FICHE DE RÉVISION : BASES DE DONNÉES RELATIONNELLES

1. Commandes de création et mise à jour de tables en SQL

Création de tables (CREATE TABLE)

Syntaxe de base :
sql

```sql
CREATE TABLE nom_table (
    colonne1 type_donnees [contraintes],
    colonne2 type_donnees [contraintes],
    ...

      );
```

- Types de données courants : INT, VARCHAR(n), DATE, DECIMAL(p,s), BOOLEAN.
- Contraintes :
    - PRIMARY KEY : Clé primaire (unicité + non null).
    - FOREIGN KEY : Clé étrangère (référence une autre table).
    - NOT NULL : Valeur obligatoire.
    - UNIQUE : Unicité des valeurs.
    - DEFAULT valeur : Valeur par défaut.

Exemple :
sql

```sql
CREATE TABLE Etudiants (
    id_etudiant INT PRIMARY KEY,
    nom VARCHAR(50) NOT NULL,
    age INT DEFAULT 18,
    id_classe INT,
    FOREIGN KEY (id_classe) REFERENCES Classes(id_classe)
```

-);

Modification de tables (ALTER TABLE)

- Ajouter une colonne :
  sql

  ```sql
  ALTER TABLE nom_table ADD colonne type_donnees [contraintes];
  ```

- Modifier une colonne :
  sql

  ```sql
  ALTER TABLE nom_table MODIFY colonne type_donnees [contraintes];
  ```

- Supprimer une colonne :
  sql

  ```
  ALTER TABLE nom_table DROP COLUMN colonne;
  ```

- Exemple :
  sql

  ```
  ALTER TABLE Etudiants ADD email VARCHAR(100);
  ```

Suppression de tables (DROP TABLE)

- Syntaxe :
  sql

  ```
  DROP TABLE nom_table [CASCADE];
  ```

  - CASCADE : Supprime aussi les dépendances (ex. clés étrangères).

Mise à jour des données

- Insertion (INSERT INTO) :
  sql

  ```
  INSERT INTO nom_table (colonne1, colonne2) VALUES (valeur1, valeur2);
  ```

- Mise à jour (UPDATE) :
  sql

  ```
  UPDATE nom_table SET colonne1 = valeur WHERE condition;
  ```

- Suppression (DELETE) :
  sql

  ```
  DELETE FROM nom_table WHERE condition;
  ```

Exemple :
sql

```
INSERT INTO Etudiants (id_etudiant, nom) VALUES (1, 'Dupont');
UPDATE Etudiants SET age = 20 WHERE id_etudiant = 1;
```

- `DELETE FROM Etudiants WHERE age < 18;`

---

2. Gestion des droits d'utilisateurs

Création d'utilisateurs

- Syntaxe :
  sql

  ```sql
  CREATE USER 'nom_utilisateur'@'hote' IDENTIFIED BY 'mot_de_passe';
  ```

- Exemple :
  sql

  ```sql
  CREATE USER 'etudiant'@'localhost' IDENTIFIED BY 'pass123';
  ```

Attribution de privilèges (GRANT)

- Syntaxe :
  sql

  ```sql
  GRANT privilege [, privilege] ON nom_base.nom_table TO
  'utilisateur'@'hote';
  ```

- Privilèges courants : SELECT, INSERT, UPDATE, DELETE, ALL PRIVILEGES.

  Exemple :
  sql

  ```sql
  GRANT SELECT, INSERT ON universite.Etudiants TO
  'etudiant'@'localhost';
  ```

Retrait de privilèges (REVOKE)

- Syntaxe :
  sql

  ```sql
  REVOKE privilege [, privilege] ON nom_base.nom_table FROM
  'utilisateur'@'hote';
  ```

- Exemple :
  sql

  ```sql
  REVOKE INSERT ON universite.Etudiants FROM 'etudiant'@'localhost';
  ```

Suppression d'utilisateurs

- Syntaxe :
  sql

```sql
DROP USER 'nom_utilisateur'@'hote';
```

---

3. Transactions et concurrence d'accès

Gestion des transactions

- Une transaction est un ensemble d'opérations qui doivent être exécutées comme une unité (propriétés ACID : Atomicité, Cohérence, Isolation, Durabilité).
- Commandes :
  - START TRANSACTION; : Débute une transaction.
  - COMMIT; : Valide la transaction (sauvegarde permanente).
  - ROLLBACK; : Annule la transaction.

Exemple :
sql

```sql
START TRANSACTION;
UPDATE Etudiants SET age = 21 WHERE id_etudiant = 1;
INSERT INTO Etudiants (id_etudiant, nom) VALUES (2, 'Martin');
```

- `COMMIT; -- ou ROLLBACK si erreur`

Concurrence d'accès

- Problèmes courants :
  - **Lecture sale (Dirty Read)** : Lire des données non validées.
  - **Lecture non répétable (Non-Repeatable Read)** : Données modifiées entre deux lectures.
  - **Lecture fantôme (Phantom Read)** : Nouvelles lignes apparaissent entre deux lectures.
- Niveaux d'isolation (SET TRANSACTION) :
  - READ UNCOMMITTED : Permet les lectures sales.
  - READ COMMITTED : Évite les lectures sales.
  - REPEATABLE READ : Évite les lectures non répétables.
  - SERIALIZABLE : Isolation maximale, évite tous les problèmes.
- Syntaxe :
  sql

```sql
SET TRANSACTION ISOLATION LEVEL niveau;
```

Verrouillage (LOCK)

- LOCK TABLES nom_table READ|WRITE; : Verrouille une table pour lecture ou écriture.
- UNLOCK TABLES; : Libère les verrous.

Exemple :
sql

```sql
LOCK TABLES Etudiants WRITE;
UPDATE Etudiants SET age = 22 WHERE id_etudiant = 1;
```

- UNLOCK TABLES;