I. Introduction & Concepts Fondamentaux

Comprendre le "Pourquoi" et le "Comment" global.

1. Qu'est-ce qu'une Base de Données (BD) ?

- **Définition :** Système permettant de mémoriser durablement des données structurées sur un support physique (disque)[1].
- **Objectifs clés :**
    - C.R.U.D. : Create (Ajouter), Read (Lire), Update (Modifier), Delete (Supprimer).
    - **Indépendance :** Séparer les données des programmes qui les utilisent.
    - **Cohérence & Non-redondance :** Éviter de stocker la même info plusieurs fois pour prévenir les erreurs.

2. Le SGBD (Système de Gestion de Bases de Données)

C'est le logiciel qui gère la BD (ex: Oracle, MySQL, PostgreSQL, SQL Server). Il assure :

- **L'accès aux données** (via requêtes).
- **L'intégrité et la cohérence** (respect des contraintes).
- **La sécurité** (gestion des accès) et la **reprise sur panne** (backup).

3. Architecture ANSI/SPARC (Les 3 niveaux)

Pour garantir l'indépendance des données, on distingue 3 niveaux :

1. **Niveau Externe (Vues) :** Ce que l'utilisateur voit (adapté à ses besoins spécifiques).
2. **Niveau Conceptuel (MCD) :** La structure globale des données (Entités, relations), indépendamment de l'informatique.
3. **Niveau Interne (Physique) :** Comment c'est stocké sur le disque (fichiers, index).

II. Le Modèle Relationnel (Théorie)

Le cœur de la matière. Il structure les données en "Tables".

1. Terminologie

- **Relation :** Une table (ex: Etudiant).

- **Attribut :** Une colonne (propriété, ex: Nom, Matricule).
- **Tuple :** Une ligne (une occurrence, ex: l'étudiant "Dupont").
- **Domaine :** L'ensemble des valeurs possibles pour un attribut (ex: Entiers, Chaîne de caractères).

2. Les Clés (Concepts vitaux)

- **Clé Candidate :** Un ensemble d'attributs qui permet d'identifier *de manière unique* un tuple. Elle doit être **minimale** (irréductible)[9].
- **Clé Primaire (PK - Primary Key) :** La clé candidate choisie pour identifier officiellement les tuples. Elle est **soulignée** et ne peut jamais être NULL (vide).
- **Clé Étrangère (FK - Foreign Key) :** Un attribut qui fait référence à la clé primaire d'une *autre* table. Elle permet de lier les tables entre elles (ex: ID_Cours dans la table Etudiant).

3. Les Contraintes d'Intégrité

- **Intégrité de domaine :** La valeur doit respecter le type (ex: pas de texte dans une colonne "Âge").
- **Intégrité d'entité :** La Clé Primaire est obligatoire (NOT NULL).
- **Intégrité référentielle :** Une clé étrangère ne peut contenir que des valeurs existant dans la table qu'elle référence (ou être NULL).
    - *Exemple :* On ne peut pas inscrire un étudiant à un cours qui n'existe pas dans la table Cours.

III. Normalisation (Assurer la qualité)

Question d'examen classique : "Normalisez cette table".

L'objectif est d'éviter la **redondance** et les **anomalies** (de mise à jour ou de suppression).

1. Dépendance Fonctionnelle (DF)

On dit que X -> Y (X détermine Y) si, connaissant la valeur de X, on connaît *forcément* une seule valeur de Y.

- *Exemple :* Matricule -> Nom (Un matricule correspond à un seul nom).

2. Les Formes Normales

Pour être propre, une table doit respecter ces règles hiérarchiques :

- **1ère Forme Normale (1FN) :**
    - Tous les attributs sont **atomiques** (non décomposables) et **monovalués** (une seule valeur par case).
    - *Exemple de violation :* Une colonne "Téléphones" contenant "0495123, 0478555". Il faut séparer.
- **2ème Forme Normale (2FN) :**
    - Être en 1FN.
    - **Aucune dépendance partielle** : Tout attribut non-clé doit dépendre de **toute** la clé primaire (et pas juste d'une partie).
    - *Astuce :* Si ta clé primaire n'a qu'un seul attribut, tu es automatiquement en 2FN.
    - *Correction :* Si un attribut dépend d'une partie de la clé, on le sort pour créer une nouvelle table.
- **3ème Forme Normale (3FN) :**
    - Être en 2FN.
    - **Aucune dépendance transitive** : Un attribut non-clé ne doit pas dépendre d'un autre attribut non-clé.
    - *Exemple :* Dans Voiture(id, modèle, marque), si modèle détermine marque, c'est une DF transitive. Il faut créer une table Modèle(modèle, marque).

IV. Algèbre Relationnelle (La logique derrière SQL)

Comprendre comment on manipule les ensembles.

1. **Sélection:** Choisir des **lignes** selon une condition (WHERE).
2. **Projection:** Choisir des **colonnes** (SELECT).
3. **Union:** Fusionner deux tables (supprime les doublons).
4. **Intersection:** Garder les éléments communs.
5. **Différence:** Éléments dans A mais pas dans B.
6. **Produit Cartésien:** Toutes les combinaisons possibles (souvent inutile sans filtre).
7. **Jointure:** Produit cartésien + Sélection (lier deux tables).

V. SQL - DDL (Définition des Données)

Créer et modifier la structure.

1. Création de table (CREATE TABLE)

SQL

```
CREATE TABLE Etudiant (
   Matricule INT PRIMARY KEY,          -- Clé primaire
   Nom VARCHAR(50) NOT NULL,           -- Obligatoire
   Prenom VARCHAR(50),
   Email VARCHAR(100) UNIQUE,          -- Doit être unique
   Age INT CHECK (Age >= 18),          -- Contrainte personnalisée
   ID_Cursus INT,
   -- Définition de la clé étrangère :
   FOREIGN KEY (ID_Cursus) REFERENCES Cursus(ID)
      ON DELETE CASCADE                -- Si on supprime le cursus, l'étudiant est supprimé
);
```

Note sur les clés étrangères : ON DELETE SET NULL mettrait ID_Cursus à vide au lieu de supprimer l'étudiant.

2. Modification (ALTER TABLE)

- Ajouter une colonne : ALTER TABLE Etudiant ADD Tel VARCHAR(15);
- Supprimer une table : DROP TABLE Etudiant;

VI. SQL - DML (Manipulation des Données)

Interroger et mettre à jour les données. C'est la partie pratique majeure.

1. Interrogation (SELECT)

Structure de base :

SQL

```
SELECT Colonne1, Colonne2    -- Projection
```

FROM Table -- Source

WHERE Condition -- Sélection

ORDER BY Colonne1 ASC; -- Tri (ASC=croissant, DESC=décroissant)

Opérateurs importants :

- **Comparaison :** =, <>, >, <, >=, <=.
- **Logique :** AND, OR (Attention aux parenthèses !).
- **Intervalles :** WHERE Age BETWEEN 18 AND 25.
- **Listes :** WHERE Couleur IN ('Rouge', 'Bleu').
- **Texte (Pattern) :** WHERE Nom LIKE 'D%' (Commence par D) ou '%a%' (Contient a).
- **Nullité :** WHERE Telephone IS NULL (Jamais = NULL).

2. Les Jointures (JOIN)

Indispensable pour récupérer des infos réparties sur plusieurs tables. **Évite les jointures naturelles (NATURAL JOIN)**, utilise des jointures explicites.

- **INNER JOIN (ou juste JOIN) :** Ne garde que les correspondances exactes.

SQL

SELECT E.Nom, C.NomCours

FROM Etudiant E

JOIN Cours C ON E.ID_Cours = C.ID;

- **LEFT OUTER JOIN :** Garde **tout** ce qu'il y a à gauche (Etudiant), même s'il n'y a pas de correspondance à droite (Cours). Les infos manquantes seront NULL.
    - *Utilité :* Trouver les étudiants inscrits à aucun cours (WHERE C.ID IS NULL).
- **RIGHT OUTER JOIN :** Même chose, mais garde tout ce qu'il y a à droite.
- **FULL OUTER JOIN :** Garde tout des deux côtés.

3. Mise à jour (UPDATE, INSERT, DELETE)

- **Insérer :**

SQL

INSERT INTO Etudiant (Matricule, Nom) VALUES (123, 'Dupont');

- **Modifier :**

SQL

UPDATE Etudiant

SET Age = 20

WHERE Matricule = 123; -- Sans WHERE, tous les âges sont modifiés !

- **Supprimer :**

SQL

DELETE FROM Etudiant

WHERE Matricule = 123; -- Sans WHERE, la table est vidée !