé	Num_Projet	Début_Affect	Fin_Affect	Supérieur
1009	122	07/03/2011	13/11/2011	NULL
1001	122	08/03/2011	28/06/2011	1009
1023	122	15/06/2011	04/10/2011	1009
1009	103	12/09/2010	01/11/2010	NULL
1001	208	15/06/2011	12/10/2011	1009
1009	208	15/06/2011	06/03/2012	NULL
1023	208	01/09/2011	17/12/2011	1009
1009	133	06/11/2011	19/02/2012	NULL
1053	208	01/09/2011	06/03/2012	1026
1026	208	19/08/2011	06/03/2012	1009

```

SELECT Num_Employé, COUNT(*) AS nb_affect
FROM Affectation
GROUP BY Num_Employé
HAVING COUNT(*) >=2
    
```

Num_Employé	Nb_Affect
1009	4
1001	2
1023	2

Afficher le budget moyen, et le budget maximum des projets pour chaque chef de projet (grouper les chefs de projet par nom et prénom). Ne conserver que les chefs de projet ayant travaillé sur au moins 5 projets.

```
SELECT Nom, Prénom, AVG(Budget), MAX(Budget)
FROM Employé e, Projet p, Affectation a
WHERE e.Num_Employé = a.Num_Employé AND
      p.Num_Projet = a.Num_Projet AND
      Fonction = 'Chef de projet'
GROUP BY Nom, Prénom
HAVING COUNT(*) >= 5
```

Chaines de caractères

Les fonctions suivantes peuvent être appliquées aux chaines de caractères (chaine VARCHAR2(60)):

UPPER(chaine): retourne chaine en majuscules UPPER('Concepteur') = 'CONCEPTEUR'

LOWER(chaine): retourne chaine en minuscules LOWER('Développeur') = 'développeur'

CHARACTER_LENGTH(chaine): retourne la taille réelle de la chaine
CHARACTER_LENGTH('Chef de Projet') = 14

SUBSTRING(chaine FROM début TO longueur): retourne une sous chaine de chaine commençant au caractère numéro début de chaine et de la longueur spécifiée
SUBSTRING('Administrateur' FROM 1 TO 5) = 'Admin'

POSITION(chaine 1 IN chaine): retourne la position de la sous chaine chaine1 dans chaine
POSITION('ste' IN 'Analyste') = 6

chaine 1 || chaine2 : concaténation de deux chaines de caractères
'Chef' || ' de projet' = 'Chef de projet'

Chaines de caractères

```
SELECT *  
FROM Employé  
WHERE LOWER(Fonction) = 'concepteur'
```

```
SELECT UPPER(Nom), UPPER(Prénom)  
FROM Employé
```

```
SELECT UPPER(Nom) || UPPER(Prénom) AS Nom_Prénom  
FROM Employé
```

```
SELECT UPPER(Nom), UPPER(Prénom)  
FROM Employé
```

UPPER(Nom)	UPPER(Prénom)
BELAID	TOUFIK
TOUATI	RACHID
KADRI	AMINE
DJABI	FATIHA
BOURAS	KAMEL
DJABI	FATIHA

```
SELECT UPPER(Nom) as Nom, UPPER(Prénom) AS Prénom  
FROM Employé
```

Nom	Prénom
BELAID	TOUFIK
TOUATI	RACHID
KADRI	AMINE
DJABI	FATIHA
BOURAS	KAMEL
DJABI	FATIHA

```
SELECT UPPER(Nom) || UPPER(Prénom) AS Nom_Prénom  
FROM Employé
```

Nom_Prénom
BELAIDTOUFIK
TOUATIRACHID
KADRIAMINE
DJABIFATIHA
BOURASKAMEL
DJABIFATIHA

```
SELECT UPPER(Nom) || ' ' || UPPER(Prénom) AS Nom_Prénom  
FROM Employé
```

Nom_Prénom
BELAID TOUFIK
TOUATI RACHID
KADRI AMINE
DJABI FATIHA
BOURAS KAMEL
DJABI FATIHA

Dates

Fonctions manipulant les dates:

TO_CHAR (var_date, format): crée une chaîne de caractère représentant la date var_date au format spécifié

TO_CHAR(MA_DATE, 'DD/MM/YYYY') = '14/03/2012'

TO_CHAR(MA_DATE, 'DD/MM/YYYY HH24:MI:SS') = '14/03/2012 16:41:47'

TO_CHAR(MA_DATE, 'MM - DD - YY') = '03 - 14 - 12'

TO_CHAR(MA_DATE, 'HH24:MI:SS') = '16:41:47'

```
SELECT Nom, Prénom, TO_CHAR(Date_Naissance, 'DD - MM - YYYY')
FROM Employé
```

Dates

TO_DATE (var_chaine, format): créé une date correspondant à la chaîne var chaine au format spécifié

TO_DATE('12/03/2011', 'DD/MM/YYYY') = '12/03/2011 00:00:00'

TO_DATE('12/03/2011 09:45:36', 'DD/MM/YYYY HH24:MI:SS') = '12/03/2011 09:45:36'

INSERT INTO Projet VALUES(

111,

'Installation de réseau filaire',

TO_DATE('20/04/2012 08:00:00'), 'DD/MM/YYYY HH24:MI:SS'),

TO_DATE('21/04/2012 15:00:00'), 'DD/MM/YYYY HH24:MI:SS'),

25000)

Dates

EXTRACT (CHAMPS FROM var_date): permet d'extraire un champs particulier de la date

`EXTRACT (YEAR FROM '02/12/2009') = 2009`

```
SELECT *
FROM Employé
WHERE EXTRACT (YEAR FROM Date_Naissance)
BETWEEN 1975 AND 1985
```

Dates

SYSDATE / CURRENT_DATE: variable contenant la date et l'heure courante

INSERT INTO Projet VALUES(

112,

'Installation progiciel gestion et comptabilité',

SYSDATE,

SYSDATE +2,

20000)

SYSDATE = '15/04/2012 09:20:25'

SYSDATE + 2 = '17/04/2012 09:20:25'

Structure générale d'une instruction SELECT

SELECT liste colonnes (projetées, calculées)

FROM listes tables

[**WHERE** prédicat de jointure AND prédicat de sélection]

[**GROUP BY** colonnes de groupement]

[**HAVING** prédicat de sélection de groupes]

[**ORDER BY** critères et ordre de tri]

Structure générale d'une instruction SELECT

Prédicat de sélection composé par:

- Comparateurs arithmétiques ($<$, \leq , $>$, \geq , \neq (\neq), $=$)
- Connecteurs logiques (AND, OR, NOT)
- Comparateur de chaines LIKE
- Intervalle BETWEEN
- Liste IN
- Valeurs nulles IS NULL
- Requête imbriquée IN, EXISTS, SOME, ANY, ALL

Instruction INSERT

L'instruction INSERT permet d'insérer un nouveau tuple dans une table.

INSERT INTO table (colonne1, colonne2, ...)VALUES (valeur1, valeur2, ...)

INSERT INTO table (colonne1, colonne2, ...) SELECT ...

INSERT INTO Employé (Num_Employé, Prénom, Nom, Date_Naissance, Fonction, Est_Cadre)

VALUES(1088, 'Kamel', 'Ketfi', '12/10/1980', 'Analyste', true)

Instruction INSERT

On ne renseigne que quelques attributs du tuple.

```
INSERT INTO Employé (Num_Employé, Nom, Prénom) VALUES (1061, 'Biri', 'Ali')
```

Quand on renseigne tous les attributs du tuple (dans l'ordre où ils sont définis dans le schéma), il n'est pas nécessaire de spécifier la liste des attributs.

```
INSERT INTO Employé VALUES (1071, 'Grir', 'Tarek', '14/03/1978', 'Webmaster', false)
```

On insère dans la table Retraité les employés de plus de 65 ans.

```
INSERT INTO Retraité (Num_Employé, Nom, Prénom, Date_Naissance, Fonction,  
Est_Cadre) SELECT *
```

```
FROM Employé  
WHERE Date_Naissance < '08/04/1947'
```

Instruction DELETE

L'instruction DELETE permet de supprimer les tuples d'une table vérifiant un prédicat logique.

DELETE FROM table [WHERE prédicat]

Si la clause WHERE n'est pas spécifiée tous les tuple de la table seront supprimés.

Supprimer tous les projet terminés depuis plus de 10 ans.

DELETE FROM Projet WHERE Date_Fin <'08/04/2002'

Supprimer toutes les affectations aux projets terminés depuis plus de 10 ans.

DELETE FROM Affectation WHERE Num_Projet IN (SELECT Num_Projet FROM Projet WHERE Date_Fin < '08/04/2002')

Instruction UPDATE

L'instruction UPDATE permet de modifier les valeurs des attributs des tuples vérifiant un prédictat logique.

UPDATE table SET colonne1 = valeur1, colonne2 = valeur2, ... [WHERE prédictat]

Si la clause WHERE n'est pas spécifiée tous les tuple de la table seront mis à jour.

Augmente le budget du projet 122 de 10000.00 DA

UPDATE Projet SET Budget = Budget + 10000 WHERE Num_Projet = 122

Augmente le budget du projet 122 de 10%

UPDATE Projet SET Budget = Budget *1.10 WHERE Num_Projet = 122

Changer la fonction de l'employé 1005 à Concepteur. Il devient cadre.

Update Employé SET Fonction ='Concepteur', Est_Cadre = true WHERE Num_Employé = 1005

Séparateur

Le point virgule est le séparateur reconnu par la quasi-totalité des SGBDR pour séparer deux instructions.

Prendre l'habitude de mettre un point virgule après chaque instruction.

```
Update Employé SET Fonction ='Concepteur', Est_Cadre = true WHERE Num_Employé =  
1005;
```

```
SELECT Num_Projet, COUNT(*) AS nb_affect  
FROM Affectation  
GROUP BY Num_Projet;
```

```
SELECT *  
FROM Employé a  
WHERE NOT EXISTS (SELECT *  
                  FROM Affectation a  
                  WHERE e.Num_Employé = a.Num_Employé);
```

Exercice 1

1. Quels sont les noms et prénoms des développeurs ?
2. Quels sont les noms et prénoms des cadres ?
3. Quels sont les noms, prénoms et dates de naissance des employés nés avant 1975 ?
4. Quels sont les noms, prénoms et dates naissance des employés nés entre 1970 et 1979 ?
5. Quels sont les Numéros des projets sur lesquels travaille l'employé 'Belaid' ?
6. Quelle est la date de début des projets sur lesquels a travaillé l'employé 'Kadri' ?
7. Quel est le nom et le prénom de l'employé responsable du projet 122 ? (Supérieur est NULL)
8. Quel est le nom et le prénom des employés ayant travaillé sur des projets dont le budget est supérieur à 70000.00 DA ?

Corrigé

1. SELECT Nom, Prénom
FROM Employé
WHERE Fonction = 'Développeur'
2. SELECT Nom, Prénom
FROM Employé
WHERE Est_Cadre = true
3. SELECT Nom, Prénom, Date_Naissance
FROM Employé
WHERE Date_Naissance < '01/01/1975'

Corrigé

5.

```
SELECT Nom, Prénom, Date_Naissance
FROM Employé
WHERE Date_Naissance BETWEEN '01/01/1970'
AND '31/12/1979'
```
6.

```
SELECT Num_Projet
FROM Employé, Affectation
WHERE Employé.Num_Employé =
Affectation.Num_Employé AND Nom = 'Belaïd'
```

Exercice 2

1. Calculer la somme des budget des projets 108, 122 et 208.
2. Calculer le nombre d'employés pour chaque fonction.
3. Calculer le nombre de cadres.
4. Ajouter un projet ayant pour numéro 255 et pour budget 20000.00 DA.
5. Diminuer de 5% les budget de tous les projets qui ont commencé en 2012.
6. Supprimer l'ensemble des affectations ayant pris fin il y a plus de 5 ans.
7. Calculer le budget minimum et maximum des projets terminés en 2011.
8. Augmenter le budget de tous les projet de 5%.
9. Mettez Est_Cadre à true pour tous les développeurs.