ESI
مدرسة علوم المعلومات
ⵜⵉⵎⵔⴰⵜ ⵏ ⵜⵓⵙⵙⵏⵉ ⵏ ⵜⵣⵔⴼⵜ
ECOLE DES SCIENCES DE L'INFORMATION

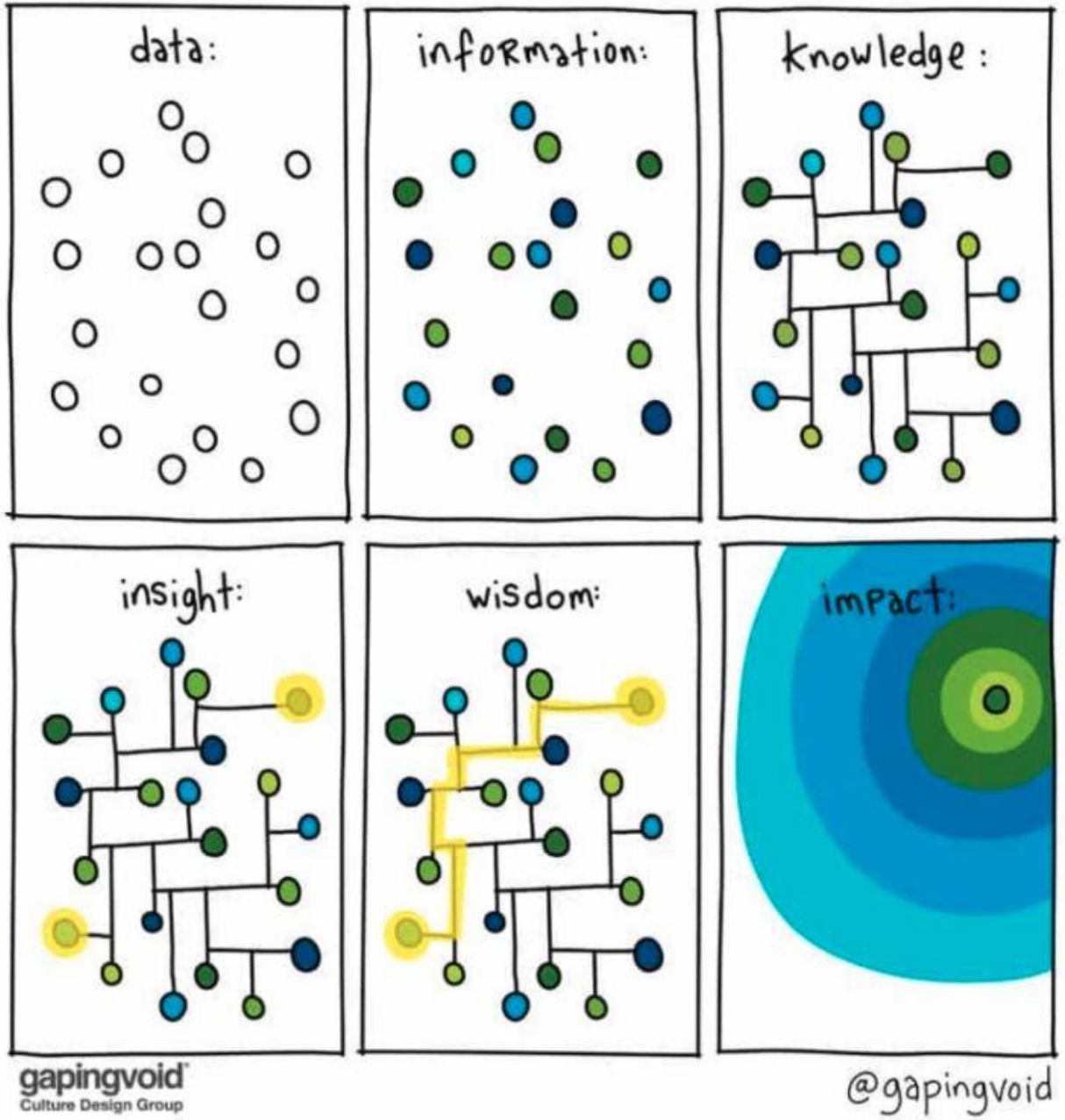
Bases de données

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


data:
information:
knowledge:
insight:
wisdom:
impact:
gapingvoid
Culture Design Group
@gapingvoid

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

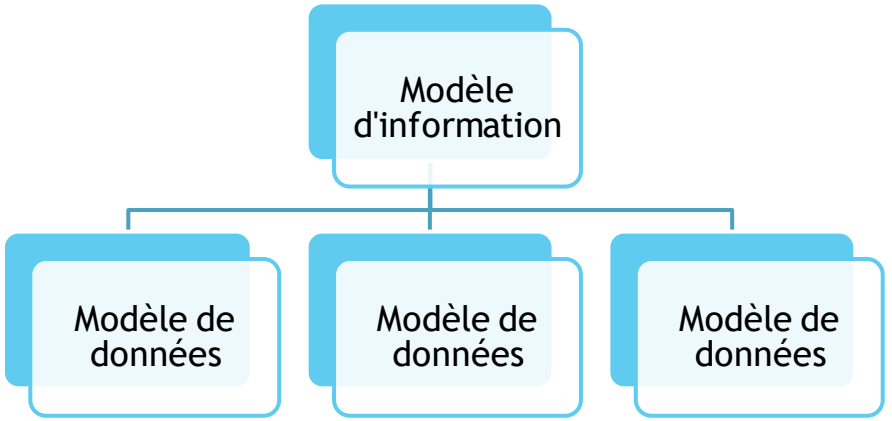
- Types de données
  - Données structurées
    - Schéma défini et une structure rigide (peuvent être représentées en lignes et en colonnes)
    - Caractéristiques des bases de données relationnelles
  - Données semi-structurées
    - Possèdent certaines propriétés organisationnelles, mais ne peuvent 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



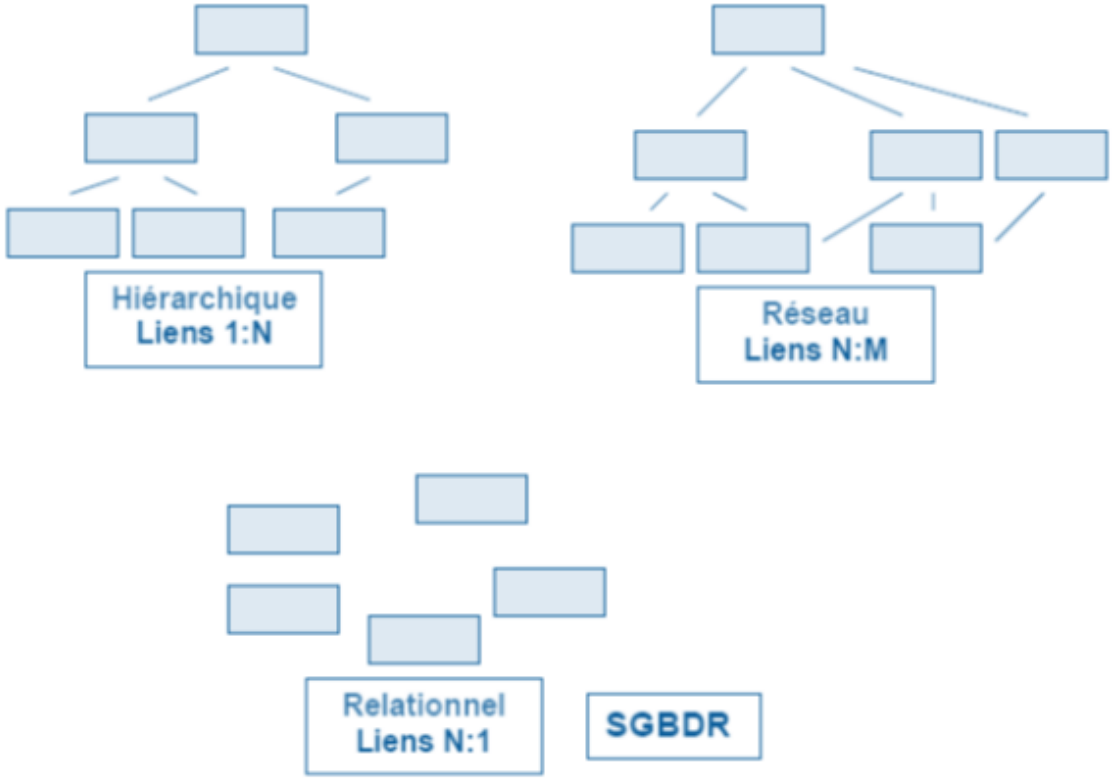
Modèles de bases de données

- 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

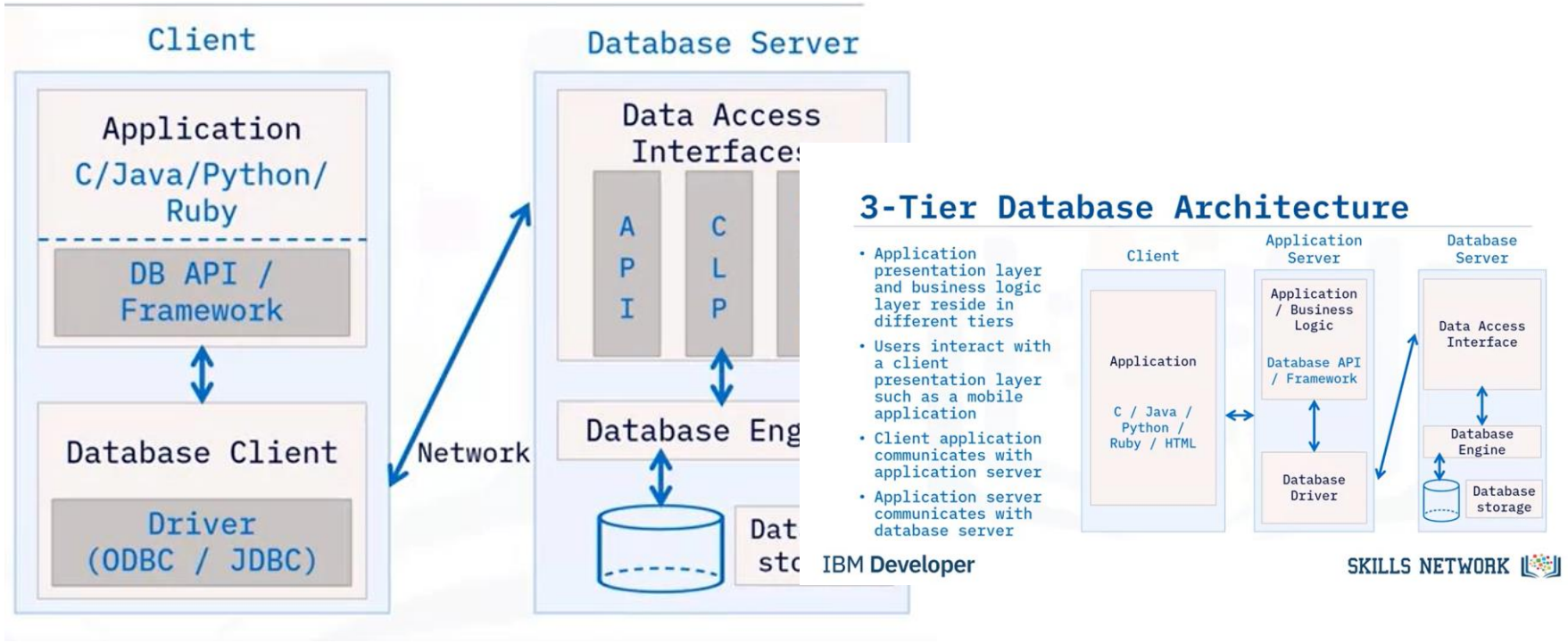
Typologie de bases de données

- 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 accèder à la base, masque la compléxité de la base de donneées 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 $A1, A2, \cdots, An$ est un sous-ensemble fini du produit cartésien $A1 \times A2 \times \cdots \times An$
  - Elément d'une relation de dimension n est un nuplet $(a1, a2, \cdots, an)$

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,⋯ ,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, <u>code</u>)

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 successives 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

C# Copier

```
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();
        }
    }
}
```

Exemple de Interactive SQL (https://docs.microsoft.com/fr-fr/dotnet/framework/data/adonet/ado-net-code-examples)

C++ Copy

```
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();
}
```


Exemple de Embedded SQL avec le langage C (https://learn.microsoft.com/en-us/sql/odbc/reference/embedded-sql-example?view=sql-server-ver16)

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 × 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 × I
  - Relation de degré n sur les domaines A1,A2,⋯ ,An est un sous-ensemble fini du produit cartésien A1 ×A2 ×⋯×An
  - Elément d'une relation de dimension n est un nuplet (a1,a2,⋯ ,an)

Modèle relationnel

- 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

- 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: ×
  - Union: ∪
  - 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 × expression logique ⟶ 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≥5

Algèbre relationnelle

- Projection
  - Supprimer les attributs autres que A1,… An d'une relation

    Π(A1… An)R

    relation × liste d'attributs ⟶ relation
  - Sélectionner des colonnes de la table
  - Si R est vide, la relation qui résulte de la projection est vide
  - Exemple: Π(NomDépartement) 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

    $R1 \cup R2$

    $relation \times relation \longrightarrow relation$
  - 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 U 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

    $R1 - R2$

    $relation \times relation \longrightarrow relation$
  - 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

    $R1 \times R2$

    $relation \times relation \longrightarrow relation$
  - 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 x 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

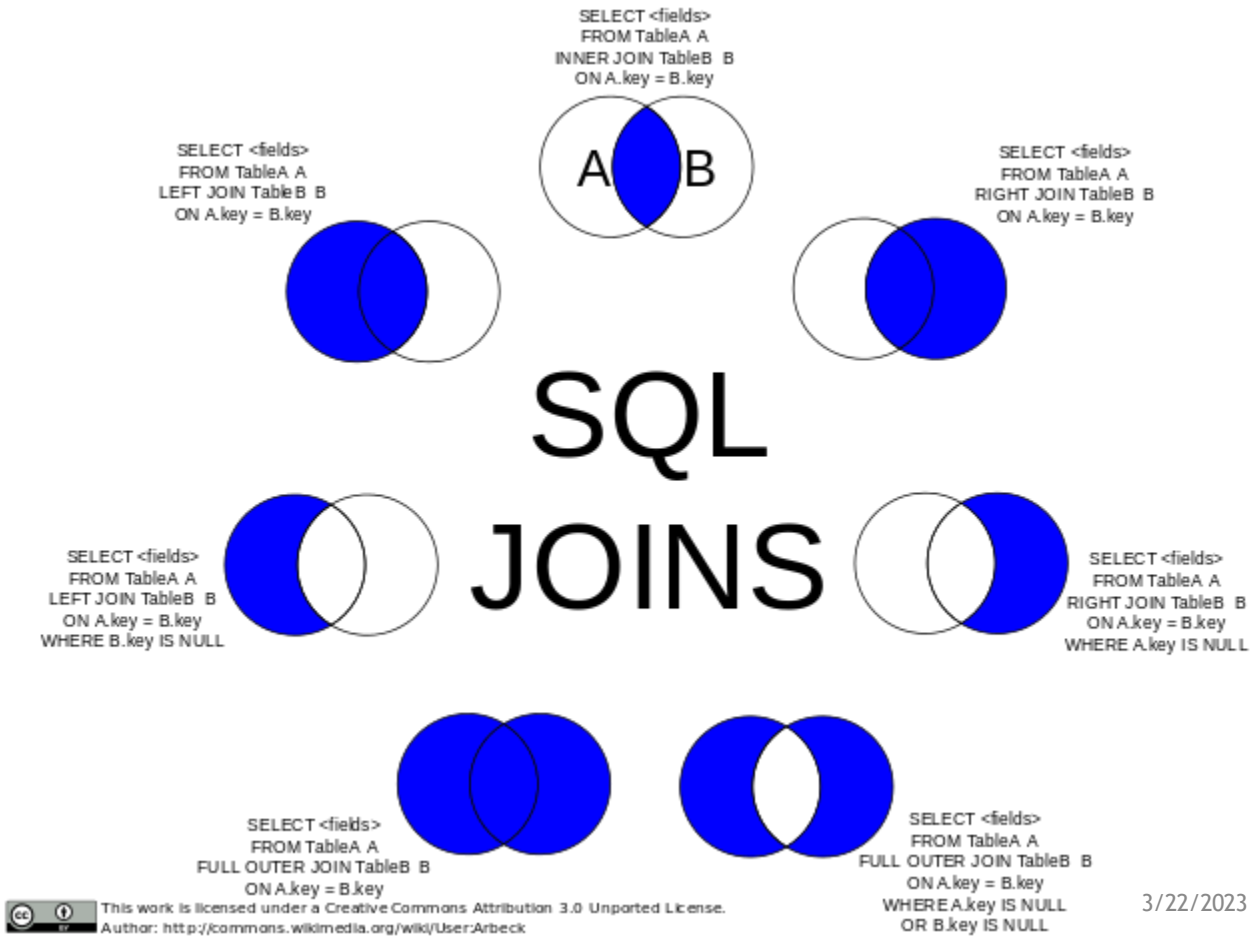
    $R1 \triangleright\!\triangleleft_E R2$

    $\text{relation} \times \text{relation} \times \text{expression logique} \longrightarrow \text{relation}$
  - Si R1 ou R2 ou les deux sont vides, la relation qui résulte de la jointure est vide
  - Jointure ≡ 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 client 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

- 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
```

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≤ML≤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 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

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

```
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)
```

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

- 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 )
```

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

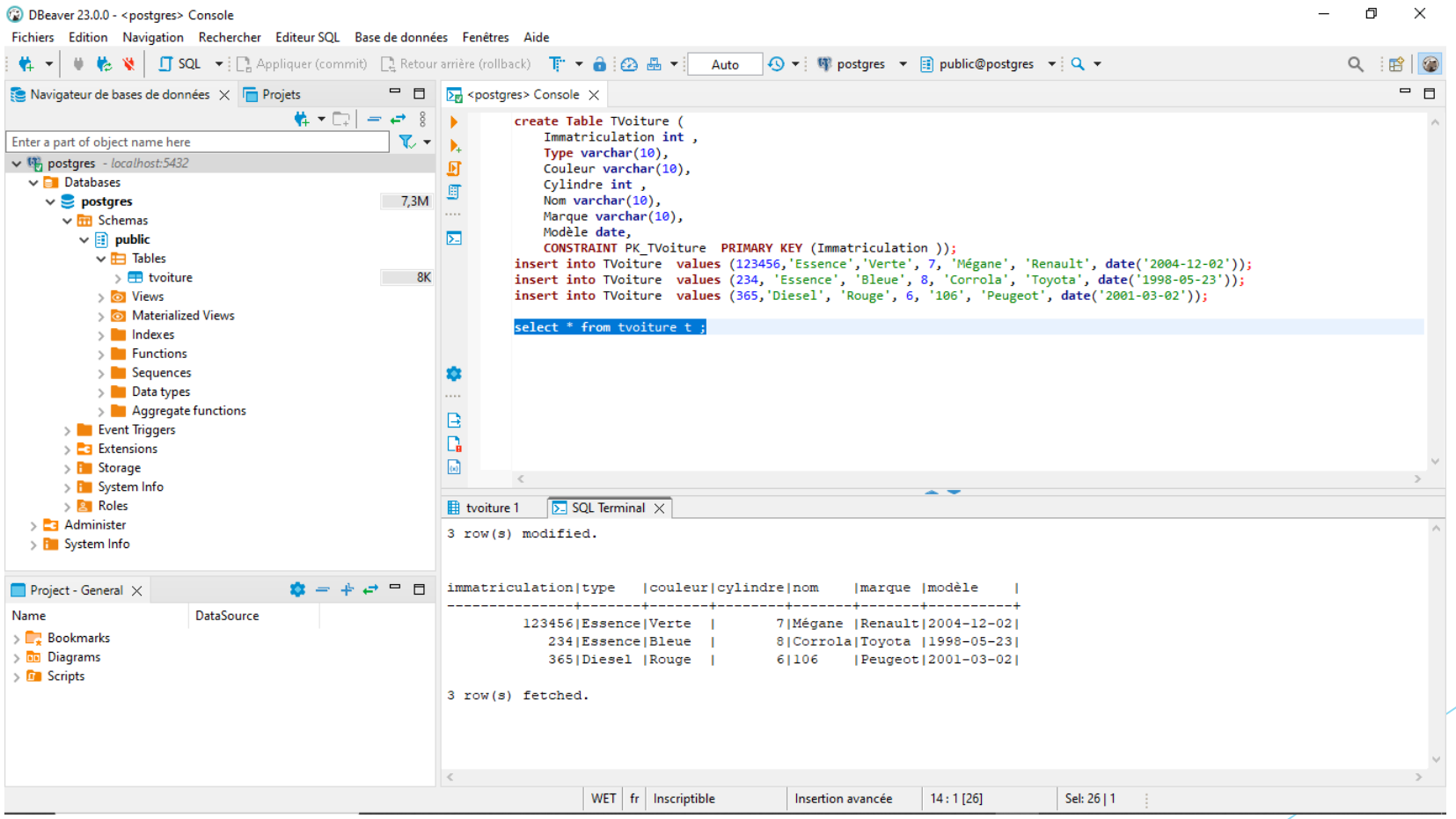
- Clause UNION ALL: similaire à la clause UNION, sauf qu'elle permet d'inclure tous les enregistrements, même les doublons

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

TPs

- 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/
  - DBeaver : 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/
  - pgAdmin permet d'administrer une base de données PostgreSQL

TPs


DBeaver 23.0.0 - <postgres> Console
Fichiers Edition Navigation Rechercher Editeur SQL Base de données Fenêtres Aide
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
<postgres> Console
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