Introduction au langage SQL : DDL et DML

Diapositives de Cours

O. DUBOIS

Plan

2

- Introduction
- Le DDL (Data Description Language)
  - Création d'une base
  - Sélection d'une base
  - Modification des caractéristiques d'une base
  - Suppression d'une base
  - Création d'une table
  - Types de données
  - Contraintes
  - Modification de la structure d'une table
  - Suppression d'une table
- Le DML (Data Manipulation Language)
  - Manipulation des données
  - Interrogation des données
  - Création et modification de tables à partir de requêtes SFW
- Exemple

Introduction

3

- **SQL** = Structured Query Language
- La définition des structures de données (composants du schéma) et la manipulation du contenu (les données elles-mêmes) s'effectuent au moyen du langage SQL
  = langage de requête proposé par les SGBD
- Normalisation
  - 1986 : 1ère norme = SQL-86
  - ensuite : SQL-89 puis SQL-92 (SQL2)
  - 1999 : SQL:99 (SQL3), modifiée plusieurs fois depuis
  - dernière version en 2023 (SQL:2023)
- Les SGBD ne respectent jamais complètement les normes
  ⇒ plusieurs variantes suivant les SGBD

Introduction

4

- Des Grands SGBDR commerciaux
  - DB2 d'IBM
  - ORACLE d'Oracle Corp.
  - SQL Server de Microsoft
- de moins importants
  - SYBASE de PowerSoft (rachetée par SAP)
  - INFORMIX d'Informix (rachetée par IBM)
  - INGRES de Computer Associates (est désormais libre)
- et d'autres plus facile d'accès
  - Access de Microsoft
- Des SGBDR Open Source
  - MySQL de MySQL AB (MySQL AB a été rachetée par SUN qui a été acquis par Oracle)
  - MariaDB (branche libre issue de MySQL)
  - PostegreSQL (développée par le créateur d'INGRES)

Introduction

5

- Le langage SQL est déclaratif (non procédural)
- Une **instruction SQL** constitue une **requête**
  = description de l'opération que doit effectuer le SGBD
- Deux possibilités pour exécuter une requête :
  - à partir d'un "terminal"
    mode interactif : le résultat apparaît à l'écran
  - par programme
    C, C++, Java, PHP, ... : le résultat est placé dans les variables du programme

Introduction

6

- Une instruction est une combinaison de une ou plusieurs **clauses** introduites par des mots-clés
- Exemple :

SELECT [ALL | DISTINCT] *col1*, *col2*, ..., *colN*
FROM *table1, ..., tableN*
[WHERE *condition*]
[GROUP BY *col* [ASC | DESC]] [HAVING *condition*]]
[ORDER BY *col* [ASC | DESC] [,*col* [ASC | DESC]...]]

  - 5 clauses : SELECT, FROM, WHERE, GROUP BY et ORDER BY
  - les mots-clés entre crochet sont facultatifs
  - " | " indique les choix possibles
  - le souligné signifie "valeur par défaut"

Introduction

7

- Les instructions SQL ne sont pas sensibles à la casse, mais les mots-clés sont généralement écrits en majuscule
- Les instructions SQL peuvent être écrites sur plusieurs lignes
- Les instructions SQL se terminent par un point-virgule
- Commentaires :
  - en début de ligne : `--`
  - pour encadrer plusieurs lignes : `/* ... */`
- Les conventions de noms des objets d'une base de données varient d'un SGBD à l'autre, mais en général :
  - de 1 à 30 caractères
  - les noms doivent commencer par une lettre
  - toute combinaison de majuscules, minuscules, chiffres ou trait de soulignement "_"

Introduction

8

Le langage SQL se compose de plusieurs "sous-langages" :

- Le **DDL** (Data Description Language)
  = définition et création des objets de la base
  CREATE, ALTER, DROP
- Le **DML** (Data Manipulation Language)
  = traitement et manipulation des objets de la base
  SELECT, INSERT, UPDATE, DELETE
- Le **DCL** (Data Control Language)
  = gestion des privilèges, ou des différents droits des utilisateurs sur la base de données
  GRANT, REVOKE, DENY
- Le **TCL** (Transaction Control Language)
  = gestion de l'environnement transactionnel (gère les propriétés ACID des transactions : Atomicité, Cohérence, Isolation, Durabilité)
  SET TRANSACTION, COMMIT, ROLLBACK

Introduction

9

Le langage SQL se compose de plusieurs "sous-langages" :

- Extension **procédurale**
  = gestion de variables, structures de contrôle, fonctions, procédures, ...
- **SQL intégré** : "Embedded SQL"
  = permet d'utiliser SQL dans un langage (C, Java, ...)

Le DDL

10

- Création d'une base
- Sélection d'une base
- Modification des caractéristiques d'une base
- Suppression d'une base
- Création d'une table
- Types de données
- Contraintes
- Modification de la structure d'une table
- Suppression d'une table

Création d'une base

11

- Une base de données est créée grâce à la requête

  CREATE DATABASE *nom_de_la_base* [*paramètres*]

  - le nom de la base doit respecter les règles de nommage du SGBD
  - Comme paramètres, il est par exemple possible d'indiquer le jeu de caractères (codage) et la collation utilisée (règles de classement)

  Exemple MySQL :

```
CREATE DATABASE clicom
COLLATE utf8_general_ci;
```

Sélection, modification des caractéristiques et suppression d'une base

12

- La sélection de la base à utiliser se fait grâce à la requête

  USE *nom_de_la_base*

- Certains paramètres d'une base de données peuvent être modifiés grâce à la requête

  ALTER DATABASE *nom_de_la_base* [*paramètres*]

- Une base de données est supprimée grâce à la requête

  DROP DATABASE *nom_de_la_base*

Création d'une table

13

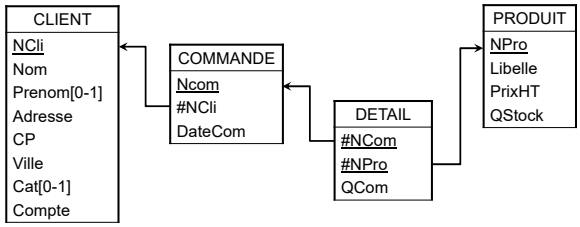
CREATE TABLE *nomtable*
(
nomcol1 *typedonnee1* [contraintes],
nomcol2 *typedonnee2* [contraintes],
...
nomcoln *typedonneen* [contraintes],
[contraintes]
)

- Pour créer une table, il faut indiquer son nom et tous ses attributs
- Pour chaque attribut, le nom, le type de données et éventuellement les contraintes sont indiqués

Exemple de base de données

14

- Structure de la base CLICOM



Exemple de base de données

15

- Contenu de la base **CLICOM**

CLIENT

NCli	Nom	Prenom	Adresse	CP	Ville	Cat	Compte
B062	Girard	Pierre	72 rue de la Gare	69001	Lyon	B2	-3200.00
B112	Herbin	Henry	23 allée Dumont	86000	Poitiers	C1	1250.00
B332	Monti	Claudine	112 rue Neuve	06000	Nice	B2	0.00
B512	Gillet	Jean-Claude	14 rue d'El Alamein	31000	Toulouse	B1	-8700.00
C003	Avron	Maurice	8 chemin de Cluzel	31000	Toulouse	B1	-1700.00
C123	Mercier	Gérard	25 rue Lemaitre	69003	Lyon	C1	-2300.00
C400	Ferard	Pierre	65 rue du Touffenet	86000	Poitiers	B2	350.00
D063	Mercier	Hélène	201 boulevard du Nord	31000	Toulouse		-2250.00
F010	Toussaint		5 rue Girouard	86000	Poitiers	C1	0.00
F011	Poncelet	Christian	17 Clos des Erables	31000	Toulouse	B2	0.00
F400	Jacob		78 chemin du Moulin	33000	Bordeaux	C2	0.00
K111	Vigneau	Jean	18 rue Faraday	59000	Lille	B1	720.00
K729	Noirot	Françoise	40 rue Fines	31000	Toulouse		0.00
L422	Franck		60 rue Bellecordière	69003	Lyon	C1	0.00
S127	Vilminot	Thierry	3 avenue des Roses	69001	Lyon	C1	-4580.00
S712	Guillaume		14a chemin des Roses	75013	Paris	B1	0.00

DETAIL

#NCom	#NPro	QCom
30178	CLA13	5
30179	CLA11	6
30179	CLE22	2
30182	CLE22	3
30184	CLA13	2
30184	CLE21	2
30185	CLA13	6
30185	CLE22	5
30185	SCA06	6
30186	CLE21	3
30188	CLA13	8
30188	CLE21	2
30188	CLE22	7
30188	IMP01	2

PRODUIT

NPro	Libelle	PrixHT	QStock
CLA11	Logitech G910 Orion Spectrum RGB	149.96	227
CLA12	Logitech Wireless Solar Keyboard K750	74.96	124
CLA13	Microsoft Wireless Comfort Desktop 5050	56.54	75
CLE21	SanDisk Extreme Go USB 3.1- 64 Go	41.63	121
CLE22	Corsair Flash Voyager USB 3.0 16 Go	14.58	70
IMP01	Canon i-SENSYS LBP113W	95.79	117
SCA06	Canon CanoScan LiDE 300	58.29	44

COMMANDE

NCom	#NCli	DateCom
30178	K111	2019-12-21
30179	C400	2019-12-22
30182	S127	2019-12-23
30184	C400	2019-12-23
30185	F011	2020-01-02
30186	C400	2020-01-02
30188	B512	2020-01-03

Création d'une table

16

- Ex : requête de création de la table CLIENT

```
CREATE TABLE CLIENT
    ( NCli char(10),
      Nom  char(30),
      Prenom  char(30),
      Adresse char(80),
      CP char(5),
      Ville char(30),
      Cat char(2),
      Compte decimal(10,2) )
```

Création d'une table

17

- Ex : requête de création de la table DETAIL :

```
CREATE TABLE DETAIL
    ( NCom int,
      NPro char(16),
      QCom int )
```

Types de données

18

- Le type de données d'une colonne précise le domaine de l'attribut
- Le langage SQL standard définit des types de données de base
- Les noms et les comportements de ces types de données varient suivant les SGBD
- Les principaux types de données sont les suivants :
  - Chaînes de caractères
  - Valeurs numériques
  - Chaînes binaires
  - Date et heure

Types de données

19

- Chaînes de caractères
  - longueur fixe CHAR(*n*)
    n = nombre de caractères
  - longueur variable VARCHAR(*n*)
    n = nombre maximum de caractères

  Rq : les valeurs des chaînes de caractères sont spécifiées entre guillemets simples (apostrophes)
- Valeurs numériques
  - entiers : TINYINT, SMALLINT, INT, BIGINT
  - nombres à virgule flottante : FLOAT(*p*)
    p : nombre total de chiffres
  - nombre à virgule fixe : DECIMAL(*p,s*) ou NUMERIC(*p,s*)
    - *p* : nombre maximal de chiffres décimaux
    - *s* : nombre de chiffres après la virgule

Types de données

20

- Chaînes binaires
  - Booléen BIT : 1 ou 0 (TRUE ou FALSE)
  - longueur fixe BINARY(*n*)
    n = nombre de bits
  - longueur variable VARBINARY(*n*)
    n = nombre maximum de bits
- Date et heure
  - DATE : année, mois, jour (date)
    sous la forme AAAA-MM-JJ
    ex : '2005-12-10'
  - TIME : heure, minute, seconde (instant)
    sous la forme HH:MM:SS
    ex : '08:18:40'
  - DATETIME : inclut DATE et TIME
    ex : '2005-12-10 08:18:40.2156483'

Spécification des contraintes

21

- Clé primaire
  La clause PRIMARY KEY (*col1*, ..., *coln*) désigne le ou les attributs qui forment la clé primaire
  Rq : s'il n'y a qu'un seul attribut, cette clause peut directement suivre l'attribut concerné
- Clé secondaire
  La clause UNIQUE (*col1*, ..., *coln*) désigne les éventuelles clés secondaires
- Contrainte référentielle : clé étrangère
  La clause FOREIGN KEY désigne le ou les attributs qui forment la clé étrangère

FOREIGN KEY (*liste*) REFERENCES *nomtable* (*col1*, ..., *coln*)
[ON DELETE {CASCADE | NO ACTION}]
[ON UPDATE NO ACTION]

Spécification des contraintes

22

- Colonnes facultatives/obligatoires
  Par défaut, toute colonne est facultative
  Le caractère obligatoire d'une colonne est spécifié par la clause NOT NULL
- Valeur par défaut pour une colonne
  La clause DEFAULT permet de définir une valeur par défaut à utiliser lors de l'insertion ou de la modification d'une donnée dans la colonne
- Contrainte sur les limites des valeurs des colonnes
  La clause CHECK (*condition*) permet d'indiquer ce genre de contrainte

Spécification des contraintes

23

- Ex : requête de création de la table CLIENT

```
CREATE TABLE CLIENT
      ( NCli char(10) PRIMARY KEY,
        Nom  char(30) NOT NULL,
        Prenom  char(30),
        Adresse char(80) NOT NULL,
        CP char(5) NOT NULL,
        Ville char(30) NOT NULL,
        Cat char(2) CHECK ( CAT IN ('B1', 'B2','C1','C2') ) ,
        Compte decimal(10,2) NOT NULL DEFAULT 0.00 );
```

Rq : CHECK n'est pas prise en compte sous MySQL

Spécification des contraintes

24

- Ex : requête de création de la table DETAIL :

```
CREATE TABLE DETAIL
      ( NCom int NOT NULL,
        NPro char(16) NOT NULL,
        QCom int NOT NULL,
PRIMARY KEY (NCom, NPro) ,
FOREIGN KEY (NCom) REFERENCES COMMANDE (NCom),
FOREIGN KEY (NPro) REFERENCES PRODUIT (NPro) );
```

Spécification des contraintes

25

- Le standard SQL impose que chaque contrainte ait un nom
- Ce nom est attribué
  - soit par le SGBD = contraintes non nommées
  - soit par l'utilisateur = contraintes nommées
    La clause CONSTRAINT *prefixeNom*
    permet de nommer une contrainte
    Ex :
    CONSTRAINT PKCLI PRIMARY KEY (NCLI)
    Conventions de préfixes (non imposées)
    PK : clé primaire (*Primary Key*)
    FK : clé étrangère (*Foreign Key*)
    NN : non NULL
    U : Unique

Modification de la structure d'une table

26

- La modification de la structure d'une table est réalisée par l'instruction
  ALTER TABLE *action*
- Différentes modifications peuvent être envisagées
  - Ajout de colonnes
  - Suppression de colonnes
  - Modification de colonnes existantes
  - Ajout de contraintes
    - clé primaire PRIMARY KEY
    - clés étrangères FOREIGN KEY
    - unicité UNIQUE
    - valeurs par défaut DEFAULT
    - contraintes de vérification CHECK
  - Suppression de contraintes

Ajouter une colonne

27

- L'instruction

  ALTER TABLE *table*
  ADD *col type* [*contrainte*]

  permet d'ajouter des colonnes dans une table existante

- La valeur initiale de chaque ligne de la nouvelle colonne est NULL

  Ex :
  ALTER TABLE CLIENT
  ADD Prenom2 VARCHAR(30);

Modifier des caractéristiques d'une colonne

28

- L'instruction

  ALTER TABLE *table*
  ALTER COLUMN *col modification*

  permet de modifier certaines caractéristiques des colonnes d'une table
  - type et dimension des données
  - valeur par défaut
  - NOT NULL
- Ces modifications ne sont possibles que si elles respectent le format des valeurs déjà présentes dans les lignes de la table
- Attention : syntaxe MySQL différente : ALTER TABLE *table*
  MODIFY *col modification*

Supprimer une colonne

29

- L'instruction

  ALTER TABLE *table*
  DROP COLUMN *col*

  permet de supprimer des colonnes dans une table existante

  Ex :
  ALTER TABLE CLIENT DROP COLUMN Prenom2 ;

Ajouter une contrainte

30

- Ajout de clés primaires
  L'instruction

  ALTER TABLE *table*
  ADD [CONSTRAINT *nomcontrainte*]
  PRIMARY KEY (*col1*[*, col2* [*, col3*] ...])

  permet d'ajouter une clé primaire dans une table existante

  Pour une clé multicomposant, il est possible de spécifier plusieurs colonnes

Ajouter une contrainte

31

- Ajout de clés étrangères
  L'instruction

  ALTER TABLE *table*
  ADD [CONSTRAINT *nomcontrainte*]
  FOREIGN KEY (*col-table*)
  REFERENCES *tablereference* (*col-tableref*)

  permet d'ajouter une clé étrangère dans une table existante

Supprimer une contrainte

32

- L'instruction

  ALTER TABLE *table*
  DROP PRIMARY KEY

  permet de supprimer une clé primaire

- L'instruction

  ALTER TABLE *table*
  DROP CONSTRAINT *nomcontrainte*

  permet de supprimer une contrainte nommée

Suppression d'une table

33

- L'instruction

  DROP TABLE *nomtable*

  permet de supprimer la table *nomtable*

  Si la table contient des lignes, la suppression est soumise aux contraintes référentielles qui concernent la table

  Ex : la suppression de la table COMMANDE pourrait entraîner la suppression de toutes les lignes de DETAIL

Le DML

34

- **Manipulation des données**
  - Insertion de lignes
  - Modification de données
  - Suppression de lignes
- Interrogation des données
  - Algèbre relationnelle
  - SELECT ... FROM ... WHERE
  - Fonctions
  - Fonctions agrégatives et regroupements
  - Opérateurs ensemblistes
  - Jointures
  - Sous-requêtes
- Création et modification de tables à partir de requêtes SFW

Insertion de lignes

35

- L'instruction

  INSERT INTO *nomtable* (col1, col2, ..., colN) VALUES
  (val11, val12, ..., val1N) ,
  ...
  (valK1, valK2, ..., valKN) ;

  permet d'ajouter des lignes dans une table

- Il n'est pas obligatoire de spécifier les colonnes à remplir
- Il faut utiliser le marqueur NULL pour insérer une valeur non spécifiée dans une colonne facultative
- Il faut utiliser le marqueur DEFAULT pour insérer la valeur par défaut définie pour une colonne

Insertion de lignes

36

- Ex :

```
INSERT INTO PRODUIT (NPro, Libelle, PrixHT, QStock)
VALUES ('SCA16','Kodak ScanMate i1150',449.13,6);
```

- Ex :

```
INSERT INTO CLIENT VALUES
('B112','Herbin', 'Henry','23 allée Dumont' ,
        '86000', 'Poitiers' ,'C1',1250),
('D063', 'Mercier', 'Hélène', '201 boulevard du Nord',
        '31000', 'Toulouse', NULL, -2250.00),
('F010', 'Toussaint', NULL, '5 rue Girouard',
        '86000', 'Poitiers', 'C1', DEFAULT);
```

Modification de données

37

- L'instruction

  UPDATE *nomtable*
  SET *col1* = *val1* [, ... , *colN* = *valN*]
  [WHERE condition]

  permet de modifier les valeurs des lignes d'une table

- La clause WHERE permet de sélectionner les lignes à modifier

  Ex :

  ```
  UPDATE PRODUIT
  SET LIBELLE = 'Kodak ScanMate i1150 (USB 2.0)'
  WHERE NPRO = 'SCA16';
  ```

Suppression de lignes

38

- L'instruction

  DELETE FROM *nomtable*
  WHERE condition

  permet de supprimer les lignes qui répondent aux conditions indiquées par WHERE

  Ex :

  ```
  DELETE FROM PRODUIT
  WHERE NPRO = 'SCA16';
  ```

- La suppression de lignes dans une table doit respecter les contraintes d'intégrité

Suppression de lignes

39

- L'instruction

  TRUNCATE TABLE *nomtable*

  permet de supprimer toutes les lignes de la table

Le DML

40

- Manipulation des données
  - Insertion de lignes
  - Modification de données
  - Suppression de lignes
- **Interrogation des données**
  - Algèbre relationnelle
  - SELECT ... FROM ... WHERE
  - Fonctions
  - Fonctions agrégatives et regroupements
  - Opérateurs ensemblistes
  - Jointures
  - Sous-requêtes
- Création et modification de tables à partir de requêtes SFW

Interrogation des données : Algèbre relationnelle

41

- L'algèbre relationnelle propose un ensemble d'opérations s'appliquant à une ou plusieurs relations (tables) pour donner une nouvelle relation (table)
- Ces opérations permettent d'exprimer des manipulations ensemblistes sur les données :
  Sélection (ou restriction), Projection, Union, Intersection, Différence, Produit cartésien, Jointure, Division
- Le langage SQL implémente ces opérations de l'algèbre relationnelle afin par exemple, de pouvoir extraire des données répondant à des critères précis (SELECT ... FROM ... WHERE)
- Opérations les plus courantes : Sélection, Projection et Jointure

Interrogation des données : Algèbre relationnelle

42

- La **sélection** est une opération unaire qui consiste à sélectionner un ensemble de lignes d'une table, selon un critère de sélection pouvant porter sur un ou plusieurs attributs
  Cette opération génère une autre table de même schéma que la table de départ
  ⇒ la sélection modifie (en réduction) le nombre de lignes mais le nombre de colonnes reste identique
- La **projection** consiste à :
  - ne retenir que certains attributs de la table
    = supprimer certaines colonnes
  - éliminer les lignes identiques
    = supprimer les lignes ayant le même ensemble de valeurs (doublons)

43

Interrogation des données :
Algèbre relationnelle

- La **jointure** permet d'obtenir une nouvelle table par la composition de deux tables
  Elle consiste à :
  - faire le produit cartésien des deux tables
  - effectuer une opération de sélection
  - effectuer ou non une opération de projection pour réduire le schéma de la table résultante

  Différents types de jointure :
  - Équi-jointure
  - Jointure naturelle
  - Theta-jointure
  - …

44

Interrogation des données
SELECT ... FROM ... WHERE

- L'extraction de données est réalisée par l'instruction SELECT
- Le résultat de l'exécution d'une requête SELECT est une table
- L'instruction

  SELECT [ALL | DISTINCT] *col1*, *col2*, ..., *colN*
  FROM *table1, ..., tableN*
  [WHERE *condition*]
  [GROUP BY *col* [ASC | DESC]] [HAVING *condition*]]
  [ORDER BY *col* [ASC | DESC] [,*col* [ASC | DESC]...]]

  permet d'interroger une base de données,
  c.-à-d. de récupérer des données
  = requête

45

Interrogation des données
SELECT ... FROM ... WHERE

- L'instruction SELECT comprend 5 parties principales :
- La clause SELECT précise les valeurs (liste les colonnes, valeurs dérivées) qui constituent chaque ligne du résultat
  Les attributs de cette clause sont donc les colonnes de la relation résultante
  L'option DISTINCT permet de supprimer les lignes doublons
- La clause FROM désigne les tables qui contiennent les colonnes à récupérer
- La clause WHERE spécifie les conditions à satisfaire par les lignes des tables indiquées par FROM

46

Interrogation des données
SELECT ... FROM ... WHERE

- La clause ORDER BY indique les critères choisis pour trier les lignes
  Cette clause ne concerne que l'affichage des données récupérées et non l'ordre interne des lignes dans la table
  ASC = croissant , DESC=décroissant
- La clause GROUP BY permet de faire des regroupements de lignes suivant la valeur de colonnes et en respectant éventuellement certaines condition précisées par HAVING
  Cette clause permet d'appliquer des fonctions agrégatives sur les colonnes du résultat

47

Interrogation des données
Clause SELECT

- Afficher toutes les colonnes
  = opérateur Sélection sans condition particulière

  Ex : pour extraire toutes les colonnes de CLIENT
  SELECT *
  FROM CLIENT;

- Extraction de certaines colonnes

  Ex : extraire les colonnes Ncli, Nom, Ville de CLIENT
  SELECT Ncli, Nom, Ville
  FROM CLIENT;

48

Interrogation des données
Clause SELECT

- Suppression des doublons (lignes en double)
  = implémentation l'opérateur Projection
  Utilisation de l'option DISTINCT du SELECT

  Ex : pour obtenir la liste des villes où résident les clients de la table CLIENT
  SELECT DISTINCT Ville
  FROM CLIENT;

Interrogation des données
Clause SELECT

49

- Alias

  Les alias permettent de renommer des colonnes à l'affichage ou des tables dans la requête
  = mot-clé AS

Ex : extraire les colonnes Ncli, Nom, Ville de CLIENT

```
SELECT Ncli AS 'Numéro de client',
       Nom AS 'Nom du client',
       Ville AS 'Ville du client'
FROM CLIENT ;
```

Résultat ⇒

Numéro de client	Nom du client	Ville du client
.	.	.
.	.	.
.	.	.

Interrogation des données
Clause SELECT

50

- Expressions

  Il est possible d'évaluer et d'afficher des expressions dans la clause SELECT

Ex :

```
SELECT 'TVA de ' , NPro , ' = ' , 0.21*PrixHT*QStock
FROM PRODUIT ;
```

Résultat ⇒

TVA de	NPro	=	0.21*PrixHT*QStock
TVA de	CLA11	=	7148.5932
TVA de	CLA12	=	1951.9584
TVA de	CLA13	=	890.5050
TVA de	CLE21	=	1057.8183
TVA de	CLE22	=	214.3260
TVA de	IMP01	=	2353.5603
TVA de	SCA06	=	538.5996

Interrogation des données
Clause ORDER BY

51

- Ordonnancement

  L'ordre des lignes du résultat d'une requête est arbitraire
  La clause ORDER BY permet de trier le résultat d'une requête suivant les valeurs de colonnes

Ex : affiche la table CLIENT triée par nom de Ville, par catégorie et par nom de client

```
SELECT * FROM CLIENT ORDER BY Ville, Cat, Nom;
```

= tri croissant (par défaut)

Ex : affiche la table PRODUIT triée par ordre décroissant des valeurs de quantité en stock

```
SELECT * FROM PRODUIT
ORDER BY QStock DESC;
```

= tri décroissant

Interrogation des données
Clause WHERE

52

- Restriction

  La clause WHERE permet de limiter le résultat aux lignes qui respectent une condition

WHERE *nomcol* opérateur-comp *valeur*

Ex : pour extraire les informations Ncli et Nom des lignes de CLIENT qui concernent des clients habitant Toulouse

```
SELECT Ncli, Nom
FROM CLIENT
WHERE Ville = 'Toulouse';
```

Interrogation des données
Clause WHERE

53

- Restriction

Ex : pour extraire les informations NPro et QCom du détail de la commande n° 30184

```
SELECT NPro AS 'Produit', QCom AS 'Quantité'
FROM DETAIL
WHERE NCom = '30184';
```

Interrogation des données
Clause WHERE

54

La condition de la clause WHERE peut être composée de prédicats incluant des colonnes ou expressions entre des opérateurs de comparaison, logiques ou intégrés

- Opérateurs de comparaison

Opérateur	Description
=	égal à
<>	différent de
<	inférieur à
<=	inférieur ou égal à
>	supérieur à
>=	supérieur ou égal à

Interrogation des données
Clause WHERE 55

- Opérateurs logiques
  La condition introduite par WHERE peut être constituée d'une expression booléenne de conditions élémentaires
  = opérateurs AND, OR, NOT et (parenthèses)

  Ex : pour afficher la liste de clients de Toulouse dont le compte est débiteur

```
SELECT NCli, Nom, Adresse, Compte
FROM CLIENT
WHERE Ville = 'Toulouse' AND Compte < 0;
```

Interrogation des données
Clause WHERE 56

- Opérateurs intégrés
  - Appartenance ou non à un intervalle

  BETWEEN *val1* AND val2
  ou NOT BETWEEN *val1* AND val2

  Ex :

```
SELECT Nom
FROM CLIENT
WHERE Compte BETWEEN 1000 AND 4000 ;
```

  - Appartenance ou non à une liste

  IN (*liste*) ou NOT IN (*liste*)

  Ex :

```
SELECT Nom
FROM CLIENT WHERE Cat NOT IN ('C1', 'C2', 'C3');
```

Interrogation des données
Clause WHERE 57

- Opérateurs intégrés
  - Présence ou non de certains caractères

  LIKE '*masque*' ou NOT LIKE '*masque*'
  '*masque*' décrit la structure générale des valeurs désignées
  Certains caractères jouent un rôle particulier
    - "_" (souligné) désigne un caractère quelconque
    - "%" (pourcent) désigne toute chaîne de caractères, éventuellement vide
    - pour rechercher explicitement "_" ou "%", il faudra les préfixer par un caractère spécial défini par une clause ESCAPE

Interrogation des données
Clause WHERE 58

- Opérateurs intégrés
  - Présence ou non de certains caractères

  Ex :

```
SELECT Nom
FROM CLIENT
WHERE CAT LIKE '_1';

SELECT Nom
FROM CLIENT
WHERE Adresse LIKE '%Neuve%';

SELECT Nom
FROM CLIENT
WHERE Adresse LIKE '%$_Canon%' ESCAPE '$';
```

Interrogation des données
Clause WHERE 59

- Opérateurs intégrés
  - Présence ou non de la valeur NULL

  IS NULL ou IS NOT NULL

  Ex :

```
SELECT Nom
FROM CLIENT WHERE Cat IS NULL;
```

Fonctions 60

Il est possible d'appliquer des fonctions sur les valeurs extraites d'une table

- Fonctions numériques
  - opérateurs arithmétiques classiques :
    + - * / et changement de signe **-**

  Ex :

```
SELECT 'TVA de ', NPro, ' = ' , 0.21*PrixHT*QStock
FROM PRODUIT
WHERE QStock > 120;
```

  - fonctions numériques :
    COS(*x*), LN(*x*), EXP(*x*), FLOOR(*x*), ROUND(*x*), RAND(*x*), ...

Fonctions

61

- Fonctions de chaînes de caractères
  - ASCII(*ch*) : valeur du code ASCII
  - CHAR(*n*) : caractère codé avec la valeur *n*
  - LEN(*ch*) ou CHAR_LENGTH(*ch*) : nombre de caractères
  - CONCAT(*ch1*,*ch2*) ou *ch1* || *ch2* : concaténation de chaînes de caractères
  - CHARINDEX(*ch1*,*ch2*) ou LOCATE(*ch1*,*ch2*) : position de *ch1* dans *ch2*
  - LOWER(*ch*) : mettre en minuscules
  - UPPER(*ch*) : mettre en majuscules
  - RIGHT(*n*) : extrait les *n* derniers caractères en partant de la droite
  - LEFT(*n*) : extrait les *n* premiers caractères en partant de la gauche
  - SUBSTRING(*str*,*pos*,*len*) : retourne une chaîne de *len* caractères de long de la chaîne *str*, à partir de la position *pos*
  - ...

Fonctions

62

- Fonctions sur les bits
  - **|** : OU bits à bits
  - **&** : ET bits à bits
  - **^** : OU exclusif bits à bits
  - **~** : inversion bits à bits
- Fonctions de conversion
  - CAST(*expr* AS *type*) : convertit *expr* dans le type *type*
  - CONVERT(*type,expr,style*) : convertit *expr* dans le type *type* selon un *style* donné
- Fonctions sur la date
  - DATEPART (*u*,*date*) ou EXTRACT(*u* FROM *date*) : donne sous forme numérique, le composant *u* de la valeur temporelle *date* (valeurs possibles de *u* : year, month, day, hour, minute, second, ..)
  - GETDATE() ou CURRENT_TIMESTAMP : retourne la date courante au format 'YYYY-MM-DD HH:MM:SS'

Fonctions

63

- Fonctions diverses
  - Fonctions de sélection

```
CASE WHEN c1 THEN v1
     WHEN c2 THEN v2
     .....
ELSE vn
END
```

renvoie *v1* si *c1* est vraie, sinon, renvoie *v2* si *c2* est vraie, ..., sinon renvoie *vn*

```
CASE expr
   WHEN c1 THEN v1
   WHEN c2 THEN v2
  ...
ELSE vn
END
```

Comme précédemment avec des conditions *expr* = *ci*

Fonctions

64

- Fonctions diverses
  - Fonctions de sélection

Ex :

```
SELECT Ncli, Nom,
    CASE SUBSTRING(Cat FROM 1 FOR 1)
        WHEN 'A' THEN 'bon'
        WHEN 'B' THEN 'moyen'
        WHEN 'C' THEN 'occasionnel'
        ELSE 'inconnu'
    END AS 'Type de client', Ville
FROM CLIENT
```

Fonctions

65

- Fonctions diverses
  - USER ou USER() : nom d'utilisateur et le nom d'hôte de la session courante
  - ISNULL(*expr*) : renvoie 1 si *expr* est NULL
  - ...

Fonctions agrégatives et regroupements

66

- Fonctions agrégatives (ou statistiques)
  - COUNT(*) : nombre de lignes trouvées
  - COUNT(*col*) : nombre de valeurs de la colonne
  - AVG(*col*) : moyenne des valeurs de la colonne
  - STDEV(*col*) ou STD(*col*) : écart type des valeurs de la colonne
  - VAR(*col*) ou VARIANCE(*col*) : variance des valeurs de la colonne
  - SUM(*col*) : somme des valeurs de la colonne
  - MIN(*col*) : minimum des valeurs de la colonne
  - MAX(*col*) : maximum des valeurs de la colonne

Fonctions agrégatives et regroupements

67

- Fonctions agrégatives (ou statistiques)

  Rq :
  - les valeurs NULL sont ignorées
  - pour ne pas prendre en compte les valeurs dupliquées, certaines fonctions acceptent DISTINCT

  Ex :

  Nombre de commandes passées :

  ```
  SELECT COUNT(NCli) FROM COMMANDE;
  ```

  Nombre de clients ayant passé au moins une commande :

  ```
  SELECT COUNT(DISTINCT NCli) FROM COMMANDE;
  ```

Fonctions agrégatives et regroupements

68

- Il est possible de regrouper des lignes grâce à GROUP BY et HAVING

  SELECT *colonnes*
  FROM *table*
  WHERE *condition*
  GROUP BY *colonnes* [HAVING *conditions*]

- GROUP BY permet d'indiquer le regroupement à réaliser
- HAVING permet de spécifier d'éventuelles conditions que le regroupement doit satisfaire

Fonctions agrégatives et regroupements

69

- Ex : Afficher pour chaque ville, son nom, le nombre de clients et le montant moyen des comptes des clients

```
SELECT Ville,
    COUNT(*) AS 'Nombre de clients',
    AVG(Compte) AS 'Moyenne des comptes'
FROM CLIENT
GROUP BY Ville;
```

Fonctions agrégatives et regroupements

70

- Ex : Afficher pour chaque ville ayant au moins 3 clients, son nom, le nombre de clients et le montant moyen des comptes des clients

```
SELECT Ville,
    COUNT(*) AS 'Nombre de clients',
    AVG(Compte) AS 'Moyenne des comptes'
FROM CLIENT
GROUP BY Ville HAVING COUNT(*) >= 3;
```

Opérateurs ensemblistes

71

- Les opérateurs ensemblistes UNION, INTERSECT et MINUS permettent de combiner les résultats de différentes instruction SFW
  - Union : l'opérateur UNION permet d'ajouter à une requête le résultat d'autres requêtes
  - Intersection : L'opérateur INTERSECT permet d'identifier les lignes communes à deux requêtes
  - Différence : L'opérateur EXCEPT permet de déterminer les lignes présentes dans la première requête mais pas dans la deuxième
- Rq :
  - ces opérateurs éliminent les lignes en double
    Pour empêcher ceci, il faut ajouter la clause ALL
  - seule l'union est possible avec MySQL

Jointures

72

- La jointure permet de regrouper dans une même table des données provenant de plusieurs autres
- Les tables sont reliées par leurs attributs communs qui se trouvent souvent dans une liaison de clé étrangère
- Les jointures sont classées selon
  - le type de condition impliquée
  - le nombre de tables en jeu
  - le type des lignes récupérées

Jointures

73

- Différents types de jointure :
  - Jointures internes (inner join) :
    - Equijointure : jointure entre deux tables connectées par des valeurs communes égales
    - Jointure naturelle : jointure entre deux tables connectées par des colonnes de même nom
    - Auto-jointure : jointure entre une table et elle-même
  - Jointure externe (outer join) : jointure entre deux tables qui inclue les lignes d'une table qui n'ont pas de correspondance dans l'autre table

Jointures

74

- Différents types d'écriture des jointures :
  Une jointure peut s'écrire dans la clause FROM ou dans la clause WHERE ⇒ plusieurs écritures possibles
  - Directives dans la clause FROM
    INNER JOIN, CROSS JOIN, NATURAL JOIN, OUTER JOIN

```
SELECT ...
FROM   t1
INNER JOIN t2 ON ... = ...
WHERE  ... ;
```

  - Clauses de jointure dans la clause WHERE

```
SELECT ...
FROM t1 , t2
WHERE  ... = ... ;
```

Jointures

75

- Différents types d'écriture des jointures :
  - Sous-requête (ou jointure procédurale) = une requête externe et des requêtes dans lesquelles se programment les jointures

```
SELECT ...
FROM   t1
WHERE  ... =
        ( SELECT ... FROM t2 WHERE ... )
AND ... ;
```

Equijointure

76

- Une équijointure utilise l'opérateur d'égalité dans la clause de jointure et compare généralement une clé étrangère avec une clé primaire (ou candidate) d'une table de référence

```
SELECT table1.col1, ..., table1.colN,
       table2.col1, ..., table2.colM
FROM table1
     INNER JOIN table2
       ON table1.col-commune = table2.col-commune;
```

Rq : la directive INNER est optionnelle (appliquée par défaut)

ou

```
SELECT table1.col1, ..., table1.colN,
       table2.col1, ..., table2.colM
FROM table1, table2
WHERE table1.col-commune = table2.col-commune;
```

Equijointure

77

- Les noms d'attribut identiques dans chaque table doivent être qualifiés (c.-à-d. précédé du nom de leur table)
- Cette jointure peut être généralisée à plus de deux tables
- Ex : Afficher la table des commandes complétée par le nom et la ville du client ayant passé la commande

```
SELECT COMMANDE.NCom, COMMANDE.NCli,
   COMMANDE.DateCom, CLIENT.Nom, CLIENT.Ville
FROM COMMANDE INNER JOIN CLIENT
ON CLIENT.NCli = COMMANDE.NCli;
```

ou

```
SELECT COMMANDE.NCom, COMMANDE.NCli,
   COMMANDE.DateCom, CLIENT.Nom, CLIENT.Ville
FROM COMMANDE, CLIENT
WHERE CLIENT.NCli = COMMANDE.NCli;
```

Equijointure

78

- Ex : Afficher pour chaque client ayant commandé au moins 2 fois le produit CLE21, son numéro, le nombre de commandes et la quantité de produit CLE21 commandée

```
SELECT NCli, COUNT(*), SUM(QCom)
FROM COMMANDE INNER JOIN DETAIL
    ON DETAIL.NCom = COMMANDE.NCom
WHERE NPro = 'CLE21'
GROUP BY NCli
HAVING COUNT(*) >=2
```

= jointure et regroupement de données

Alias de table

79

- Pour simplifier les requêtes, il est possible d'utiliser des alias pour les noms de tables
- Ex :

```
SELECT COMMANDE.NCom, COMMANDE.NCli,
   COMMANDE.DateCom, CLIENT.Nom, CLIENT.Ville
FROM COMMANDE INNER JOIN CLIENT
   ON CLIENT.NCli = COMMANDE.NCli;
```

Peut s'écrire :

```
SELECT A.NCom, A.NCli, A.DateCom, B.Nom, B.Ville
FROM COMMANDE AS A INNER JOIN CLIENT AS B
   ON B.NCli = A.NCli;
```

- Ces alias ne sont valables que dans l'instruction SELECT concernée
- Rq : AS est facultatif

Autojointure

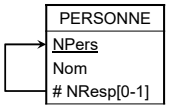
80

- Une auto-jointure est la jointure d'une table avec elle-même
- Dans ce cas, pour SELECT, il est alors plus facile de donner deux alias différents pour une même table
- Ex : soit la table PERSONNE qui présente la liste du personnel avec pour chacun, le numéro de son responsable
  Afficher pour chaque personne ayant un responsable, le numéro et le nom de celui-ci

```
SELECT S.NPers, R.NPers, R.Nom
FROM PERSONNE S INNER JOIN PERSONNE R
   ON R.NResp = S.NPers;
```

ou

```
SELECT S.NPers, R.NPers, R.Nom
FROM PERSONNE S, PERSONNE R
WHERE S.NResp = R.NPers;
```



Thêta-jointure

81

- A l'inverse des équijointures, la clause d'une thêta-jointure n'est pas forcément basée sur l'égalité (le plus souvent de clés), mais inclut tout autre type d'opérateur de comparaison (!=, <>, >, <, >=, <=, BETWEEN, LIKE, IN)

Jointures externes

82

Une jointure externe permet d'ajouter au résultat d'une jointure classique (interne) les lignes d'une des tables qui n'ont pas de correspondance dans l'autre table (lignes célibataires)

- Jointure externe gauche :

```
SELECT table1.col1, ..., table1.colN,
            table2.col1, ..., table2.colM
FROM table1 LEFT OUTER JOIN table2
   ON condition_de_jointure
```

ajoute au résultat de la jointure classique (INNER JOIN) toutes les lignes de *table1* qui n'ont pas de correspondance dans *table2*

Jointures externes

83

- Jointure externe droite :

```
SELECT table1.col1, ..., table1.colN,
            table2.col1, ..., table2.colM
FROM table1 RIGHT OUTER JOIN table2
   ON condition_de_jointure
```

ajoute au résultat de la jointure classique (INNER JOIN) toutes les lignes de *table2* qui n'ont pas de correspondance dans *table1*

Jointures externes

84

- Ex : Afficher la table des commandes complétée par le nom et la localité de tous les clients, même de ceux qui n'ont pas passé de commande

```
SELECT M.NCom, C.NCli, M.DateCom, C.Nom, C.Ville
FROM COMMANDE M
   RIGHT OUTER JOIN CLIENT C
      ON M.NCli = C.Ncli
ORDER BY M.NCom;
```

NULL apparaîtra dans les colonnes NCom et DateCom pour ceux qui n'ont pas passé de commande

Jointures externes

85

- Jointure externe complète :

  SELECT *table1.col1*, ..., *table1.colN*,
  table2.col1, ..., *table2.colM*
  FROM *table1* FULL OUTER JOIN *table2*
  ON *condition_de_jointure*

  ajoute au résultat de la jointure classique (INNER JOIN) toutes les lignes de *table2* qui n'ont pas de correspondance dans *table1* et toutes les lignes de *table1* qui n'ont pas de correspondance dans *table2*

Sous-requêtes

86

- Extraction de données d'une seule table en fonction des liaisons des lignes vers d'autres tables
  Ex :
  - les commandes des clients de Lyon
  - les clients qui commandent le produit CLE22
- Les sous-requêtes (ou requêtes imbriquées) permettent d'obtenir les valeurs d'une table dont les lignes satisfont à une condition WHERE qui dépend des valeurs renvoyées par une autre instruction SELECT

Sous-requêtes

87

- Syntaxe générale d'une sous-requête :

```
SELECT...
FROM    ...
WHERE   ...
```

← Requête externe
Cette requête s'exécute en dernier

```
( SELECT    ...
  FROM      ...
  WHERE     ... )
```

← Sous-Requête
Cette requête interne s'exécute en premier

Sous-requêtes

88

- Ex : Afficher les commandes des clients de Lyon

1. sans sous-requête :

   - affichage des numéros des clients de Lyon

```
SELECT Ncli
FROM CLIENT WHERE Ville = 'Lyon';
```

⇒ résultat :

NCli
C132
S127
B062
L422

   - affichage des numéros de commandes

```
SELECT NCom, DateCom
FROM COMMANDE WHERE NCli
    IN ('C132','S127','B062', 'L422');
```

Sous-requêtes

89

- Ex : Afficher les commandes des clients de Lyon

2. avec une sous-requête :

```
SELECT NCom, DateCom
FROM COMMANDE
WHERE NCli IN (  SELECT Ncli
                 FROM CLIENT
                 WHERE Ville = 'Lyon');
```

Sous-requête à une seule ligne

90

- Les sous-requêtes qui retournent une seule valeur s'appellent des sous-requêtes à une seule ligne
- Avec ces sous-requêtes, il est possible d'utiliser des opérateurs de comparaison classiques
  =, <>, >, >=, <, et <=

Sous-requête à une seule ligne

91

- Ex : Afficher les clients dont le compte est supérieur au compte du client C400

```
SELECT *
FROM CLIENT
WHERE Compte > (
  SELECT Compte  FROM CLIENT
  WHERE NCli = 'C400' );
```

Sous-requête à lignes multiples

92

- Les sous-requêtes à lignes multiples retournent une ou plusieurs valeurs
- Ces sous-requêtes peuvent être incorporées à la clause WHERE au moyen des opérateurs multilignes de comparaison suivants :

Opérateur	Signification
IN	égal à une valeur récupérée dans une sous-requête
NOT IN	différent de ou non égal à une valeur récupérée dans une sous-requête
ANY	compare à chaque valeur retournée par une sous-requête au moins un élément de l'ensemble doit satisfaire la comparaison
ALL	compare à chaque valeur retournée par une sous-requête tous les éléments de l'ensemble doivent satisfaire la comparaison

Sous-requête à lignes multiples

93

- Ex : Afficher les détails de commande spécifiant la quantité minimale de du produit CLE22

```
SELECT *
FROM DETAIL
WHERE QCom <= ALL ( SELECT QCom
            FROM DETAIL WHERE NPro = 'CLE22' )
AND NPro = 'CLE22';
```

<=> la valeur de QCom doit être inférieure à chacun des éléments de (SELECT QCom ...)

Sous-requête à lignes multiples

94

- Ex : Afficher les détails de commande de CLE22 dont la quantité commandée n'est pas minimale

```
SELECT *
FROM DETAIL
WHERE QCom > ANY ( SELECT QCom
            FROM DETAIL WHERE NPro = 'CLE22' )
AND NPro = 'CLE22';
```

<=> la valeur de QCom est supérieure à au moins un élément de (SELECT QCom ...)

Sous-requête corrélée

95

- Une sous-requête corrélée, est une requête imbriquée pour laquelle la requête interne se réfère toujours à la table mentionnée dans la clause FROM de la requête externe

```
SELECT  col1, col2, ..., coln
FROM  table-externe
WHERE  valeur-col-externe  IN
      ( SELECT nom-col-interne
      FROM  table-interne
      WHERE  col-interne = col-externe )
```

variable de corrélation

Sous-requête corrélée

96

- Ex : Afficher les clients dont le compte est supérieur à la moyenne des comptes des clients de la même ville

```
SELECT NCli, Nom, Ville, Compte
FROM CLIENT AS S
WHERE Compte > (
      SELECT AVG(Compte)
      FROM CLIENT
      WHERE Ville = S.Ville )
```

dans cet exemple, la table externe est identique à la table interne
⇒ utilisation d'un alias de table

Sous-requête corrélée

97

- La condition peut également porter sur l'existence (EXISTS) ou l'inexistence (NOT EXISTS) d'au moins une ligne dans le résultat d'une sous-requête
- Ex : Afficher la liste des produits qui n'ont pas été commandés

```
SELECT NPro, Libelle
FROM PRODUIT AS P
WHERE NOT EXISTS ( SELECT *
                   FROM DETAIL
                   WHERE NPro = P.NPro );
```

Le DML

98

- Manipulation des données
  - Insertion de lignes
  - Modification de données
  - Suppression de lignes
- Interrogation des données
  - Algèbre relationnelle
  - SELECT ... FROM ... WHERE
  - Fonctions
  - Fonctions agrégatives et regroupements
  - Opérateurs ensemblistes
  - Jointures
  - Sous-requêtes
- **Création et modification de tables à partir de requêtes SFW**

Création et modification de tables à partir de requêtes SELECT ... FROM ... WHERE

99

- Il est possible de créer ou de modifier des tables grâce à des sous-requêtes combinant une expression **SFW** aux instructions de création ou de modification de tables

Création d'une table à partir de données extraites d'autres tables

100

- Il est possible de créer une table à partir d'une expression SFW
  - Sous SQL SERVER : grâce à la clause INTO

```
SELECT col1, col2, col3, ..., coln
INTO nouvelletable
FROM liste_table
WHERE conditions
```

  - Sous Mysql : grâce à la CREATE TABLE ... AS ...

```
CREATE TABLE nouvelletable AS
SELECT col1, col2, col3, ..., coln
FROM liste_table
WHERE conditions
```

Création d'une table à partir de données extraites d'autres tables

101

- Ex : Créer une table CLIENT_TOULOUSE qui ne présente que les clients habitant Toulouse
  - Sous SQL SERVER

```
SELECT NCli, Nom, Prenom, Adresse, Cat, Compte
INTO CLIENT_TOULOUSE
FROM CLIENT
WHERE Ville = 'Toulouse'
```

  - Sous MySQL

```
CREATE TABLE  CLIENT_TOULOUSE AS
SELECT NCli, Nom, Prenom, Adresse, Cat, Compte
FROM CLIENT
WHERE Ville = 'Toulouse'
```

Modification de lignes à partir de données extraites de tables

102

- Il est possible de modifier les données d'une table avec des valeurs extraites d'autres tables grâce à SFW attachée à l'instruction UPDATE

```
UPDATE table
SET colonne = (SELECT nomcol
               FROM table
               WHERE condition )
WHERE nomcol = (SELECT nomcol
                FROM table
                WHERE condition )
```

Modification de lignes à partir de données extraites de tables

103

- Ex : Déduire du stock de chaque produit la quantité en commande

```
UPDATE PRODUIT
SET QStock = QStock - ( SELECT SUM(QCom)
                        FROM DETAIL
                        WHERE NPro = PRODUIT.NPro )
WHERE EXISTS (SELECT *
              FROM DETAIL
              WHERE   NPro = PRODUIT.NPro);
```

Ajout de lignes à partir de données extraites de tables

104

- Il est possible d'insérer dans une table des données extraites d'autres tables grâce à une expression SFW attachée à l'instruction INSERT

```
INSERT INTO table (col1, col2, ..., coln)
SELECT col1, col2, ..., coln
FROM table
WHERE nomcol = (SELECT nomcol
                FROM table
                WHERE condition )
```

Ajout de lignes à partir de données extraites de tables

105

- Ex : Ajouter à la table CLIENT_TOULOUSE, une ligne extraite de la table CLIENT

```
INSERT INTO CLIENT_TOULOUSE
SELECT NCli, Nom, Adresse, Cat, Compte
FROM CLIENT
WHERE Ville = 'Toulouse';
```

Suppression de lignes à partir de données extraites de tables

106

- Il est possible de supprimer dans une table des données extraites d'autres tables grâce à une expression SFW attachée à l'instruction DELETE

```
DELETE FROM table
WHERE nomcol = (SELECT nomcol
                FROM table
                WHERE condition )
```

Suppression de lignes à partir de données extraites de tables

107

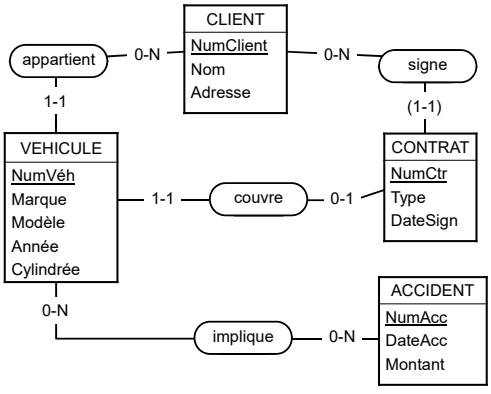
- Ex : Supprimer de la table DETAIL les lignes qui spécifient un produit en rupture de stock

```
DELETE FROM DETAIL
WHERE NPro IN ( SELECT Npro
                FROM PRODUIT
                WHERE Qstock <= 0 );
```

Exemple : Assurance

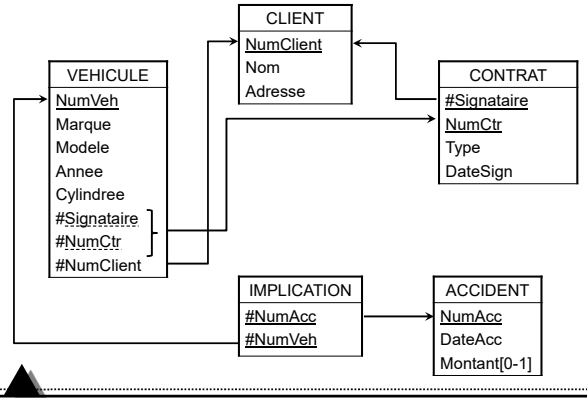
108

1. Modèle entité-association



Exemple : Assurance 109

2. Tables



Exemple : Assurance 110

3. Requêtes SQL de création des tables

```
create table Client (
NumClient char(12) not null,
Nom char(38) not null,
Adresse char(60) not null,
primary key (NumClient));

create table Contrat (
Signataire char(12) not null,
NumCtr int not null,
Type char(12) not null,
DateSign date not null,
primary key (Signataire,NumCtr),
foreign key (Signataire) references Client (NumClient));
```

Exemple : Assurance 111

3. Requêtes SQL de création des tables

```
create table Vehicule (
NumVeh char(16) not null,
Marque char(30) not null,
Modele char(30) not null,
Annee decimal(4) not null,
Cylindree decimal(6) not null,
Signataire char(12) not null,
NumCtr int not null,
NumClient char(12) not null,
primary key (NumVeh),
unique (Signataire,NumCtr),
foreign key (NumClient) references Client (NumClient),
foreign key (Signataire,NumCtr)
  references Contrat (Signataire,NumCtr));
```

Exemple : Assurance 112

3. Requêtes SQL de création des tables

```
create table Accident (
NumAcc char(10) not null,
DateAcc date not null,
Montant decimal(6),
primary key (NumAcc));

create table Implication (
NumVeh char(16) not null,
NumAcc char(10) not null,
primary key (NumVeh,NumAcc),
foreign key (NumVeh) references Vehicule (NumVeh),
foreign key (NumAcc) references Accident (NumAcc));
```