

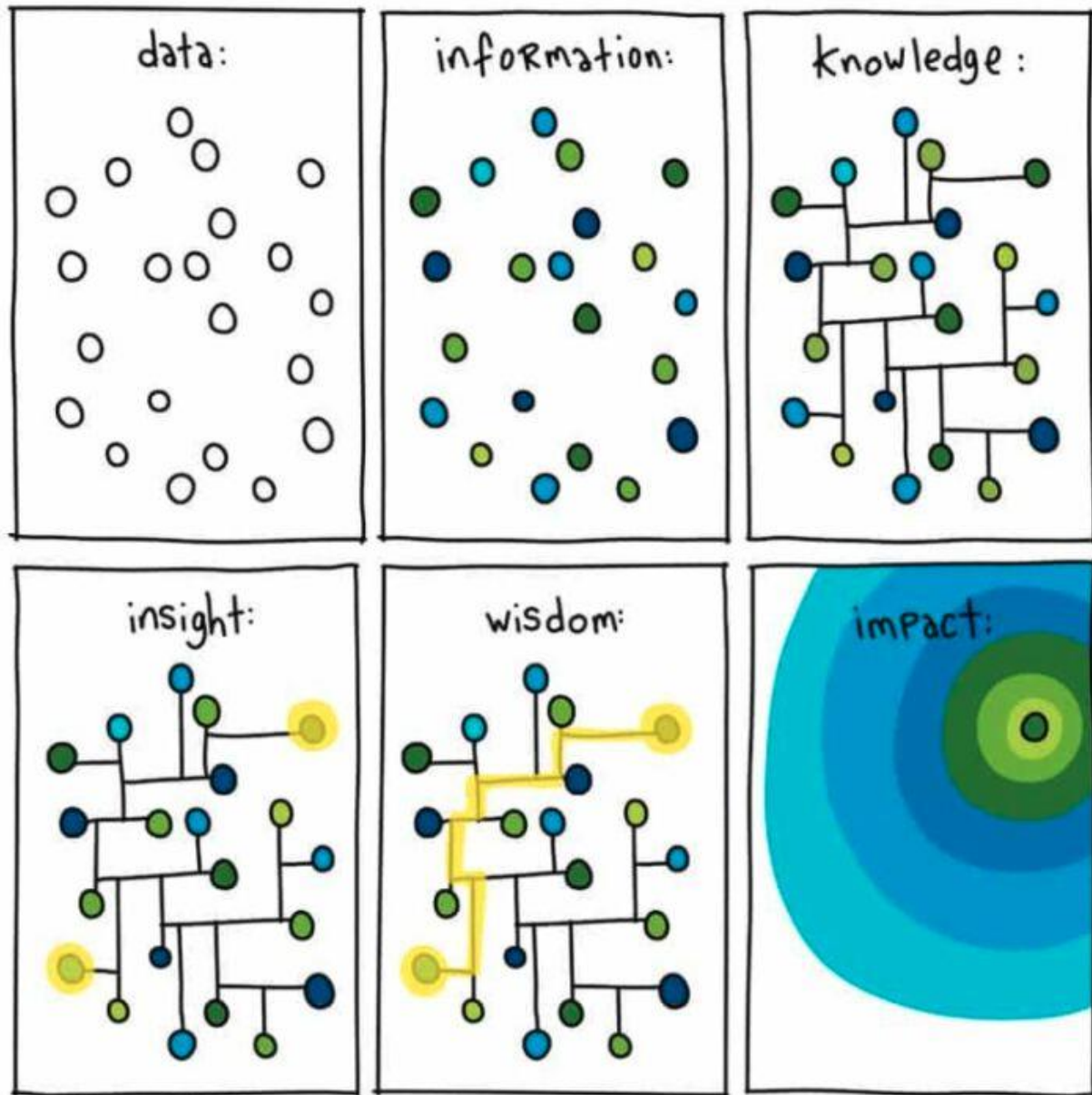
Systemes de Gestion de Bases de Données Relationnelles- Langage SQL

Pr. Fatima MOURCHID

Partie I: Introduction

Introduction

- ▶ Toute solution de base de données commence par une stratégie de conception et de mise en œuvre
- ▶ Base de données bien conçue garantit
 - ▶ Précision: peut-on se fier aux données lorsqu'elles sont insérées ou modifiées
 - ▶ Facile d'accès: les données sont elles organisées de telle façon à pouvoir y accéder, peut-on la maintenir d'une façon simple et rapide
 - ▶ Fiable: garantir l'intégrité des données, maintenir des données volumineuses et fiables
 - ▶ Flexible: permettre une mise à niveau afin de répondre aux nouveaux besoins



Introduction

- ▶ Les données sont des informations non organisées, qui sont traitées pour leur donner un sens
 - ▶ Faits réels, observations, perceptions
 - ▶ Symboles, caractères, nombres
 - ▶ Images, etc...
 - ▶ Ou un mix de ces types

Introduction

► Types de données

► Données structurées

- Schéma défini et une structure rigide (peuvent être représentées en lignes et en colonnes)
- Caractéristiques des bases de données relationnelles

► Données semi-structurées

- Possèdent certaines propriétés organisationnelles, mais ne peuvent pas être facilement stockées dans les lignes et les colonnes
- Organisées en hiérarchie à l'aide de balises et de métadonnées

► Données non structurées

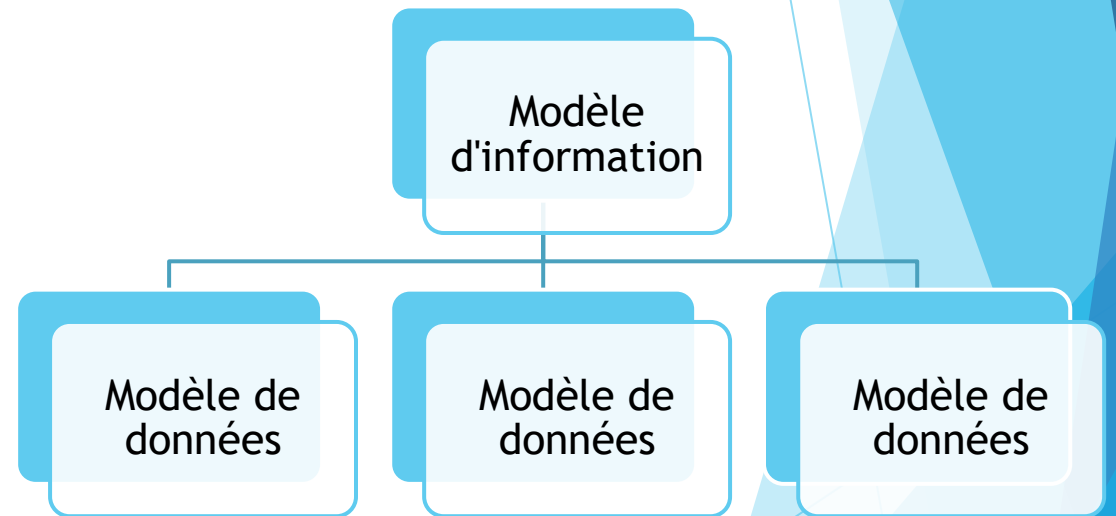
- Pas de structure identifiable: aucun format, séquence, sémantique ou règle spécifique
- Ne peuvent pas être organisées sous forme de tableau
- Souvent stockées dans des bases de données NoSQL

Introduction

- ▶ Sources de données:
 - ▶ Bases de données
 - ▶ Fichiers: (CSV, XLSX...), Sources de données XML, JSON
 - ▶ API et services Web
 - ▶ Web scraping
 - ▶ Plateformes des réseaux sociaux
 - ▶ Données collectées par les systèmes IOT
 - ▶ Etc...

Modèle d'Information et modèle de données

- ▶ Modèle d'information est une représentation abstraite et formelle d'entités qui inclut leurs propriétés, leurs relations et les opérations qui peuvent être effectuées sur elles
 - ▶ Définit les relations entre les entités
 - ▶ Entités modélisées peuvent provenir du monde réel
 - ▶ Attributs : caractéristiques des entités
- ▶ Modèle de données est plus spécifique et inclut les détails des entités modélisées pour l'implémentation



Modèles de bases de données

- ▶ Relationnel
- ▶ Hiérarchique
 - ▶ Créé dans les années 1960
 - ▶ Informations sont groupées dans des enregistrements et chaque enregistrement comporte des champs
 - ▶ Enregistrements sont reliés entre eux de manière hiérarchique : à chaque enregistrement correspond un enregistrement parent
- ▶ Réseau
 - ▶ Semblable au modèle hiérarchique
 - ▶ Informations sont groupées dans des enregistrements et chaque enregistrement possède des champs.
 - ▶ Enregistrements sont reliés entre eux par des pointeurs

Modèles de bases de données

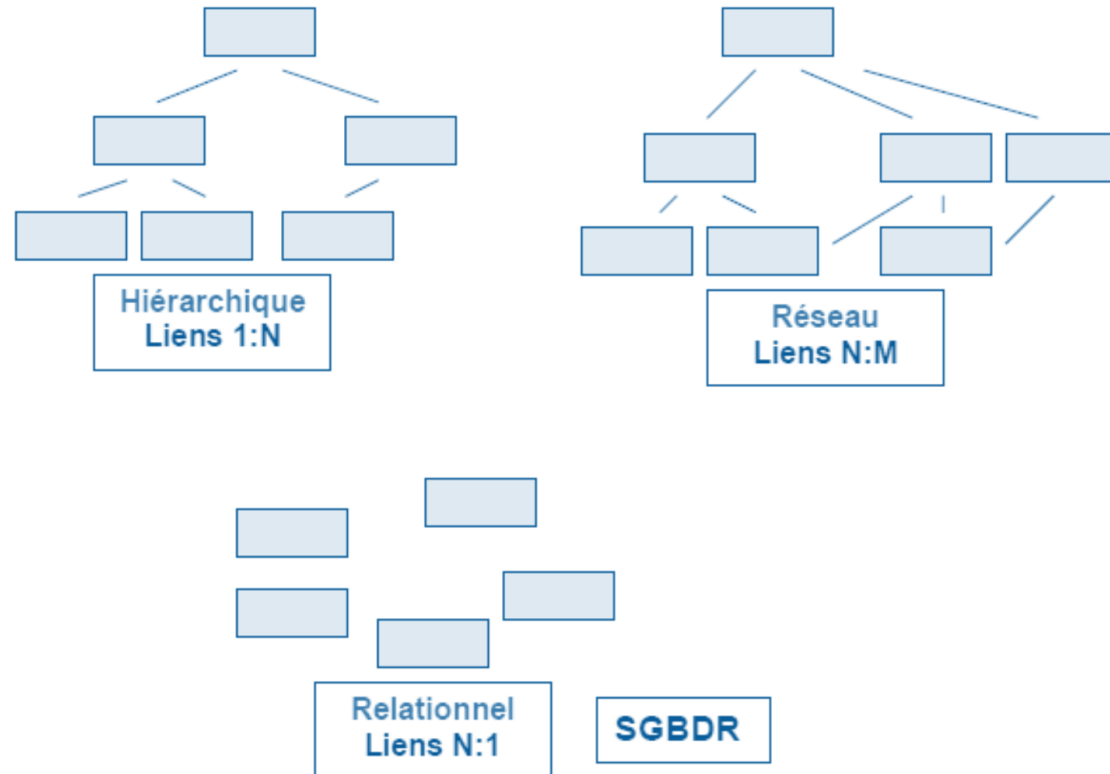
▶ Objet

- ▶ Fondé sur la notion d'objet de la programmation orientée objet
- ▶ Lot d'objets de différentes classes
- ▶ Chaque objet possède des propriétés et des méthodes

▶ No SQL

- ▶ Données brutes peu structurées
 - ▶ Cassandra, fondation Apache, utilisé par Twitter
 - ▶ MongoDB, MongoDB Inc., NoSQL orienté-document
 - ▶ HBase, fondation Apache, utilisé par Facebook
 - ▶ Neo4j, Neo Technology Inc, NoSQL orienté graphe

Modèles de bases de données



Typologie de bases de données

- ▶ Bases de données peuvent être aussi classées selon l'usage qui en est fait, et l'aspect temporel du contenu
 - ▶ Bases opérationnelles ou OLTP (online transaction processing)
 - ▶ Permettre aux utilisateurs de tenir l'état d'activités quotidiennes
 - ▶ Permettre aussi d'avoir la journalisation à chaque opération effectuée dans le cadre de l'activité
 - ▶ Vitesse de réponse et la capacité de traiter plusieurs opérations simultanément

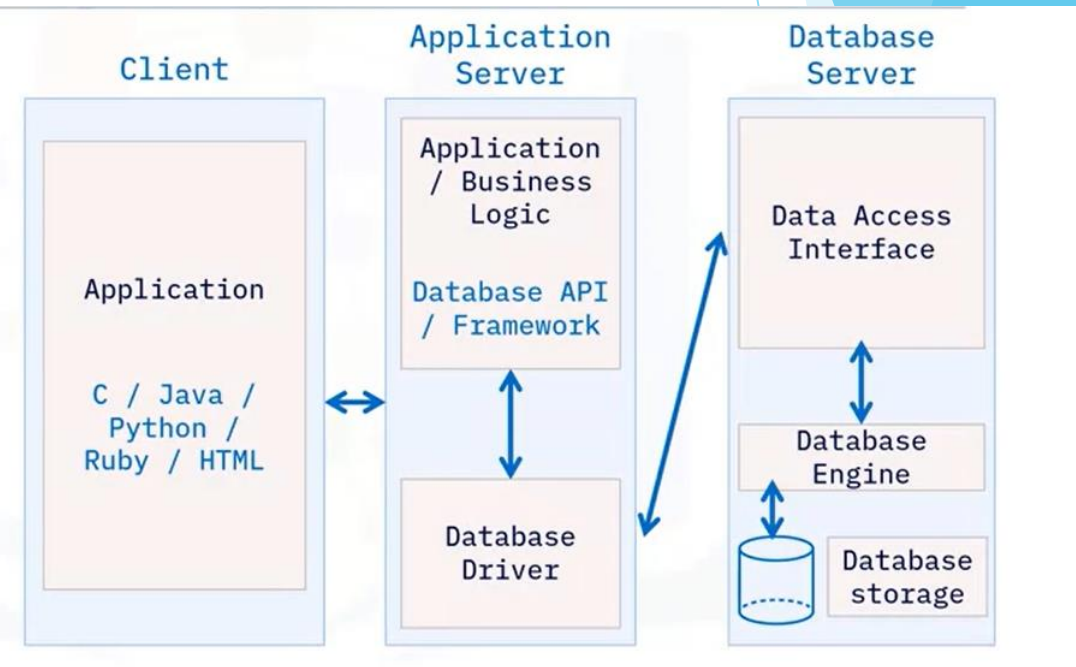
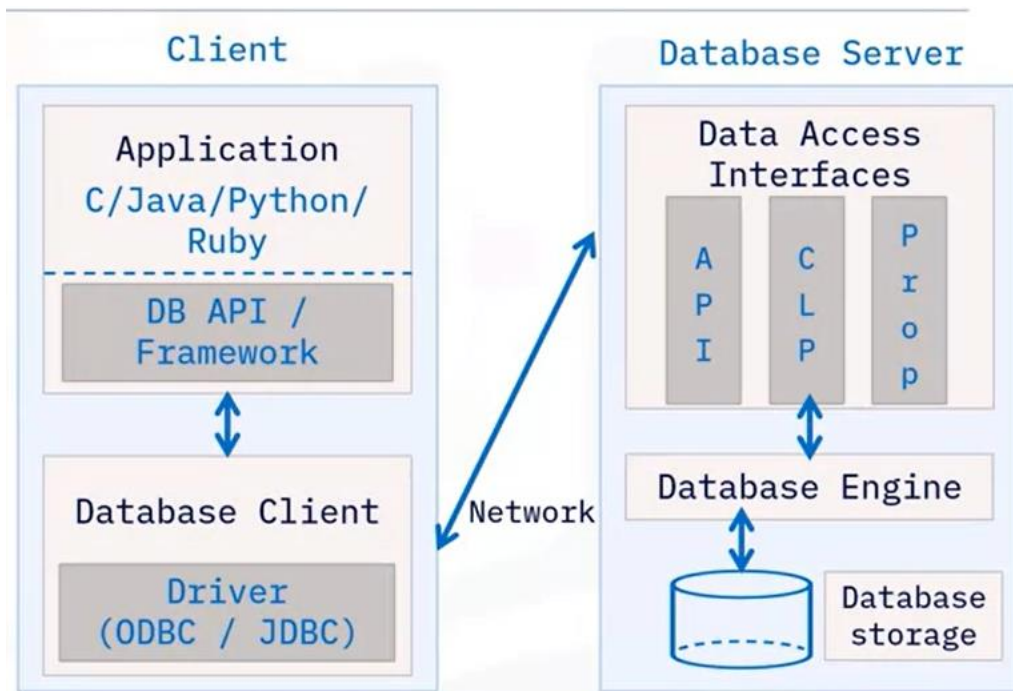
Typologie de bases de données

- ▶ Bases d'analyse ou OLAP (online analytical processing)
 - ▶ Regroupe des informations historiques sur lesquelles sont effectuées des opérations massives en vue d'obtenir des statistiques et des prévisions
 - ▶ Entrepôts de données (datawarehouse) utilisés pour collecter d'énormes quantités de données depuis une base de données opérationnelle
 - ▶ Optimisées pour effectuer des analyses d'évolution temporelle et des statistiques (projection des ventes, etc...)
 - ▶ Capacité d'effectuer des traitements très complexes

Architecture bases de données

- ▶ Topologie single-tier
 - ▶ Installée sur le poste local de l'utilisateur
- ▶ Topologie 2-tier
 - ▶ Utilisateurs accèdent à la base de données installée sur un serveur distant , à partir des applications-client
- ▶ Topologie 3-tier
 - ▶ Base de données installée sur un serveur distant , et les utilisateurs y accèdent à partir d'un serveur d'application
- ▶ Dans les environnements cloud, les utilisateurs accèdent à la base de données à partir d'un serveur d'application ou une interface cloud

Architecture bases de données

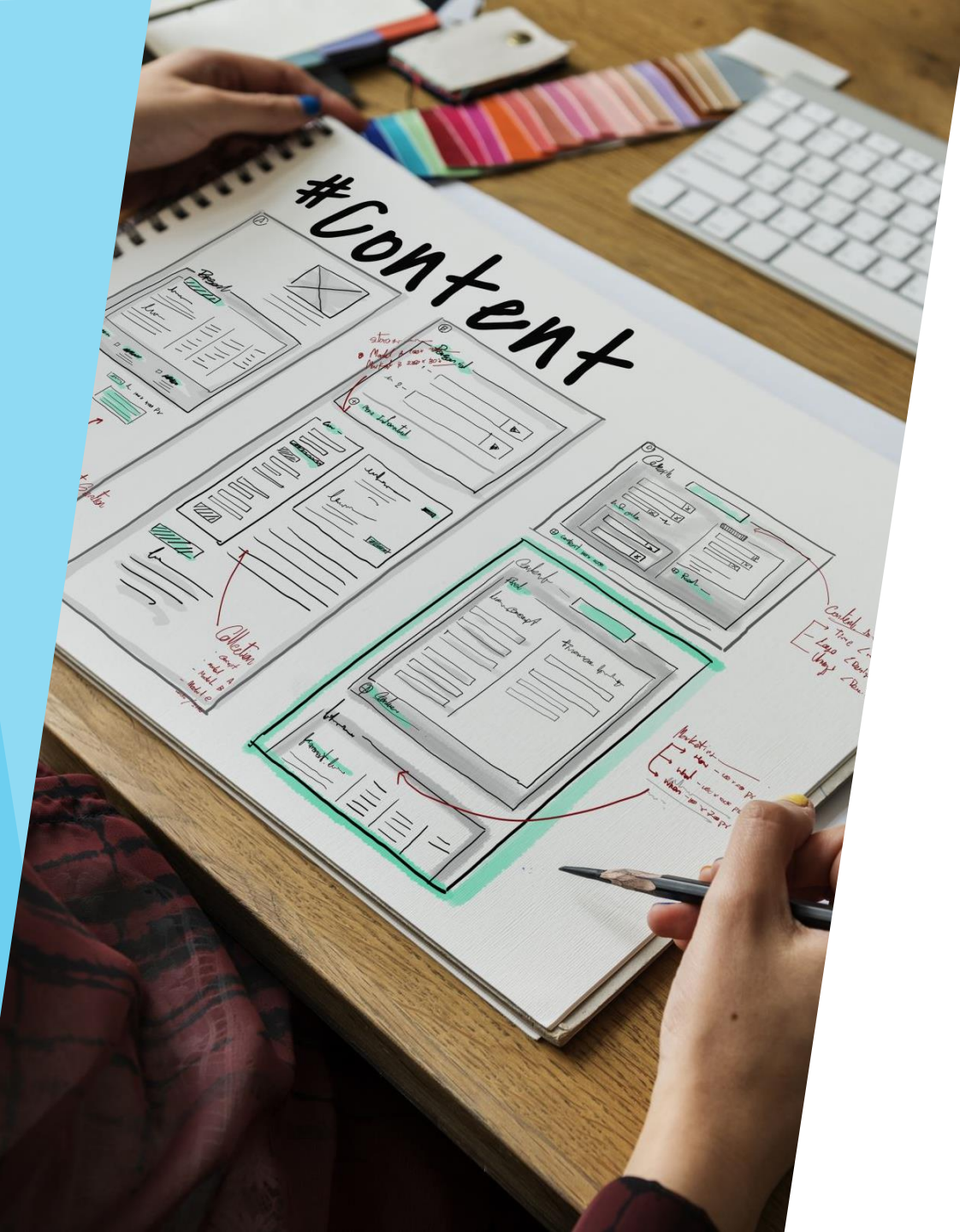


Profils d'utilisation des bases de données

- ▶ Ingénieurs de données / administrateurs des bases de données (DBA)
 - ▶ Création et gestion d'objets de base de données, définition de contrôles d'accès, surveillance et réglage des performances (Tuning)
 - ▶ Outils: GUI, interface web, ligne de commandes (commandes, scripts ou fichiers batch), API
- ▶ Data analysts, data scientists, analystes BI
 - ▶ Analyser les données, produire des statistiques et faire des prédictions
 - ▶ Outils
 - ▶ Outils D'analyse de données: Rstudio, SPSS, etc...
 - ▶ Outils BI: Excel, PowerBI, etc...
 - ▶ Interroger la base de données avec le langage SQL

Profils d'utilisation des bases de données

- ▶ Développeurs d'applications
 - ▶ Accès direct à la base de données en mode read/write
 - ▶ Outils: interface SQL et API (ODBC, JDBC et ORM)
 - ▶ ODBC (Open Database Connectivity): manipuler plusieurs types de bases de données (Microsoft)
 - ▶ JDBC (Java Database Connectivity): permet à des applications écrites pour la machine virtuelle Java de manipuler des bases de données (Sun Microsystems)
 - ▶ Framework ORM (Object Relational Mapping) : simple à utiliser pour accéder à la base, masque la complexité de la base de données du langage SQL
 - ▶ Django pour Python et Entity Framework pour .NET



Partie II: Modèle relationnel

Introduction

- ▶ **Modèle relationnel**
 - ▶ Développé par Codd en 1970 en se basant sur la théorie mathématique des relations
 - ▶ Modèle logique orienté enregistrement
 - ▶ Structure unique, la relation (tableau ou table)
 - ▶ Des contraintes qui définissent des formes normales afin d'éviter les défauts de conception
 - ▶ Des langages de requête concrétisés par le langage SQL (Structured Query Language)

Modèle relationnel

- ▶ Etant donné un ensemble d'objets O , une relation (binaire) sur O est un sous-ensemble du produit cartésien $O \times O$
 - ▶ Objets: valeurs élémentaires (I: entiers, F: flottants, S: chaînes de caractères, etc...)
Par exemple : Ensemble des paires constituées (noms de département, code) est une relation sur $S \times I$
 - ▶ Relation de degré n sur les domaines A_1, A_2, \dots, A_n est un sous-ensemble fini du produit cartésien $A_1 \times A_2 \times \dots \times A_n$
 - ▶ Élément d'une relation de dimension n est un nuplet (a_1, a_2, \dots, a_n)

Modèle relationnel

► Relation

- Nom de la relation
- Attribut: nom distinct pour chaque dimension
- Domaine de valeurs pour chaque dimension
- Schéma de la relation: $R(A1 : D1, A2 : D2, \dots, An : Dn)$
- Représenté par un tableau à deux dimensions
- Composé d'un nombre fini d'attributs
- Nuplets sont uniques

Exemple : Département (nom: string, code: string)

Département (nom, code)

Modèle relationnel

- ▶ Relation est composée de:
 - ▶ un schéma de relation spécifiant le nom d'une relation et les attributs
 - ▶ une instance de relation: un tableau composé des attributs et des nuplets
- ▶ Degré fait référence au nombre d'attributs (colonnes) dans une relation
- ▶ Cardinalité fait référence au nombre de nuplets (lignes) dans une relation
- ▶ Types de données définissent le type de données pouvant être stockées dans une colonne: chaînes de caractères, valeurs numériques, dates/heures, valeurs booléennes, etc.
 - ▶ Avantages: intégrité des données, tri des données, sélection de plage, calculs sur les données

Modèle relationnel

► Avantages

- Indépendance des données logiques, l'indépendance des données physiques et l'indépendance du stockage physique
- Systèmes SGBD peuvent utiliser des techniques de stockage (fichiers séquentiels, indexage, séries de pointeurs, compression...)
 - Relier ces structures à des relations/tables au niveau logique
 - Relations/tables représentent une abstraction de l'enregistrement physique des données
- Relations peuvent être un à un, un à plusieurs ou plusieurs à plusieurs
- Traduction d'un diagramme entité-relation (E-R) en une table de base de données relationnelle: l'entité devient la table et les attributs deviennent les colonnes de la table

Transformation des digrammes E-R en schéma relationnel

- ▶ Triplet pour le modèle E-R: Objets-Attributs-Valeurs
 - ▶ Objets ->Entités: Noms des tableaux
 - ▶ Attributs -> Attributs: Schémas des tableaux
 - ▶ Valeurs -> Valeurs : Instanciations des tableaux
- ▶ Règles de base
 - ▶ Chaque type d'entités devient une relation avec les mêmes attributs
 - ▶ Clé primaire d'une entité = clé primaire de la relation
 - ▶ Types d'entités faibles subissent la même transformation,
 - ▶ Ajouter une clé étrangère qui correspond à la clé primaire de l'entité forte
 - ▶ Chaque type de relations subit la transformation suivant la cardinalité

Transformation des digrammes E-R en schéma relationnel

- ▶ Relations statiques
 - ▶ Dimensions et contenus ont tendance à se stabiliser avec le temps
 - ▶ Données varient occasionnellement
 - ▶ Exemple : Employé, Client, Produit
- ▶ Relations dynamiques
 - ▶ Représentent les interactions entre les relations statiques
 - ▶ Exemple: Congé, Projet, Commande, Facture

Transformation des digrammes E-R en schéma relationnel

Terme du modèle	Terme de la représentation par table
Relation	Table
nuplet	Ligne
Nom d'attribut	Nom de colonne
Valeur d'attribut	Donnée d'une cellule
Domaine	Type de la donnée

Transformation des digrammes E-R en schéma relationnel

- ▶ Contraintes relationnelles à vérifier pour chaque instance du schéma relationnel
 - ▶ De domaine
 - ▶ De clés
 - ▶ D'intégrité
 - ▶ D'intégrité de référence
- ▶ Contrainte de domaine
 - ▶ Valeur de chaque attribut dans un nuplet est atomique (pas d'attributs composés ou multivalués)
 - ▶ Valeur doit respecter le format des données du domaine de l'attribut

Modèle relationnel

- ▶ Contraintes de clé
 - ▶ Chaque nuplet dans une relation doit être unique
 - ▶ Toute relation doit posséder une clé qui identifie un tuple de façon unique
 - ▶ Une relation peut posséder plusieurs clés candidates
 - ▶ Il se peut que l'ensemble de tous les attributs constitue une clé
 - ▶ La clé choisie est appelé clé primaire

Exemple: Département (nom, code)

Modèle relationnel

- ▶ Contraintes d'intégrité-entité
 - ▶ Aucune clé primaire ne doit être nulle
- ▶ Contraintes d'intégrité de référence
 - ▶ Contraintes spécifiées entre deux relations et utilisées pour maintenir la consistance entre les tuples de deux relations.
 - ▶ Concept de clé étrangère (Foreign Key)

Normalisation

- ▶ Afin de construire un schéma relationnel correct et éviter les problèmes de mise à jour: la théorie de la normalisation
- ▶ Série de règles pour garantir que la relation obéit à une certaine forme normale
- ▶ Basée sur les dépendances fonctionnelles entre les attributs d'une relation, afin de caractériser des relations pouvant être décomposées sans perte d'informations
- ▶ Codd a proposé trois formes normales
 - ▶ 1NF : une entité ou une association ne contient pas de propriétés répétitives ou décomposables
 - ▶ 2NF : tout identifiant non-clé dépend entièrement de la totalité de la clé
 - ▶ 3NF : tout attribut non-clé dépend entièrement de la totalité de la clé, et non d'un autre attribut non-clé

Normalisation

- ▶ D'autres formes normales ont été développées afin de traiter des cas particuliers non résolus par l'application successive des premières formes normales
 - ▶ BCNF (forme normale de Boyce-Codd): détection des informations redondantes ne doit pas se limiter aux attributs non clés, car les clés composées peuvent aussi être redondantes
 - ▶ 4NF (quatrième forme normale): dépendances multivaluées dans une table entraîne des redondances et des anomalies
 - ▶ 5NF: cinquième forme normale
 - ▶ PJNF: forme normale à projections jointives
 - ▶ DKN: forme normale à domaines -clés



Partie III: Langage SQL

Introduction

- ▶ Créé en 1970 chez IBM par Donald Chamberlin et Raymond Boyce
 - ▶ Nommé Structured English QUery Language (SEQUEL) et par la suite SQL
- ▶ Au milieu des années 1980, l'American National Standards Institute (ANSI) a lancé la première norme pour le langage SQL, qui a été publiée en 1986
- ▶ Des améliorations ultérieures ont conduit à de nouvelles versions de la norme en 1989, 1992, 1999, 2003, 2006, 2008, 2011 et 2016
- ▶ De nouvelles fonctionnalités ont été ajoutées au langage SQL pour intégrer des fonctionnalités orientées objet, etc...
 - ▶ Dernières normes se concentrent sur l'intégration des technologies telles que XML et la notation d'objet JavaScript(JSON)

Introduction

► Utilisation SQL

- Interactive SQL (ISQL): Instructions SQL transmises à partir d'un programme écrit dans un langage de programmation donné, à l'aide d'une interface du SGBD: ODBC, JDBC, etc...
- Embedded SQL (ESQL): instructions SQL incorporées dans le code source d'un programme écrit dans un langage de programmation donné
- Procédures stockées
 - Fonctions écrites dans un langage procédural propre à chaque SGBD et sont enregistrées dans la base de données en vue d'être réexécutées au besoin

```

using System;
using System.Data;
using System.Data.Odbc;

class Program
{
    static void Main()
    {
        // The connection string assumes that the Access
        // Northwind.mdb is located in the c:\Data folder.
        string connectionString =
            "Driver={Microsoft Access Driver (*.mdb)};"
            + "Dbq=c:\\Data\\Northwind.mdb;Uid=Admin;Pwd=";

        // Provide the query string with a parameter placeholder.
        string queryString =
            "SELECT ProductID, UnitPrice, ProductName from products "
            + "WHERE UnitPrice > ? "
            + "ORDER BY UnitPrice DESC;";

        // Specify the parameter value.
        int paramValue = 5;

        // Create and open the connection in a using block. This
        // ensures that all resources will be closed and disposed
        // when the code exits.
        using (OdbcConnection connection =
            new OdbcConnection(connectionString))
        {
            // Create the Command and Parameter objects.
            OdbcCommand command = new OdbcCommand(queryString, connection);
            command.Parameters.AddWithValue("@pricePoint", paramValue);

            // Open the connection in a try/catch block.
            // Create and execute the DataReader, writing the result
            // set to the console window.
            try
            {
                connection.Open();
                OdbcDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    Console.WriteLine("\t{0}\t{1}\t{2}",
                        reader[0], reader[1], reader[2]);
                }
                reader.Close();
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
            Console.ReadLine();
        }
    }
}

```

```
int main() {
    EXEC SQL INCLUDE SQLCA;
    EXEC SQL BEGIN DECLARE SECTION;
        int OrderID;          /* Employee ID (from user)      */
        int CustID;           /* Retrieved customer ID   */
        char SalesPerson[10]  /* Retrieved salesperson name */
        char Status[6]        /* Retrieved order status   */
    EXEC SQL END DECLARE SECTION;

    /* Set up error processing */
    EXEC SQL WHENEVER SQLERROR GOTO query_error;
    EXEC SQL WHENEVER NOT FOUND GOTO bad_number;

    /* Prompt the user for order number */
    printf ("Enter order number: ");
    scanf_s("%d", &OrderID);

    /* Execute the SQL query */
    EXEC SQL SELECT CustID, SalesPerson, Status
        FROM Orders
        WHERE OrderID = :OrderID
        INTO :CustID, :SalesPerson, :Status;

    /* Display the results */
    printf ("Customer number: %d\n", CustID);
    printf ("Salesperson: %s\n", SalesPerson);
    printf ("Status: %s\n", Status);
    exit();

query_error:
    printf ("SQL error: %ld\n", sqlca->sqlcode);
    exit();

bad_number:
    printf ("Invalid order number.\n");
    exit();
}
```

Introduction

- ▶ Langage SQL couvre 4 domaines
 - ▶ Langage de définition de données (LDD)
 - ▶ Langage de manipulation de données (LMD)
 - ▶ Langage de contrôle de données (LCD)
 - ▶ Langage de contrôle des transactions (LCT)
- ▶ Autres langages d'accès aux bases de données
 - ▶ Langage QBE (Query By Example)
 - ▶ SGBDR de type « fichier »: Paradox (Borland) ou Microsoft Access (Microsoft)

Modèle relationnel- rappel

- ▶ Etant donné un ensemble d'objets O , une relation (binaire) sur O est un sous-ensemble du produit cartésien $O \times O$
 - ▶ Objets: valeurs élémentaires (I: entiers, F: flottants, S: chaînes de caractères, etc...)
Par exemple : Ensemble des paires constituées (noms de département, code) est une relation sur $S \times I$
 - ▶ Relation de degré n sur les domaines A_1, A_2, \dots, A_n est un sous-ensemble fini du produit cartésien $A_1 \times A_2 \times \dots \times A_n$
 - ▶ Élément d'une relation de dimension n est un nuplet (a_1, a_2, \dots, a_n)

Modèle relationnel- rappel

- ▶ Modèle relationnel
 - ▶ Structuration des données
 - ▶ Base de données relationnelle : un ensemble de tables associées les unes aux autres
 - ▶ Conception du schéma relationnel doit obéir à certaines règles et satisfaire certaines propriétés
 - ▶ Normalisation permet de s'assurer que l'on a construit un schéma correct

Modèle relationnel- rappel

- ▶ Langages d'interrogation
 - ▶ Construire des expressions (requêtes) qui s'appuient sur une base de données en entrée et fournissent une table en sortie
 - ▶ Deux langages d'interrogation équivalents et complémentaires
 - ▶ Déclaratif (basé sur la logique mathématique): on formule les résultats à obtenir et le système décide comment les calculer
 - ▶ Procédural (algèbre basé sur la théorie des ensembles): identifie l'ensemble minimal des opérateurs dont le système doit disposer pour évaluer une requête

Langage déclaratif: SQL

- ▶ Langage SQL: syntaxe pratique, intuitive et naturelle pour le langage relationnel déclaratif
- ▶ Langage SQL exprime des requêtes comme des formules que doivent satisfaire les nuplets résultat
- ▶ Langage SQL illustre des caractéristique d'indépendance logique / physique
 - ▶ Aucune référence à la méthode qui permet de calculer le résultat des requêtes
 - ▶ Plusieurs méthodes pour calculer le résultat existent et le SGBD choisit la meilleure méthode en fonction de l'organisation physique des données

Algèbre relationnelle

- ▶ Algèbre se compose d'un ensemble d'opérateurs
 - ▶ On peut exprimer toutes les requêtes en algèbre
 - ▶ Il existe une syntaxe SQL pour toutes les requêtes algébriques
- ▶ Notion de clôture : toute opération s'applique à des relations et produit une relation
- ▶ Notion de composition : on crée des requêtes complexes en combinant des opérateurs
- ▶ Opérateurs sont nécessaires et suffisants et permettent de définir les autres par composition
 - ▶ Sélection : σ
 - ▶ Projection: π
 - ▶ Renommage: ρ
 - ▶ Produit cartésien: \times
 - ▶ Union: \cup
 - ▶ Différence: $-$

Algèbre relationnelle

- ▶ Opérateurs unaires: Sélection et Projection
 - ▶ Produire une nouvelle table à partir d'une autre table
- ▶ Opérateurs binaires ensemblistes: Union et Différence
 - ▶ Produire une nouvelle relation à partir de deux relations de même degré et de même domaine
- ▶ Opérateurs binaires ou n-aires: Produit cartésien et Jointure

Algèbre relationnelle

► Sélection

- Générer une relation regroupant exclusivement toutes les occurrences de la relation R qui satisfont l'expression logique E

$$\sigma(E)R$$

relation \times expression logique \rightarrow relation

- Sélectionner des occurrences de la relation et le résultat est une nouvelle relation qui a les mêmes attributs que R

- Exemple: $\sigma(\text{NuméroDépartement} \geq 5)\text{Département}$

select * from Département where NuméroDépartement \geq 5

Algèbre relationnelle

► Projection

- Supprimer les attributs autres que A_1, \dots, A_n d'une relation

$$\Pi(A_1 \dots A_n)R$$

relation \times liste d'attributs \rightarrow relation

- Sélectionner des colonnes de la table
- Si R est vide, la relation qui résulte de la projection est vide
- Exemple: $\Pi(\text{NomDépartement}) \text{ Département}$

`select NomDépartement from Département`

Algèbre relationnelle

► Union

- Porte sur deux relations R1 et R2 ayant le même schéma et construisant une troisième relation constituée des n-uplets appartenant à chacune des deux relations R1 et R2 sans doublons

$$R1 \cup R2$$

relation \times relation \rightarrow relation

- R1 et R2 doivent avoir les mêmes attributs
- Si une même occurrence existe dans R1 et R2, elle n'apparaît qu'une seule fois dans le résultat de l'union
- Exemple: DépartementR1 \cup DépartementR2

```
select * from DépartementR1 union select * from DépartementR2
```

Algèbre relationnelle

► Différence

- Porte sur deux relations R1 et R2 ayant le même schéma et construisant une troisième relation dont les n-uplets sont constitués de ceux de la relation R1

$$R1 - R2$$

$$\text{relation} \times \text{relation} \rightarrow \text{relation}$$

- R1 et R2 doivent avoir les mêmes attributs
- Résultat de la différence est une nouvelle relation qui a les mêmes attributs que R1 et R2
- Exemple: DépartementR1 - DépartementR2

`select ID from DépartementR1 except select ID from DépartementR2`

Algèbre relationnelle

▶ Produit cartésien

- ▶ Porte sur deux relations R1 et R2 et qui construit une troisième relation regroupant exclusivement toutes les possibilités de combinaison des occurrences des relations R1 et R2

$$R1 \times R2$$

relation \times relation \rightarrow relation

- ▶ Résultat du produit cartésien est une nouvelle relation qui a tous les attributs de R1 et tous ceux de R2
- ▶ Nombre d'occurrences de la relation résultat est le nombre d'occurrences de R1 multiplié par le nombre d'occurrences de R2
- ▶ Exemple: DépartementR1 \times DépartementR2

```
select * from DépartementR1 cross join DépartementR2
```

```
select * from DépartementR1,DépartementR2
```


Algèbre relationnelle

► Renommage

- Peut s'appliquer soit à la relation, soit aux attributs ou aux deux à la fois

$\rho(R1)$

$R1 \times \rho A \rightarrow C, B \rightarrow D(R2)$

► Cas du produit cartésien pour les relations R1 et R2

- Quand les schémas des relations R1 et R2 sont complètement distincts, il n'y a pas d'ambiguïté
- Quand les deux relations ont des attributs qui portent le même nom
- Distinguer l'origine des colonnes dans la relation résultat en donnant un nom distinct à chaque attribut

► Exemple: renommer les attributs a,b des relations R1,R2

```
select R1.a as premier_a, R1.b as premier_b, R2.a as second_a,  
R2.b as second_b from R1, R2
```

Algèbre relationnelle

► Jointure

- Porte sur deux relations R1 et R2 qui construit une troisième relation regroupant exclusivement toutes les possibilités de combinaison des occurrences des relations R1 et R2 qui satisfont l'expression logique E

$$R1 \triangleright \triangleleft E R2$$

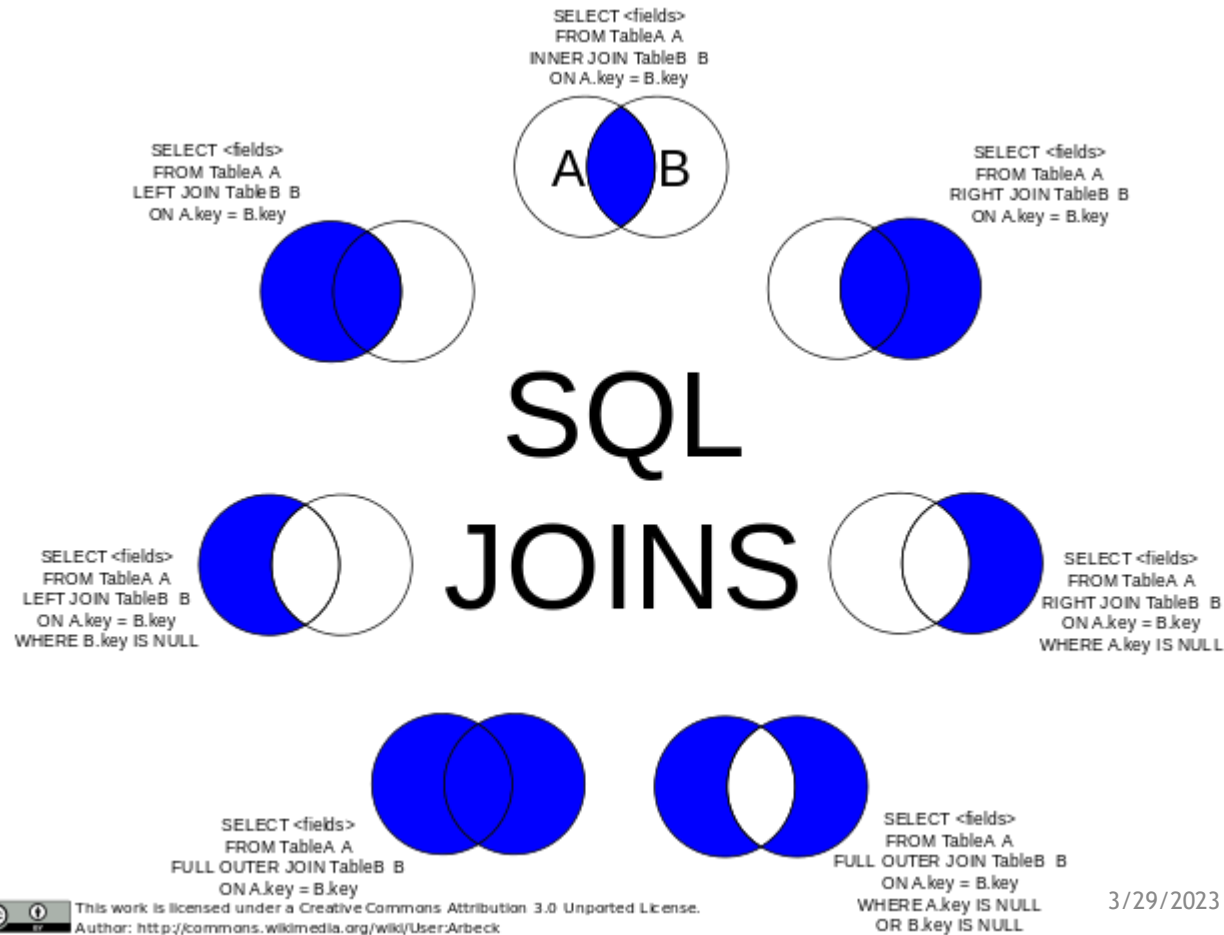
relation \times relation \times expression logique \rightarrow relation

- Si R1 ou R2 ou les deux sont vides, la relation qui résulte de la jointure est vide
- Jointure \equiv produit cartésien suivi d'une sélection

$$R1 \triangleright \triangleleft E R2 \equiv \sigma(E) R1 \times R2$$

select * from R1, R2 where R1.A1 = R2.A2

Jointures SQL



Jointures SQL

- ▶ CROSS JOIN : jointure croisée permettant de faire le produit cartésien de 2 tables
- ▶ INNER JOIN : jointure interne pour retourner les enregistrements quand la condition est vrai dans les 2 tables. C'est l'une des jointures les plus communes
- ▶ LEFT JOIN (ou LEFT OUTER JOIN) : jointure externe pour retourner tous les enregistrements de la table de gauche même si la condition n'est pas vérifié dans l'autre table
- ▶ RIGHT JOIN (ou RIGHT OUTER JOIN) : jointure externe pour retourner tous les enregistrements de la table de droite même si la condition n'est pas vérifié dans l'autre table
- ▶ FULL JOIN (ou FULL OUTER JOIN) : jointure externe pour retourner les résultats quand la condition est vrai dans au moins une des 2 tables
- ▶ SELF JOIN : permet d'effectuer une jointure d'une table avec elle-même
- ▶ NATURAL JOIN : jointure naturelle entre 2 tables s'il y a au moins une colonne qui porte le même nom dans les 2 tables

Jointures SQL

► Table A: commandes et Table B: clients

- Commandes faites par des clients => commandes **INNER JOIN** clients **ON** comamndes.id_client = clients.id_client
- Commandes des clients même celles qui n'ont pas de clients => commandes **LEFT JOIN** clients **ON** comamndes.id_client = clients.id_client
- Clients mêmes ceux qui n'ont pas de commandes => commandes **RIGHT JOIN** clients **ON** comamndes.id_client = clients.id_client
- Clients qui n'ont pas de commandes => commandes **RIGHT JOIN** clients **ON** comamndes.id_client = clients.id_client **WHERE** comamnde.id_client **IS NULL**

- Commandes qui n'ont pas de clients => commandes **LEFT JOIN** clients **ON** comamndes.id_client = clients.id_client **WHERE** clients.id_client **IS NULL**
- Toutes les commandes qui n'ont pas clients et tous les clients qui n'ont pas de commandes => commandes **FULL OUTER JOIN** **ON** comamndes.id_client = clients.id_client
- Toutes les commandes et tous les clients => commandes **FULL OUTER JOIN** **ON** comamndes.id_client = clients.id_client **WHERE** comamndes.id_client **IS NULL OR** id_client **IS NULL**

Clause Select

```
SELECT nom_table.nom_colonne*  
FROM nom_table*  
[WHERE conditions_de_sélection_sur_lignes*]  
[GROUP by nom_colonne_de_regroupement*]  
[HAVING conditions_de_sélection_sur_groupe*]  
[ORDER BY nom_colonne_tri*] ;
```

*: plusieurs occurrences possibles, []: optionnel

Clause Select

- ▶ Données extraites et données dérivées
 - ▶ Données extraites avec la clause Select proviennent directement de la base de données (noms des colonnes des tables)
 - ▶ Cependant, la clause select peut spécifier des données dérivées, ou même des constantes

Clause Select

- ▶ Exemple : construire un tableau pour les montants TVA des articles en stock dont la quantité restante est supérieure à 500 unités

Select ' TVA de ', NPRO, ' = ', 0.21*PRIX*QSTOCK from PRODUIT where QSTOCK > 500

TVA de	NPRO	=	0,21*PRIX*QSTOCK
TVA de	CS264	=	67788
TVA de	PA45	=	12789
TVA de	PH222	=	37770,6
TVA de	PS222	=	47397

Clause Select

- ▶ DISTINCT: permet d'éviter des redondances dans les résultats

`SELECT DISTINCT colonne1 FROM table1`

- ▶ IS NULL: permet de filtrer les résultats qui contiennent la valeur NUL

`SELECT * FROM table1 WHERE colonne1 IS NULL`

- ▶ IS NOT NULL: permet d'obtenir uniquement les enregistrements qui ne sont pas null

`SELECT * FROM table1 WHERE colonne1 IS NOT NULL`

Clause Select

- ▶ **SELECT INTO:** permet de copier les données d'une table vers une nouvelle table (dans la même base de données ou une autre base de données)

```
SELECT *  
INTO NouvelleTable [IN BasedeDonnéesExterne]  
FROM AncienneTable  
WHERE condition
```

Expression des unions

- ▶ Clause UNION: permet de concaténer les résultats de plusieurs requêtes
- ▶ Chaque requête à concaténer retourne le même nombre de colonnes avec les mêmes types de données et dans le même ordre

```
SELECT * FROM table1  
UNION  
SELECT * FROM table2
```

Expression des unions

- ▶ Clause UNION ALL: similaire à la clause UNION, sauf qu'elle permet d'inclure tous les enregistrements, même les doublons

```
SELECT * FROM table1  
UNION ALL  
SELECT * FROM table2
```

Types de données SQL

Type	Description	Taille
integer	Type des entiers relatifs	4 octets
smallint	idem	2 octets
bigint	idem	8 octets
float	Flottants simple précision	4 octets
double	Flottants double précision	8 octets
real	Flottant simple ou double	8 octets
numeric (M, D)	Numérique avec précision fixe.	M octets
decimal(M, D)	Idem.	M octets
char(M)	Chaînes de longueur fixe	M octets
varchar*(M*)	Chaînes de longueur variable	L+1 avec $L \leq M \leq M$
bit varying	Chaînes d'octets	Longueur de la chaîne.
date	Date (jour, mois, an)	env. 4 octets
time	Horaire (heure, minutes, secondes)	env. 4 octets
datetime	Date et heure	8 octets
year	Année	2 octets

Opérateurs SQL

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo

Opérateur	Description
&	AND
	OR
^	OR exclusif

Opérateurs SQL

Opérateur	Description
=	Egal à
>	Supérieur à
<	Inférieur
>=	Supérieur ou égal
<=	Inférieur ou égal
<>	Différent

Fonctions SQL

- ▶ Fonctions mathématiques / numérique

- ▶ RAND()
- ▶ ROUND()
- ▶ TRUNC ()
- ▶ FLOOR()
- ▶ CEIL()

- ▶ Fonctions dates et heure

- ▶ DATE_FORMAT()
- ▶ DATEDIFF()
- ▶ DAYOFWEEK()
- ▶ MONTH()
- ▶ NOW()

- ▶ SEC_TO_TIME()

- ▶ TIMEDIFF()
- ▶ TIMESTAMP()
- ▶ YEAR()

- ▶ Fonctions de chiffrement

- ▶ MD5()

- ▶ Fonctions de conversion entre types

- ▶ TO_CHAR()
- ▶ TO_NUMBER()

Fonctions SQL

▶ Traitement de chaîne de caractères

- ▶ CONCAT()
- ▶ LENGTH()
- ▶ REPLACE()
- ▶ SOUNDEX()
- ▶ SUBSTRING()
- ▶ LEFT()
- ▶ RIGHT()
- ▶ REVERSE()
- ▶ TRIM()

- ▶ LTRIM()
- ▶ RTRIM()
- ▶ LPAD()
- ▶ UPPER()
- ▶ LOWER()
- ▶ UCASE()
- ▶ LCASE()
- ▶ LOCATE()
- ▶ INSTR()

Requêtes composées

- ▶ Requêtes d'agrégation
- ▶ Requêtes imbriquées / corrélées
- ▶ Requêtes avec quantificateurs ensemblistes

Requêtes d'agrégation

- ▶ Requêtes d'agrégation consistent à effectuer des regroupements de nuplets en fonction des valeurs d'une ou plusieurs expressions
 - ▶ Clause GROUP BY
 - ▶ Partitionner le résultat d'un bloc « select from where » en fonction d'un critère (un ou plusieurs attributs ou encore une expression sur des attributs)
 - ▶ Utilisation de fonctions d'agrégation pour les attributs qui n'apparaissent pas dans le group by est obligatoire
 - ▶ Clause HAVING
 - ▶ Spécifier des conditions sur le résultat de fonctions d'agrégation appliquées à des groupes de nuplets

Requêtes d'agrégation

- ▶ Fonctions d'agrégation
 - ▶ `count(expression)`: nombre de nuplets pour lesquels `expression` est not null
 - ▶ `Avg(expression)`: calcule la moyenne de `expression`
 - ▶ `min(expression)`: calcule la valeur minimale de `expression`
 - ▶ `max(expression)`: calcule la valeur maximale de `expression`
 - ▶ `sum(expression)`: calcule la somme de `expression`
 - ▶ `std(expression)`: calcule l'écart-type de `expression`
 - ▶ `Variance (expression)`: calcule la variance de `expression`

Requêtes d'agrégation

Immeuble (id, nom, adresse)

Appart (id , no , surface , niveau , idImmeuble)

Propriétaire (idPropriétaire , idAppart, pourcentage)

Requêtes d'agrégation

```
select idImmeuble, min(niveau) as minEtage, max(niveau) as maxEtage,  
sum(surface) as totalSurface from Appart GROUP BY idImmeuble
```

```
select idAppart, count(*) as nbProprios from Propriétaire GROUP BY idAppart  
HAVING count(*) >= 2
```

Requêtes et sous-requêtes

- ▶ Requêtes imbriquées ou en cascade
 - ▶ Offrir une alternative syntaxique à l'expression de certains types de jointures
 - ▶ Résultat est constitué avec les attributs provenant d'une seule des deux relations, l'autre ne servant que pour exprimer des conditions
 - ▶ Clause In : filtrer les lignes qui possèdent une des valeurs retournées par la sous-requête

Requêtes et sous-requêtes

```
select surface, niveau from Apart where id IN (select idAppart  
from Personne where nom='TOTO')
```

```
select * from Immeuble where id IN (select idImmeuble from Appart  
where surface=50)
```


Requêtes et sous-requêtes

▶ Requêtes corrélées

- ▶ requêtes exécutées en deux phases: requêtes imbriquées évaluées indépendamment de la requête principale
- ▶ Clause EXISTS: exprimer les requêtes en basant la sous-requête sur une ou plusieurs valeurs issues de la requête principale

```
select prénom, nom, profession from Personne  
where idAppart IN (select id from Appart where surface >= 70)
```

≡

```
select prénom, nom, profession from Personne p  
where EXISTS (select * from Appart a  
              where a.id=p.idAppart and surface >= 70)
```

Requêtes et sous-requêtes

- ▶ Requêtes avec négation
 - ▶ Clause NOT IN ou NOT EXISTS: exprimer des négations

```
select * from Appart  
where id NOT IN (select idAppart from Personne)
```

```
select * from Appart  
where id NOT EXISTS (select idAppart from Personne)
```

Requêtes et sous-requêtes

- ▶ Requêtes avec des quantificateurs ensemblistes
 - ▶ Comparer une expression de valeurs à tous les résultats d'une sous-question ou seulement à l'une quelconque des valeurs générées
 - ▶ Clause ALL: permet de comparer une valeur dans l'ensemble de valeurs d'une sous-requête en appliquant certains opérateurs
 - ▶ Opérateurs conditionnels : =, <, >, <>, !=, <=, >=, != ou !=
 - ▶ Un prédicat quantifié par ALL est vrai s'il est vérifié pour tous les éléments de l'ensemble

```
SELECT * FROM table1
```

```
WHERE condition > ALL ( SELECT * FROM table2 WHERE condition2 )
```

Requêtes et sous-requêtes

- ▶ Requêtes avec des quantificateurs ensemblistes
 - ▶ Clause ANY/SOME: permet de comparer une valeur avec le résultat d'une sous-requête en appliquant certains opérateurs
 - ▶ Opérateurs conditionnels : =, <, >, <>, !=, <=, >=, != ou !=
 - ▶ Un prédicat quantifié par ANY ou SOME est vrai s'il est vérifié par au moins un élément de l'ensemble

```
SELECT * FROM table1
```

```
WHERE condition > ANY ( SELECT * FROM table2 WHERE condition2 )
```

```
SELECT * FROM table1
```

```
WHERE condition > SOME ( SELECT * FROM table2 WHERE condition2 )
```



Partie IV: SQL avancé

Introduction

- ▶ Environnement des bases de données
 - ▶ Objets : tables, vues etc... créés par les utilisateurs
 - ▶ Dictionnaire des données ou le catalogue du système: collection de tables dont le contenu décrit les structures de cette base de données

Catalogue

- ▶ Catalogue
 - ▶ Méta-données ou le dictionnaire des données ou encore information de schéma
 - ▶ Tables: renseignent sur les tables des utilisateurs, leurs colonnes, les index, les privilèges, les utilisateurs/les programmes qui utilisent la base de données (connexions)
 - ▶ Nombre, noms , structures et interprétation des tables du catalogue varient selon les SGBD et leurs versions

Catalogue

- ▶ Norme SQL précise 23 vues standards afin de connaître les éléments constituant l'architecture de données du CATALOG du SGBDR
- ▶ Deux tables importantes
 - ▶ Tables des tables: renseigne sur les tables qui constituent la base de données
 - ▶ Table des colonnes: décrit les colonnes des tables de la base de données
- ▶ Cependant, certains SGBDR ne proposent pas ces vues standards pour accéder aux méta données
 - ▶ ORACLE : USER_CATALOG, USER_TABLES, ALL_TABLES, USER_SYNONYMS...
 - ▶ MS SQL SERVER : SYSDATABASES, SYSOBJECTS, SYSFOREIGNKEYS, SYSREFERENCES...
 - ▶ PostgreSQL: pg_catalog.pg_class, pg_catalog.pg_attribute, pg_catalog.pg_proc, pg_catalog.pg_user, pg_catalog.pg_views

Catalogue

Élément du SGBDR	SQL_LANGUAGES	liste des langages supportés au niveau SQL API
Élément du CATALOG	SCHEMATA	liste des bases
Éléments d'une base	DOMAINS	liste des domaines de la base
	TABLES	liste des tables de la base
	VIEWS	liste des vues de la base
	ASSERTIONS	liste des contraintes de la base
	CHARACTER_SETS	liste des jeux de caractères de la base
	COLLATIONS	liste des collations (schémas d'équivalence de caractères) de la base
	TRANSLATIONS	liste des "translations" (schémas de remplacement de caractères) de la base
Éléments d'une table	COLUMNS	liste des colonnes de TOUTES les tables de la base
	TABLE_CONSTRAINTS	liste des contraintes des tables de la base
	REFERENTIAL_CONSTRAINTS	liste des intégrités référentielles de la base
	CHECK_CONSTRAINTS	liste des contraintes de validité de la base
	KEY_COLUMN_USAGE	liste des colonnes définissant les clefs (primaire ou étrangère) de la base
	CONSTRAINT_COLUMN_USAGE	liste des colonnes définissant les contraintes de la base
	CONSTRAINT_TABLE_USAGE	liste des tables utilisée par les contraintes de la base
Éléments d'une vue	VIEW_TABLE_USAGE	liste des tables composant les vues de la base
	VIEW_COLUMN_USAGE	liste des colonnes composant les vues de la base
Éléments d'un domaine	DOMAIN_CONSTRAINT	liste des contraintes des domaines de la base
	DOMAIN_COLUMN_USAGE	liste des colonnes basées sur les domaines de la base
Privilèges	TABLE_PRIVILEGES	liste des privilèges des tables de la base
	COLUMN_PRIVILEGES	liste des privilèges de colonnes de table de la base
	USAGE_PRIVILEGES	liste des privilèges des autres objets de la base

Architecture des données du Catalogue d'un SGBD selon la norme SQL2

Catalogue

SYS_TABLE		
TNAME	CREATOR	TTYPE
SYS_TABLE	SYSTEM	R
SYS_COLUMN	SYSTEM	R
SYS_KEY	SYSTEM	R
SYS_KEY_COMP	SYSTEM	R
....	-
CLIENT	AGF	R
COMMANDE	AGF	R
DETAIL	AGF	R
PRODUIT	FDE	R
VAL_STOCK	AFB	V
....	-

SYS_COLUMN					
TNAME	CNAME	CTYPE	LEN1	LEN2	NULLS
SYS_TABLE	TNAME	varchar	18		N
SYS_TABLE	CREATOR	char	8		N
....	-
CLIENT	ADRESSE	char	60		N
CLIENT	CAT	char	2		Y
CLIENT	COMPTE	decimal	9	2	N
CLIENT	LOCALITE	char	30		N
CLIENT	NCLI	char	10		N
CLIENT	NOM	char	32		N
COMMANDE	DATECOM	date			N
COMMANDE	NCLI	char	10		N
COMMANDE	NCOM	char	12		N
DETAIL	NCOM	char	12		N
DETAIL	NPRO	char	15		N
DETAIL	QCOM	decimal	8	0	N
PRODUIT	LIBELLE	char	60		N
PRODUIT	NPRO	char	15		N
PRODUIT	PRIX	decimal	6	0	N
PRODUIT	QSTOCK	decimal	8	0	N
VAL_STOCK	STOCK	date			N
VAL_STOCK	VALEUR	decimal	12	0	N
....	-

Exemple de tables du catalogue: table des tables (SYS_TABLE) et table des colonnes (SYS_COLUMN)

Catalogue

- ▶ Utilisation du catalogue par le SGBD
 - ▶ Tables du catalogue sont modifiées par le SGBD lors de l'exécution d'une requête: create table, create index, drop table, alter table, create view, grant, revoke, etc...
 - ▶ Vérifier la validité des requêtes, les traduire en algorithmes efficaces et les exécuter
 - ▶ Vérifier que la structure de la base de données n'a pas été modifiée depuis la dernière compilation des procédures stockées, etc...
- ▶ Utilisation du catalogue par les utilisateurs de la Base de données
 - ▶ Tables du catalogue peuvent être consultées par les utilisateurs qui ont le droit

Catalogue

- Quelles sont les colonnes de la table DETAIL ?

```
select  CNAME, CTYPE, LEN1, NULLS
from    SYS_COLUMN
where   TNAME = 'DETAIL'
```

CNAME	CTYPE	LEN1	NULLS
NCOM	CHAR	12	N
NPRO	CHAR	15	N
QCOM	DECIMAL	8	N

- Dans quelles tables de base existe-t-il des colonnes dont le nom commence par 'NCOM' ?

```
select TNAME
from   SYS_TABLE
where  TNAME in (select TNAME
                  from   SYS_COLUMN
                  where   CNAME like 'NCOM%')

and    TTYPE = 'R'
```

TNAME
COMMANDE
DETAIL

Vues

- ▶ Vue
 - ▶ Schéma de table calculée qui est le résultat de requêtes stockées
 - ▶ Interroger et faire des mises à jour des vues comme des tables « réelles »
- ▶ Avantages
 - ▶ Aucun stockage n'est requis, car la vue n'existe pas physiquement (pas tous les SGBD)
 - ▶ Obtenir une représentation différente des tables sur lesquelles la vue est basée
 - ▶ Fournir des requêtes prédéfinies afin de faciliter l'interrogation des données
 - ▶ Masquer certaines informations aux utilisateurs
 - ▶ Protéger un utilisateur de l'effet de modifications de la structure de la base de données

Vues

```
CREATE OR REPLACE VIEW view_name AS  
SELECT column1, column2, ...  
FROM table_name  
WHERE condition  
[WITH CHECK OPTION]
```

[WITH CHECK OPTION]: imposer des contraintes d'intégrité afin de s'assurer que les lignes insérées vérifient les conditions de sélection

Index

- ▶ Pourquoi créer un Index ?
 - ▶ Offre un chemin d'accès rapide aux enregistrements d'une table par rapport au parcours des données de cette table
 - ▶ Permet de vérifier rapidement les contraintes de clé primaire et d'intégrité référentielle
 - ▶ Permet d'améliorer le temps de réponse pour les requêtes de jointure
 - ▶ Peut être créée sur un ou plusieurs colonnes, afin d'être utilisés dans les requêtes avec des critères de recherche qui portent sur des colonnes autres que les clés primaire ou étrangères
- ▶ Cependant, les index ont un impact négatif sur les requêtes d'insertion et de suppression, étant donné que les index de la table en question doivent être aussi mis à jour

Index

```
CREATE [UNIQUE] INDEX <NOMINDEX> ON <NOMTABLE> (<ATTRIBUT1> [, ...])
```

```
CREATE UNIQUE INDEX IDXNOM ON CLIENT (NOM, PRÉNOM)
```

```
CREATE UNIQUE INDEX IDXCAT ON CLIENT (CATEGORIE)
```


Procédures stockées

- ▶ Procédures stockées illustrent l'intégration du langage SQL à un langage de programmation procédural
- ▶ Procédure stockée: séquence d'instructions SQL précompilées dont l'exécution peut être demandée par un utilisateur, un programme d'application, un trigger ou une autre procédure
- ▶ Avantages
 - ▶ Ressource stockée dans la base de données et partagée pour les différents utilisateurs
 - ▶ Définir des comportements complexes, en particulier l'intégrité des données

Procédures stockées

- ▶ Langages de programmation procédurale dans lesquels peuvent être ajoutées des instructions en langage SQL
 - ▶ PL/SQL: Oracle
 - ▶ PL/pgSQL : PostgreSQL
 - ▶ SQL/PSM : MySQL
 - ▶ Tansact-SQL : Microsoft

Procédures stockées

```
DECLARE
  CURSOR c1 is
    SELECT ename, empno, sal FROM emp
      ORDER BY sal DESC;  -- start with highest paid employee
  my_ename VARCHAR2(10);
  my_empno  NUMBER(4);
  my_sal    NUMBER(7,2);
BEGIN
  OPEN c1;
  FOR i IN 1..5 LOOP
    FETCH c1 INTO my_ename, my_empno, my_sal;
    EXIT WHEN c1%NOTFOUND;  /* in case the number requested */
                           /* is more than the total          */
                           /* number of employees             */
    INSERT INTO temp VALUES (my_sal, my_empno, my_ename);
    COMMIT;
  END LOOP;
  CLOSE c1;
END;
```

Triggers

- ▶ Déclencheur (trigger)
 - ▶ Définir un ensemble d'actions qui sont déclenchées automatiquement par le SGBD lorsque des mises à jour sont effectuées
 - ▶ Ces actions sont enregistrées dans la base et non plus dans les programmes ou les applications

```
CREATE [OR REPLACE] TRIGGER <NOMTRIGGER>
{BEFORE | AFTER}
{DELETE | INSERT | UPDATE [OF COLUMN, [,
COLUMN] ...]}
[ OR {DELETE | INSERT | UPDATE [OF COLUMN,
[, COLUMN] ...]}] ...
ON <NOMTABLE> [FOR EACH ROW]
[WHEN <CONDITION>]
<BLOCPLSQL>
```

Triggers

```
CREATE OR REPLACE TRIGGER difference_salaire
BEFORE UPDATE ON Emp
FOR EACH ROW
WHEN (:new.Empno > 0)
DECLARE
sal_diff number;
BEGIN
sal_diff := :new.sal - :old.sal;
dbms_output.put(' Old : ' || :old.sal || ' New : ' || :new.sal || ' Difference : ' || sal_diff);
END ;
```

Prédicats

- ▶ Prédicat: condition associée à un schéma, à une table ou à une colonne
 - ▶ Prédicat de table ou de colonne
 - ▶ Définit une propriété sur les lignes de la table à respecter
 - ▶ Évalué lors de mise à jour d'une ligne de la table, et en cas de violation, la mise à jour est rejetée
 - ▶ Prédicats de schéma: nommé aussi assertion
 - ▶ Contraintes dont l'étendue dépasse les types de données, les colonnes et la table
 - ▶ Définir des règles de validation entre différentes colonnes de différentes tables
 - ▶ Considérés comme des objets de la base de données dans la norme SQL
 - ▶ Certains SGBDR proposent des mécanismes similaires, au lieu des assertions, en l'occurrence RULE (règles)

Prédicats : Clause Check

```
CREATE TABLE CLIENT  
( NUMCLIENT ..., ...,  
  CATEGORIE CHAR(2),  
  PRIMARY KEY (NUMCLIENT),  
  CHECK (CATEGORIE IS NULL OR CATEGORIE  
        IN ('C1','C2'))  
)
```

```
ALTER TABLE CLIENT  
ADD CONSTRAINT CHK_ CATEGORIE  
CHECK (CATEGORIE IS NULL OR CATEGORIE  
IN('C1','C2'))
```

```
CREATE ASSERTION NOM_ASSERTION  
CHECK (PREDICAT)  
[ATTRIBUT_ASSERTION]
```

```
DROP ASSERTION assertion_name;
```

```
CREATE ASSERTION salary_assertion CHECK  
(salary <= 100000)
```

Droits d'accès


- ▶ Langage SQL offre des fonctions de réglementation des droits d'accès aux utilisateurs pour les ressources de la base de données sous forme de privilège
- ▶ Privilège: autorisation qui est accordée à un utilisateur donné afin effectuer une opération sur une ressource de la base de données

Droits d'accès

GRANT <PRIVILRGES> ON <NOM DE TABLE> TO
<USER>* [WITH GRANT OPTION]

REVOKE < PRIVILRGES > ON < NOM DE TABLE >
FROM <USER>

<PRIVILEGES> ::= ALL PRIVILEGES | <ACTION>*
<ACTION> ::= SELECT | INSERT | UPDATE [(<NOM
DE COLONNE>+)]
| REFERENCE [(<NOM DE COLONNE>+)]
<USER> ::= PUBLIC | <IDENTIFIANT
D'AUTORISATION>

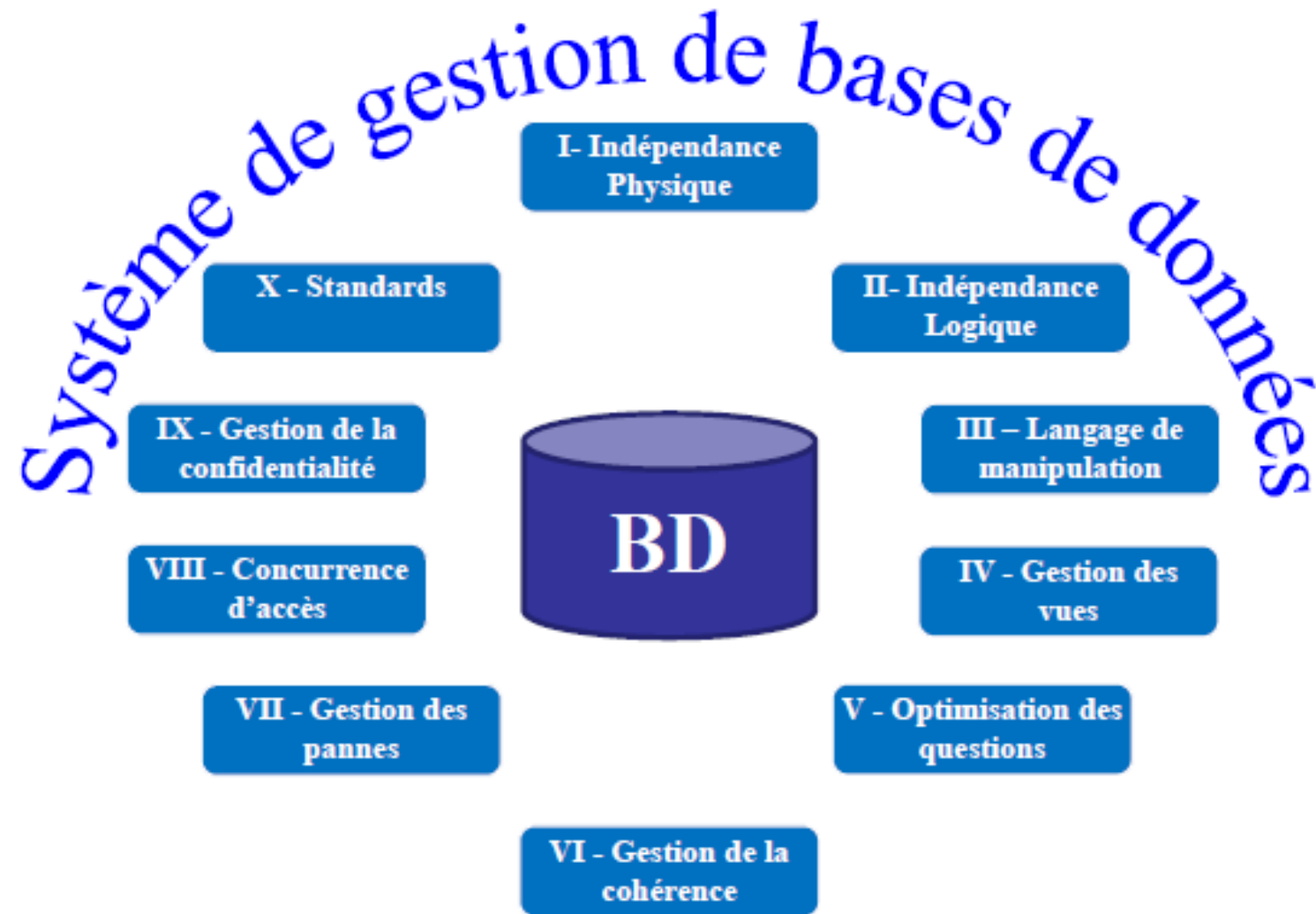


Partie V: Systèmes de Gestion des Bases de Données- Aspects système

Introduction

- ▶ Systèmes de Gestion de Bases de Données (SGBD)
 - ▶ Gestion de données par un langage de création, manipulation et interrogation
 - ▶ Gestionnaire de stockage sur disque
 - ▶ Gestionnaire de concurrence d'accès
 - ▶ Interfaces de programmation et d'administration
 - ▶ Etc...

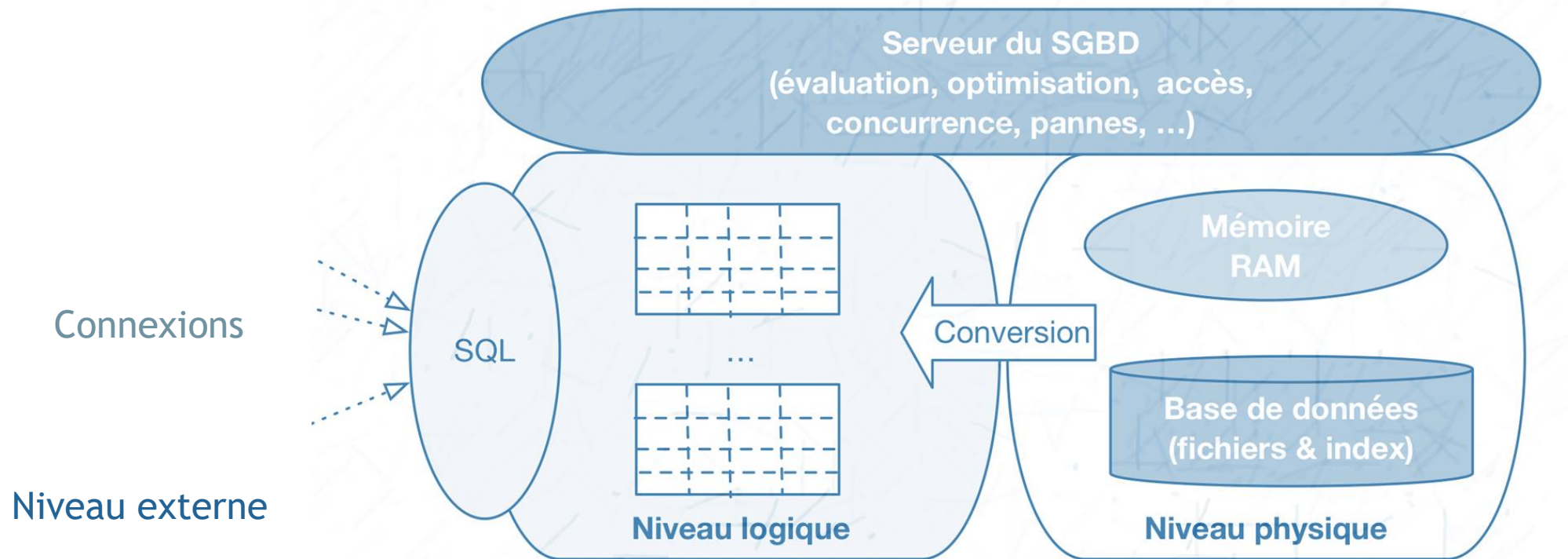
Introduction



Introduction

- ▶ Techniques des SGBD relationnels pour:
 - ▶ Contrôle de concurrence
 - ▶ Récupération sur erreur
 - ▶ Evaluation, optimisation et exécution des requêtes
 - ▶ Gestionnaire de données

Introduction



Introduction

▶ Niveau physique

- ▶ Gestion des fichiers, des données, du schéma, des index
- ▶ Partage de ressources de la base de données et gestion de la concurrence d'accès
- ▶ Reprise sur erreur
- ▶ Etc...

▶ Niveau logique

- ▶ Définition de la structure des données
- ▶ Consultation et Mise à Jour des données
- ▶ Gestion de la confidentialité (sécurité)
- ▶ Maintien de l'intégrité
- ▶ Etc...

▶ Niveau externe

- ▶ Interfaces conviviales
- ▶ Outils d'aide à la conception de schémas, chargement des données, etc...
- ▶ Environnement de programmation pour intégration avec les langages de programmation
- ▶ Etc...

Contrôle de concurrence

- ▶ Bases de données sont accessibles à plusieurs utilisateurs
- ▶ Accès simultanés à des informations partagées soulèvent plusieurs problèmes de cohérence
- ▶ SGBD doit garantir que l'exécution des programmes concurrents effectuant des mises à jour, se passe « correctement »
 - => transaction
 - => Contrôle d'accès concurrents

Contrôle de concurrence

- ▶ Transaction: séquence d'opérations de lecture ou de mise à jour sur une base de données, se terminant par:
 - ▶ Commit: indique la validation de toutes les opérations effectuées par la transaction
 - ▶ Rollback: indique l'annulation de toutes les opérations effectuées par la transaction

Contrôle de concurrence

► Approche Contrôle continu (approche pessimiste)

- Vérifier au fur et à mesure de l'exécution des opérations que le critère de sérialisabilité (exécutions concurrentes sont équivalentes à des exécutions en série) est bien respecté
- Repose sur l'idée que les conflits sont fréquents et qu'il faut les traiter le plus tôt possible

► Approche Contrôle par certification (approche optimiste)

- Vérifier la sérialisabilité quand la transaction s'achève
- Repose sur l'idée que les conflits sont rares et que l'on peut accepter de ré-exécuter les transactions qui posent problèmes

Récupération sur erreur

- ▶ Garantir l'intégrité physique des données en assurant une reprise correcte après des incidents système : panne/erreur
- ▶ Panne: tout événement logique ou physique qui provoque une fin anormale des transactions
 - ▶ Exemple: coupure électrique qui affecte le serveur, défaillance logicielle, défaillance matérielle, etc...
 - ▶ Panne d'une action (une commande au SGBD mal exécutée)
 - ▶ Panne de transaction (opération illégale sur les données, etc...)
 - ▶ Panne de système/mémoire principale
 - ▶ Panne de mémoire secondaire (disque dur)

Récupération sur erreur

- ▶ Journal des transactions : ensemble de fichiers complémentaires à ceux de la base de données, servant à stocker sur un support non volatile les informations nécessaires à la reprise sur erreur
 - ▶ Etat de la base de données= journaux de transactions + fichiers de la base de données
 - ▶ Journal contient les types d'enregistrements:
 - ▶ start(T)
 - ▶ write(T, x, old_val, new_val)
 - ▶ commit
 - ▶ rollback
 - ▶ checkpoint

Récupération sur erreur

HomeViewResources

New

Open

Session Settings

Save

Sessions

Print

Print Preview

Print

Export

Copy Rows to Clipboard

Actions

Find

Refresh

Results

Select All

Select Transaction

Check/Uncheck

Tools

Options

Old table ID mapping...

<input type="checkbox"/> Operation	Schema	Object	User	Begin Time	End Time
<input type="checkbox"/> INSERT	HumanResources	Shift	Fujitsu\Milena	2014-05-16 13:25:09	2014-05-16 13:25:09
<input type="checkbox"/> INSERT	HumanResources	Shift	Fujitsu\Milena	2014-05-16 13:25:28	2014-05-16 13:25:28
<input type="checkbox"/> UPDATE	Person	Address	Fujitsu\Milena	2014-05-16 13:26:07	2014-05-16 13:26:07
<input type="checkbox"/> UPDATE	Person	Address	Fujitsu\Milena	2014-05-16 13:26:19	2014-05-16 13:26:19
<input type="checkbox"/> INSERT	Person	ContactType	Fujitsu\Milena	2014-05-16 13:26:33	2014-05-16 13:26:33
<input type="checkbox"/> DELETE	Person	ContactType	Fujitsu\Milena	2014-05-16 13:26:44	2014-05-16 13:26:44

Row: 3 of 6

Operation Details

Row History

Undo Script

Redo Script

Transaction Information

Field	Type	Old Value	New Value
AddressID	int	1	1
AddressLine1	nvarchar(90)	1970 Napa Ct.	1970 Napa Street
AddressLine2	nvarchar(90)	NULL	NULL
City	nvarchar(30)	Bothell	Bothell
StateProvinceID	int	79	79
PostalCode	nvarchar(15)	98011	98011
SpatialLocation	geography	0xa6100000010ca8b6c28bce447406...	0xa6100000010ca8b6c28bce447406...
rowguid	uniqueidentifier	9a0dcb0d-36cf-483f-84d8-585c2d4ec...	9a0dcb0d-36cf-483f-84d8-585c2d4ec...
ModifiedDate	datetime	2002-01-04 00:00:00.000	2002-01-04 00:00:00.000

Fujitsu/SQL2012

AdventureWorks2012

Checked: 0 / 6

Excluded: 0 / 6

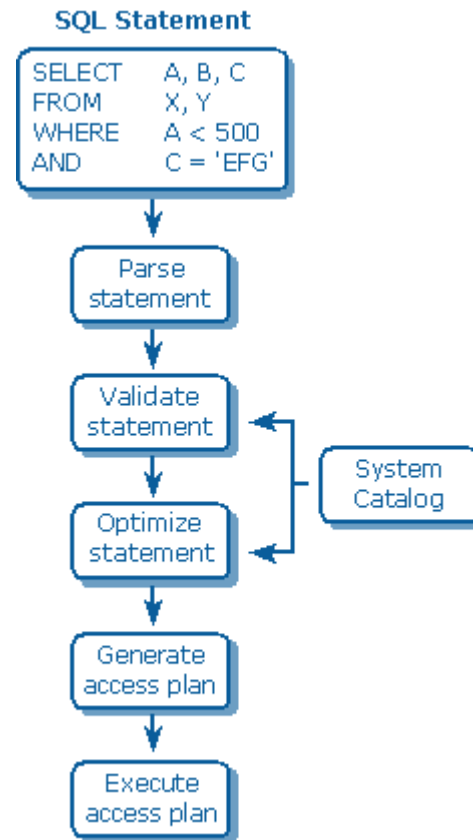
29/03/2023

Exemple de journal de transactions (Microsoft SQL Server)

Récupération sur erreur

- ▶ Mécanismes de récupération
 - ▶ Basés sur la base de données et les journaux des transactions
- ▶ Panne légère : algorithme avec deux types d'opérations
 - ▶ Redo: refaire les transactions validées avant la panne qui ne seraient pas correctement écrites dans les fichiers de la base et pour lesquelles « commit » figure dans le journal
 - ▶ Undo: Défaire les transactions actives au moment de la panne qui avaient effectué des mises à jour dans les fichiers de la base mais qui n'ont ni « commit », ni « rollback » dans le journal
- ▶ Panne de disque
 - ▶ Effectuer une reprise sur panne à partir des journaux, en appliquant les opérations Redo et Undo à la dernière sauvegarde disponible

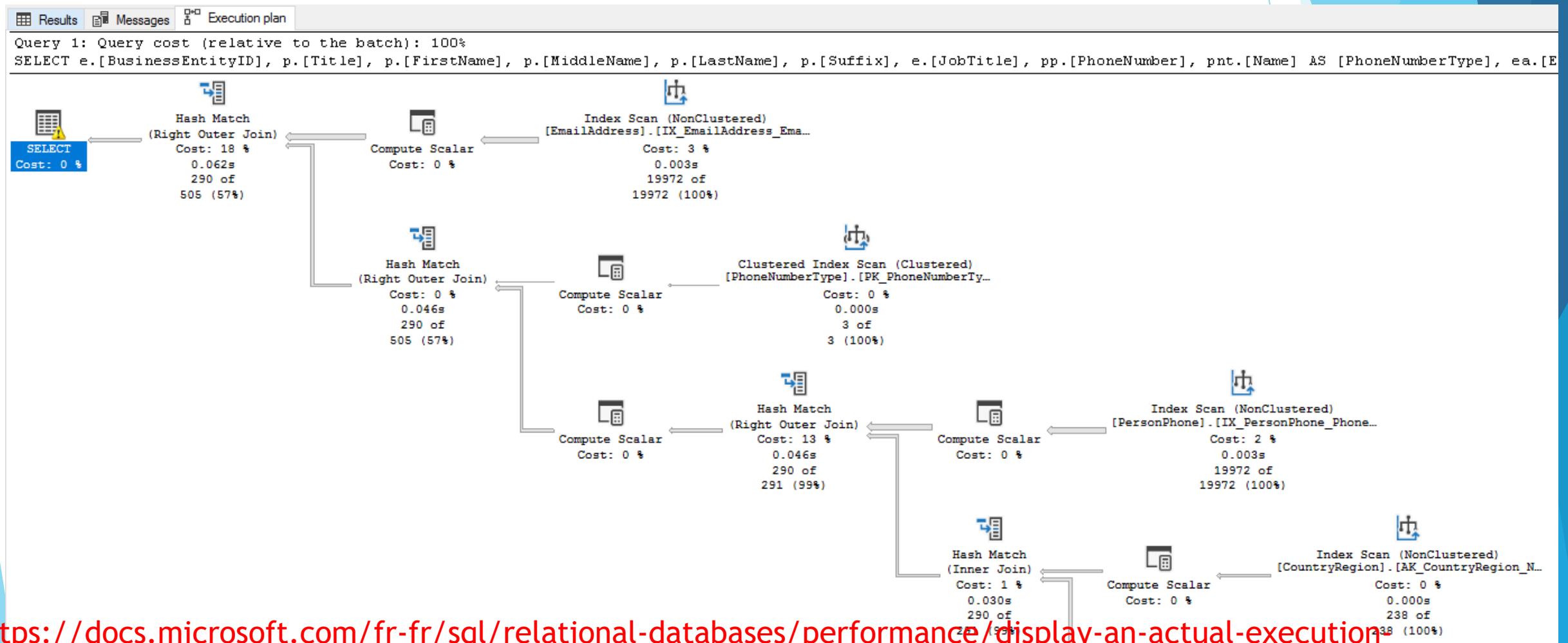
Evaluation, optimisation et exécution des requêtes



Evaluation, optimisation et exécution des requêtes

- ▶ Module d'évaluation de requêtes
 - ▶ Construire dynamiquement le plan d'exécution d'une requête
 - ▶ Plan d'exécution logique (PEL) : traduire la requête SQL en une expression de l'algèbre relationnelle, construite à partir de cinq opérateurs de base (sélection, projection, etc...)
 - ▶ Plan d'exécution physique (PEP): choisir les opérateurs adéquats en fonction de: présence ou non d'index, la taille des tables, la mémoire disponible, etc...

Evaluation, optimisation et exécution des requêtes



Evaluation, optimisation et exécution des requêtes

- ▶ Module d'optimisation de l'exécution des requêtes
 - ▶ Module important d'un SGBD
 - ▶ S'appuie sur la structure physique des données : fichiers séquentiels, index, etc...
 - ▶ Bénéficie de l'organisation physique des données et des caractéristiques de la requête afin de choisir le meilleur ordonnancement des opérations à effectuer sur la base de données

Gestionnaire de données

- ▶ Scénarios de gestion des données
 - ▶ Remplir initialement la base de données
 - ▶ Créer une nouvelle table en utilisant des données provenant d'autres sources/fichiers externes
 - ▶ Ajouter des données à des tables de la base de données
 - ▶ Créer des versions de développement et de test de la base de données
 - ▶ Créer un « snapshot » de la base de données pour la récupération sur erreurs

Gestionnaire de données

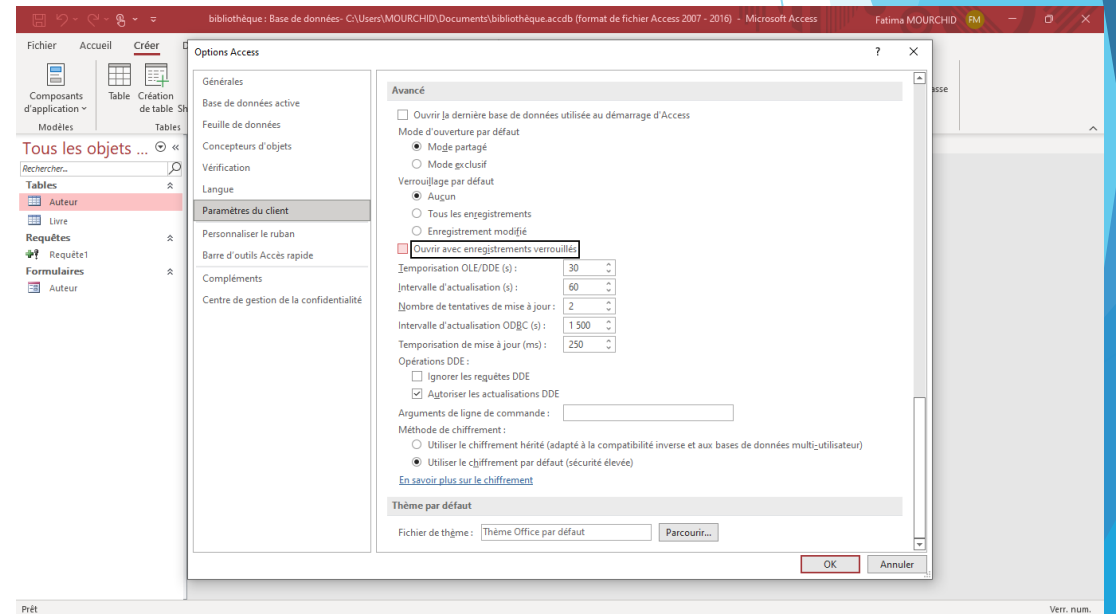
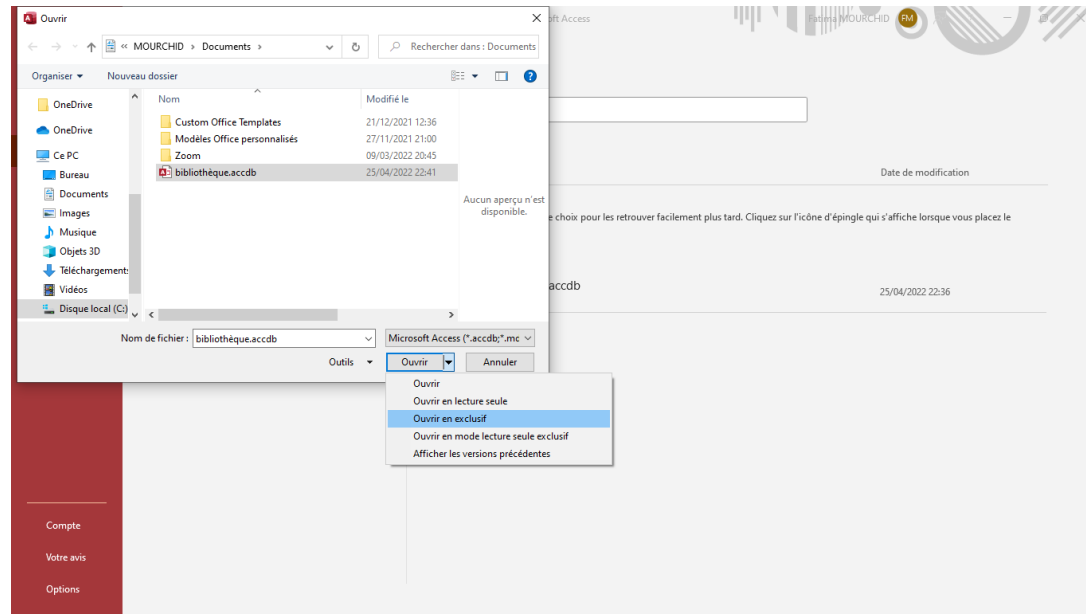
- ▶ Utilitaires de gestion de données
 - ▶ Utilitaires BACKUP et RESTORE: créer et restaurer des copies de bases de données y compris les objets: tables, vues, contraintes etc...
 - ▶ Utilitaire IMPORT: insérer des données dans une table spécifique à partir de différents formats, tels que DEL/CSV, ASC et IXF
 - ▶ Utilitaire EXPORT: enregistrer les données d'une table spécifique dans différents formats, tels que CSV
 - ▶ Utilitaires LOAD (au lieu de SQL INSERT): insérer rapidement de grandes quantités de données provenant de différentes sources de données

Pour conclure...

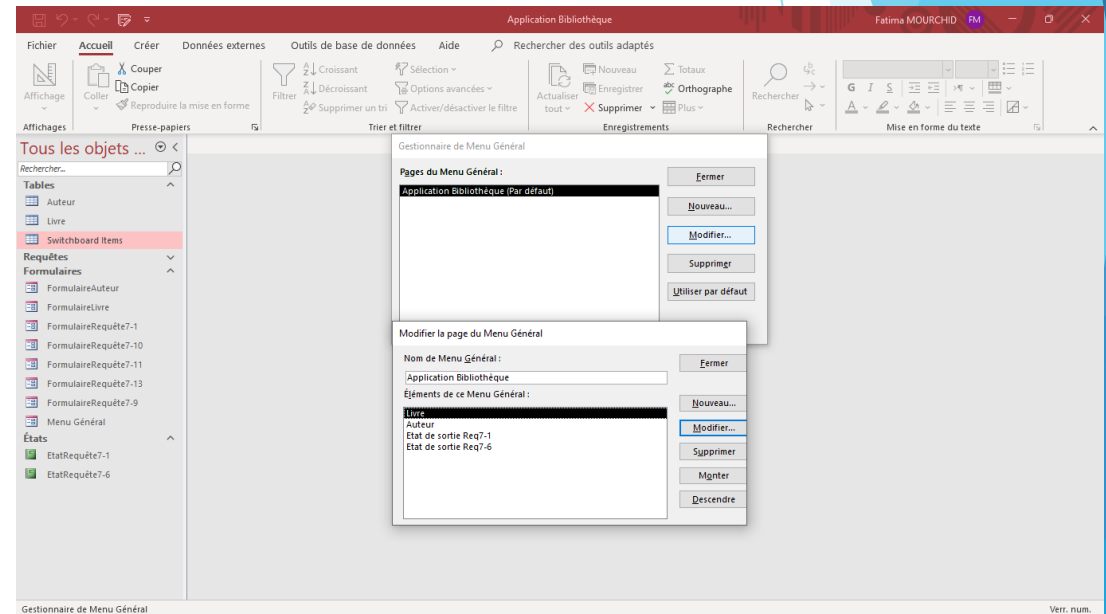
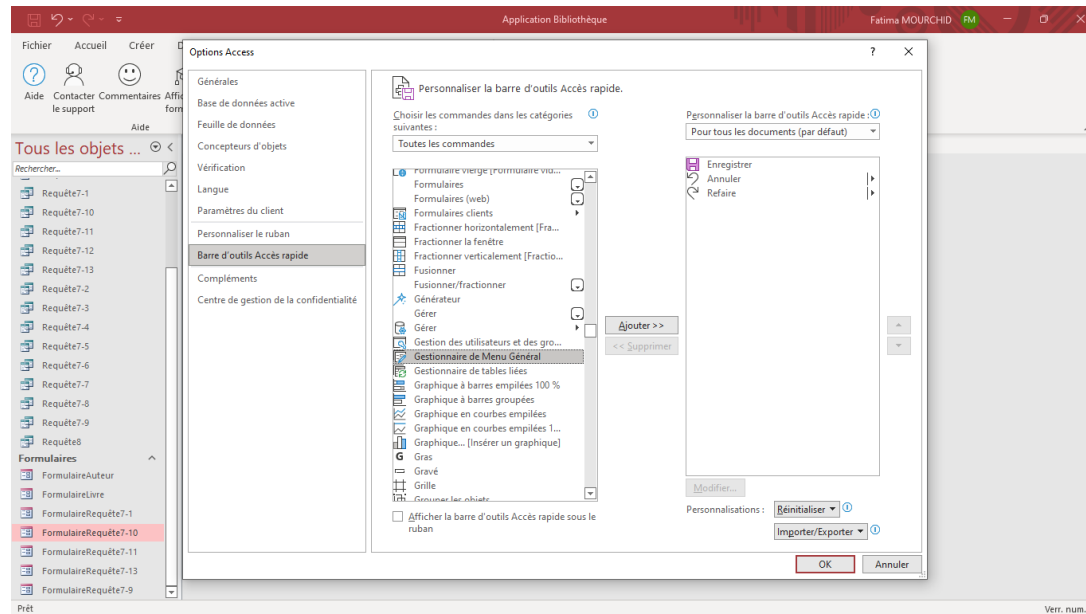
- ▶ Il n'existe pas de règle générale pour bien formuler une requête
 - ▶ Maîtriser les principes de logique relationnelle, d'équivalence de syntaxe et leurs interprétations
 - ▶ Aborder la formulation des requêtes de manière efficace et non pas produire une syntaxe simple
- ▶ La norme SQL a laissé la possibilité aux éditeurs de SGBD d'y ajouter des instructions spécifiques et non normalisées
 - => Provoque des différences dans la compréhension et l'interprétation d'un code source en SQL par les différents SGBD

► Correction des TPS

TP3- Bibliothèque

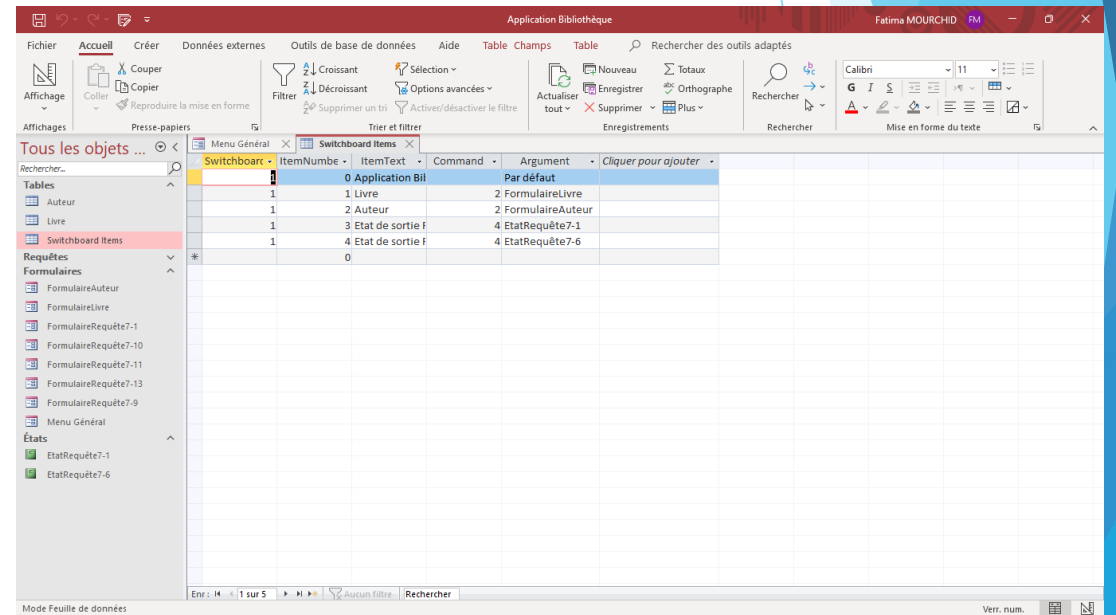
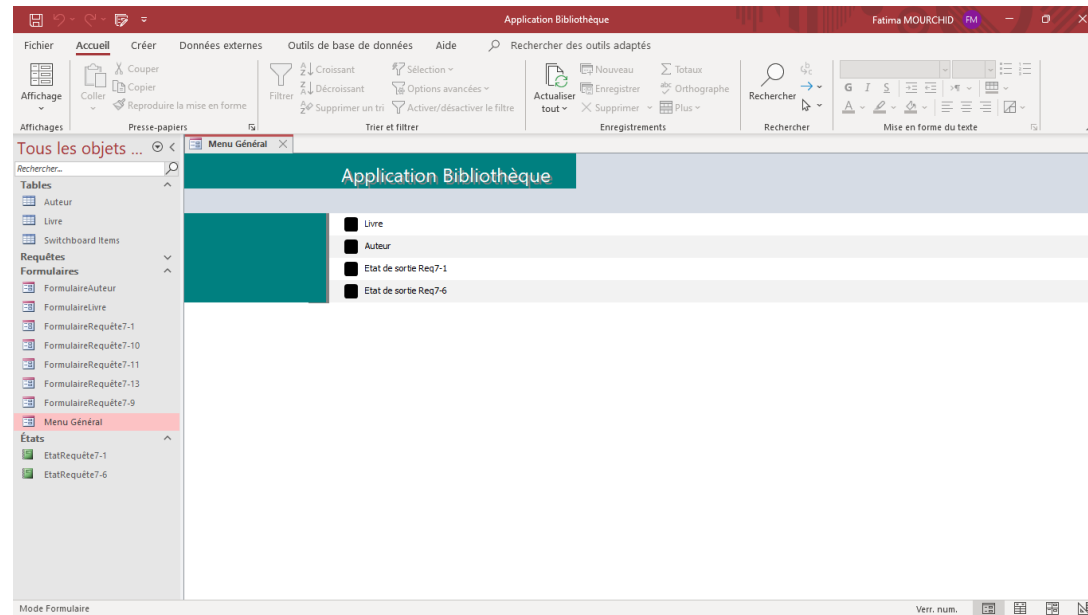


TP3- Bibliothèque



<https://support.microsoft.com/fr-fr/office/qu-est-devenu-le-menu-g%C3%A9n%C3%A9ral-f8b3d607-8f1f-4ecf-9979-79b1565f5471#:~:text=Cliquez%20sur%20Fichier%20%E2%80%92Options%20pour,Cliquez%20sur%20OK.>

TP3- Bibliothèque



TP5 et TP6

- ▶ PostgreSQL: <https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>
 - ▶ Système de gestion de base de données relationnelle et objet. C'est un outil libre disponible selon les termes d'une licence de type BSD
- ▶ DBeaver Community 23.0.0: <https://dbeaver.io/download/>
 - ▶ Permet l'administration et le requêtage de base de données. Pour les bases de données relationnelles, il utilise un driver JDBC. Pour les autres bases de données, il utilise des pilotes de base de données propriétaire
- ▶ pgAdmin: <https://www.pgadmin.org/download/>
 - ▶ Permet d'administrer une base de données PostgreSQL

TP5 et TP6

DBEaver 23.0.0 - <postgres> Console

Fichiers Edition Navigation Rechercher Editeur SQL Base de données Fenêtres Aide

SQL Appliquer (commit) Retour arrière (rollback) Auto postgres public@postgres

Navigateur de bases de données Projets

Enter a part of object name here

postgres - localhost:5432

- Databases
 - postgres 7,3M
 - Schemas
 - public
 - Tables
 - tvoiture 8K
 - Views
 - Materialized Views
 - Indexes
 - Functions
 - Sequences
 - Data types
 - Aggregate functions
 - Event Triggers
 - Extensions
 - Storage
 - System Info
 - Roles
 - Administer
 - System Info

Project - General

Name DataSource

- Bookmarks
- Diagrams
- Scripts

```
create Table Tvoiture (  
  Immatriculation int ,  
  Type varchar(10),  
  Couleur varchar(10),  
  Cylindre int ,  
  Nom varchar(10),  
  Marque varchar(10),  
  Modèle date,  
  CONSTRAINT PK_Tvoiture PRIMARY KEY (Immatriculation));  
insert into Tvoiture values (123456,'Essence','Verte', 7, 'Mégane', 'Renault', date('2004-12-02'));  
insert into Tvoiture values (234, 'Essence', 'Bleue', 8, 'Corrola', 'Toyota', date('1998-05-23'));  
insert into Tvoiture values (365,'Diesel', 'Rouge', 6, '106', 'Peugeot', date('2001-03-02'));  
  
select * from tvoiture t;
```

tvoiture 1 SQL Terminal

3 row(s) modified.

immatriculation	type	couleur	cylindre	nom	marque	modèle
123456	Essence	Verte	7	Mégane	Renault	2004-12-02
234	Essence	Bleue	8	Corrola	Toyota	1998-05-23
365	Diesel	Rouge	6	106	Peugeot	2001-03-02

3 row(s) fetched.

WET fr Inscriptible Insertion avancée 14: 1 [26] Sel: 26 | 1

TP5 et TP6

The screenshot shows the pgAdmin 4 web interface. On the left is a 'Browser' pane with a tree view of database objects. The 'public' schema is expanded, and the 'tvoiture' table is selected. The main pane shows a SQL query editor with the query: `select * from Tvoiture`. Below the editor is the 'Data Output' pane, which displays the results of the query in a table format. The table has 8 columns: `immatriculation` (integer), `type` (character varying), `couleur` (character varying), `cylindre` (integer), `nom` (character varying), `marque` (character varying), and `modèle` (date). The results show 3 rows of data. At the bottom, a status bar indicates 'Total rows: 3 of 3', 'Query complete 00:00:00.463', and a green message box stating 'Successfully run. Total query runtime: 463 msec. 3 rows affected.'

pgAdmin 4

File Object Tools Help

Browser

- > Casts
- > Catalogs (2)
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
 - public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > 1.3 Sequences
 - > Tables (1)
 - tvoiture
 - > Columns
 - > Constraints
 - > Indexes
 - > RLS Policies
 - > Rules

Dashboard Properties SQL Statistics Dependencies Dependents Processes postgres/postgres@PostgreSQL 15*

postgres/postgres@PostgreSQL 15

Query Query History

```
1 select * from Tvoiture
2
```

Scratch Pad

Data Output Messages Notifications

	immatriculation [PK] integer	type character varying (10)	couleur character varying (10)	cylindre integer	nom character varying (10)	marque character varying (10)	modèle date
1	123456	Essence	Verte	7	Mégane	Renault	2004-12-02
2	234	Essence	Bleue	8	Corrola	Toyota	1998-05-23
3	365	Diesel	Rouge	6	106	Peugeot	2001-03-02

Total rows: 3 of 3 Query complete 00:00:00.463

Successfully run. Total query runtime: 463 msec. 3 rows affected.

Correction des TPs

Les documents relatifs à la correction des TPs seront envoyés par Email

Bibliographie

- ▶ Alain Beaulieu, Learning SQL: Genrate, Manipulate and Retrieve Data
- ▶ Jean-Luc Hainaut, Bases de données Et modèles de calcul: Outils et méthodes Pour l'utilisateur
- ▶ Antoine Cornuéjols, Bases de données: concepts et programmation
- ▶ Laurent Audibert, Bases de données - de la modélisation au SQL
- ▶ Bases de données Access: <https://docs.microsoft.com/en-us/office/vba/api/overview/access>
- ▶ SQL Tutorial: <https://www.w3schools.com/sql/default.asp>