BD : Langage SQL

I. Contrainte d'intégrité :

Une contrainte d'intégrité est une règle appliquée à une colonne ou à une table et qui doit être toujours vérifiée.

Il existe 3 types de contraintes d'intégrité :

- **Les contraintes de domaines :**
  Ce sont des contraintes appliquées à des colonnes. Elles fixent l'obligation ou non d'une colonne et les régles de validité des valeurs de cette colonne.
- **Les contraintes d'intégrité de tables :**
  Elles permettent d'assurer que chaque table a une clé primaire.
- **Les contraintes d'intégrité référentielles :**
  Représentées sous forme de lien entre tables et permettent de s'assurer que les valeurs introduites dans une colonne (clé étrangère) et doivent figurer dans une autre colonne en tant que clé primaire.

Exemples :

Code_Produit INT (5) **PRIMARY KEY**,

Nom VARCHAR (20) **NOT NULL**,

Date_vente DATE,

Prix DECIMAL (5,3) **CHECK (Prix > 0)**

Exercice :

On présente ci-dessous le contenu des trois tables **Client**, **Article** et **Facture** d'une base de données. Cette base a été conçue par un débutant et présente certaines anomalies.

Table Client

CodCl	NomCl	PrenCl
123	ELBAHI	Anas
426	JELLITI	Marouen
456	CHEMLI	Anouar
789	ELBAHI	Cyrine
789	GRISSA	Aicha

Table Article

CodArt	LibArtl	PrixArt
003445	PC HP	1380
004516	PC IBM	-1490
012365	PC SIEM.	1320
023146	PC DELL	1200
045696	PC SIEM.	1325
098745	IMP. HP	420

Table Facture

NumFact	DatFact	CodCl	CodArt
125/09	25/01/09	123	012365
126/09	26/01/09	426	045696
127/09	18/02/09	456	004516
128/09	22/02/09	456	023146
129/09	03/03/09	789	111111
130/09	03/03/09	123	003445

NB : On suppose qu'une facture ne concerne qu'un seul article.

En se basant sur les contenus de ces tables, il apparait que trois contraintes d'intégrité n'ont pas été respectées.

Encercler l'anomalie et remplir le tableau suivant en expliquant à partir d'un exemple significatif l'anomalie (l'erreur) rencontrée et nommer la contrainte d'intégrité correspondante qui n'a pas été respectée.

Anomalie rencontrée (exemple et explication)	Contrainte d'intégrité non respectée

II. Commandes du langage SQL :

- Application 1 :

Donner les commandes SQL permettant de créer les 3 tables de la base de données ‘Bibliothèque’ ayant la représentation textuelle suivante et en se basant sur la liste de colonnes.

- Livre (Code_livre, Titre, Auteur, Editeur, Prix, Date_parution)
- Abonne (Num_abonne, Nom_abonne, Prenom_abonne, Adresse, Tel)
- Emprunt (Code_liv_emp#, Num_ab_emp#, Date_emprunt, Date_retour)

Liste des colonnes							
Nom colonne	Description	Type de données	Taille	Obligatoire	Valeur par défaut	Valeurs autorisées	Sujet
Code_livre	Code unique du livre	Numerique	4	O			Livres
Titre	Titre du livre	Caractère	50	O			Livres
Auteur	Noms des auteurs	Caractère	50	N			Livres
Editeur	Nom de la maison d’édition	Caractère	30	N			Livres
Prix	Prix d’un livre	Numérique	8,3	N		>0	Livres
Date_parution	Date de sortie du livre	Date		N			Livres
Num_abonne	Numéro de l’abonné	Numérique	5	O			Abonnés
Nom_abonne	Nom de famille de l’abonné	Caractère	20	O			Abonnés
Prenom_abonne	Prénom de l’abonné	Caractère	20	O			Abonnés
Adresse	Adresse de l’abonné	Caractère	50		TUNIS		Abonnés
Tel	N° de téléphone de l’abonné	Caractère	20				Abonnés
Code_liv_emp	Code du livre emprunté	Numérique	4	O			Emprunts
Num_ab_emp	Numéro de l’abonné empruntant un livre	Numérique	5	O			Emprunts
Date_emprunt	Date de l’emprunt du livre	Date		O			Emprunts
Date_retour	Date de retour du livre	Date					Emprunts

```
CREATE TABLE livre (
Code_livre INT (4) PRIMARY KEY,
Titre VARCHAR (50) NOT NULL,
Auteur VARCHAR (50),
Editeur VARCHAR (30),
Prix DECIMAL (8,3) CHECK (Prix >0),
Date_parution Date ) ;

CREATE TABLE Abonne (
Num_abonne INT (5) PRIMARY KEY,
Nom_abonne VARCHAR (20) NOT NULL,
Prenom_abonne VARCHAR (20) NOT NUL,
Adresse VARCHAR (50) DEFAULT "TUNIS",
Tel VARCHAR (20) ) ;

CREATE TABLE Emprunt (
Code_liv_emp INT (4) REFERENCES Livre (Code_livre),
Num_ab_emp INT (5) REFERENCES Abonne (Num_abonne),
Date_emprunt DATE NOT NULL,
Date_retour DATE,
PRIMARY KEY (Code_liv_emp, Num_ab_emp, Date_emprunt) ) ;
```

Remarques :

- L'option **NULL** veut dire que la colonne n'est pas obligatoire. On peut lors de la saisie d'une ligne de la table, laisser la valeur de cette colonne à vide. A l'inverse, l'option **NOT NULL** veut dire que la colonne est obligatoire.

→Lorsqu'une colonne est une clé primaire, on n'a pas besoin de préciser l'option NOT NULL. Elle est implicite.

- L'option **DEFAULT** permet d'attribuer une valeur par défaut à cette colonne lorsqu'aucune valeur ne lui a été affectée. Cette option ne peut pas être indiquée lorsque la colonne est obligatoire (NOT NULL).
- **PRIMARY KEY** spécifie que la colonne est utilisée comme clé primaire.
- **REFERENCES** définit une contrainte d'intégrité référentielle. Le nom de la table précisé après le mot-clé REFERENCES est celui de la table mère. Le nom de la colonne est celui de la colonne vers laquelle on se réfère et il ne doit être précisé que lorsqu'il est différent du nom de la colonne courante.
- **CHEK** est utilisé lorsqu'on veut qu'une condition soit vérifiée pour chaque valeur insérée.

Recherche de données : Requêtes

Une requête est une opération de recherche de données à partir d'une ou plusieurs tables.

Cette recherche peut concerner :

- Certaines colonnes d'une table : **Projection**
- Certaines lignes d'une table : **Sélection**
- Deux tables en relation : **jointure**

→Une requête peut être réalisée en combinant ces trois actions.

A. Requêtes de projection :

Cette requête ne concerne qu'une seule table de la BD, elle doit comporter au moins une colonne de la table, et toutes les lignes de cette colonne seront affichées au résultat.

SELECT **[DISTINCT]** * / liste_nom_colonnes FROM **nom_table** ;

Remarques :

• La liste_nom_colonnes précise les colonnes à afficher au résultat, elle peut être remplacée par ***** pour tout afficher.

• **DISTINCT** permet d'éliminer les lignes en double dans le résultat.

• Le résultat de la commande **SELECT** est une nouvelle table résultat.

• On peut utiliser des **ALIAS**, pour modifier les noms des colonnes de la table résultat.

Exemple :

Personnes

nom	*prénom*	*adresse*	*téléphone*
Martin	Pierre	7 allée des vers	258941236
Dupond	Jean	32 allé Poivrot	526389152
Dupond	Marc	8 rue de l'octet	123456789

```
SELECT nom, prénom
FROM   Personnes
```

On projette la table ***Personnes*** sur les colonnes *nom* et *prénom*.

nom	*prénom*
Martin	Pierre
Dupond	Jean
Dupond	Marc

B. Requêtes de Sélection :

Cette requête permet d'afficher les lignes qui vérifient une condition.

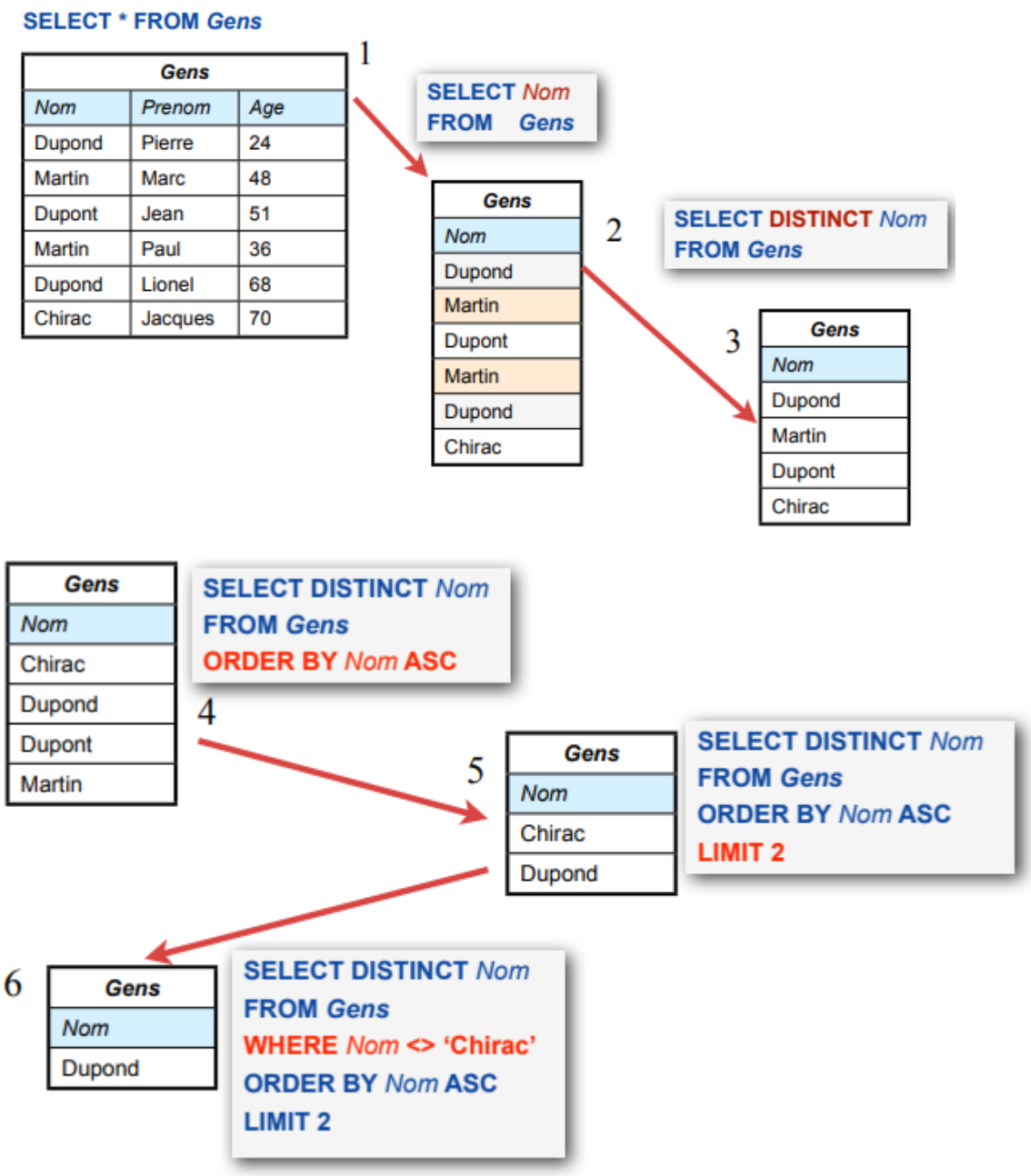
SELECT [DISTINCT] * / liste_nom_colonnes

FROM nom_table WHERE condition ;

Remarques :

• Le paramètre condition précise le critère qui doit être vérifié par les lignes à afficher.

Exemple :



C. Requêtes Jointure :

C'est une recherche à partir de plusieurs tables.

SELECT [DISTINCT] * / liste_nom_colonnes

FROM nom_table1 [Alias1], nom_table2 [Alias2]...

WHERE condition ;

Remarques :

• La condition de jointure doit porter sur les colonnes en communs aux tables (clé primaire-clé étrangère) qui ne doit pas être confondue avec la condition critère de sélection.

• Pour alléger l'écriture de la commande SELECT, un Alias peut être utilisé. Il permet de donner un nom abrégé à une table.

Personnes

nom	*prénom*	*adresse*	*téléphone*
Martin	Pierre	7 allée des vers	258941236
Dupond	Jean	32 allé Poivrot	526389152

Bibliothèque

nom	*Dernierlivre*
Dupond	Robinson
Jospin	Faust
Martin	Misère

```
SELECT Personnes.prénom, dernierlivre
FROM   Personnes, Bibliothèque
WHERE Personnes.nom = Bibliothèque.nom
```

prénom	*Dernierlivre*
Jean	Robinson
Pierre	Misère

On joint les deux tables, grâce à la colonne *nom*.

Et on combine cette jointure à une projection sur les attributs *nom* et *dernierlivre*.

Attention à lever toute ambiguïté sur les noms d'attribut dans le cas où deux tables possèdent des colonnes de même nom.

D. Recherche de données avec Tri :

SELECT [DISTINCT] * / liste_nom_colonnes

FROM nom_table1 [Alias1], nom_table2 [Alias2]...

[WHERE condition]

ORDER BY nom_colonne1 [**ASC / DESC**] [, nom_colonne2 [ASC/ DESC]....]

Remarques :

• ORDER BY réalise le tri.

• ASC /DESC ordre croissant / décroissant.

• Le tri peut être associé à n'importe quelle opération de recherche (projection, sélection et jointure).

E. Requête de calcul : Groupement

- Application :

Il est possible de grouper des lignes de données ayant une valeur commune à l'aide de la clause GROUP BY et des fonctions de groupe.

Syntaxe :

SELECT Attr1, Attr2 … Fonction_Groupe
FROM Nom_Table1.NomTable2…
WHERE Liste_Condition
GROUP BY Liste_Groupe
HAVING Condition ;

✓ Donner le nombre de commandes par client.

```
SELECT NCl, COUNT((NCmd) NbCmd
FROM Commande
GROUP BY NCl ;
```

Commande

NCmd	DateCmd	NCl
C001	10/12/2003	CL02
C002	13/02/2004	CL05
C003	15/01/2004	CL03
C004	03/09/2003	CL10
C005	11/03/2004	CL03

NCl	NbCmd
CL02	1
CL05	1
CL03	2
CL10	1

✓ Donner la quantité totale commandée par produit.

```
SELECT NP, SUM(Qte) Som
FROM Ligne_Cmd
GROUP BY NP;
```

Ligne_Cmd

NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

NP	Som
P001	950
P002	400
P004	450
P005	70
P006	100
P007	550
P008	90

✓ Donner le nombre de produits par commande.

```
SELECT NCmd, COUNT(NP) NbProd
FROM Ligne_Cmd
GROUP BY NCmd;
```

Ligne_Cmd

NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

NCmd	NbProd
C001	3
C002	2
C003	1
C004	4
C005	2

✓ Donner les commandes dont le nombre de produits dépasse 2.

```
SELECT NCmd, COUNT(NP) NbProd
FROM Ligne_Cmd
GROUP BY NCmd
HAVING COUNT(NP) > 2;
```

Ligne_Cmd

NCmd	NP	Qte
C001	P001	250
C001	P004	300
C001	P006	100
C002	P002	200
C002	P007	550
C003	P001	50
C004	P002	100
C004	P004	150
C004	P005	70
C004	P008	90
C005	P001	650
C005	P002	100

NCmd	NbProd
C001	3
C004	4

Exercice 1 :

L'agence de voyages **Tunisia Tours** organise des voyages avec des visites d'endroits touristiques.

Le schéma relationnel relatif à son système d'information est décrit par les relations suivantes :

- Client (NoClient, NomClient, Adresse)
- Voyages (NoVoyage, VilleDépart, VilleArrivée, DateDépart, DateRetour, Prix)
- Visite (NoVisite, Endroit)
- Inscription (#NoVoyage, #NoClient, DateInscription)
- Programmes (#NoVisite, #NoVoyage, DateVisite)

Déduire, à partir de ce modèle relationnel, le modèle graphique Tunisia Tours. Préciser les entités, les associations et les cardinalités correspondantes.

Exercice 2 :

On donne la représentation textuelle simplifiée d'une base de données 'Etudiants' concernant un cycle de formation destiné à des étudiants.

ETUDIANT (CodeEt, NomEt, DatnEt**)**

MATIERE (CodeMat, NomMat, CoefMat**)**

NOTE (CodeEt#, CodeEns#, Note**)**

ENSEIGNANT (CodeEns, NomEns, GradeEns, CodeMat#**)**

Intitulé	Libellé
CodeEt	Code de l'étudiant
NomEt	Nom de l'étudiant
DatnEt	Date de naissance de l'étudiant
CodeMat	Code de la matière
NomMat	Nom de la matière
CoefMat	Coefficient de la matière
Note	Note obtenue par l'étudiant dans une matière
CodeEns	Code de l'enseignant
NomEns	Nom de l'enseignant
GradeEns	Grade de l'enseignant (Grd1, Grd2, …)

Écrire les requêtes SQL permettant de :

1. Créer la base de données « Etudiants » puis toutes les tables de la base en respectant toutes les contraintes.
2. Afficher les informations relatives aux étudiants (Code, Nom et Date de naissance) selon l'ordre alphabétique croissant du nom.
3. Afficher les noms et les grades des enseignants de la matière dont le nom est **'BD'**.
4. Afficher la liste distincte formée des noms et les coefficients des différentes matières qui sont enseignées par des enseignants de grade **'Grd3'.**
5. Afficher le nombre d'enseignants de la matière dont le nom est **'TIC'**.

Exercice 3 :

Dans la colonne Commande SQL, placer la commande adéquate et cocher la famille de langage SQL correspondante (LDD, LMD ou LCD).

ALTER, REVOKE, DELETE, DROP, TABLE, INSERT, CREATE TABLE, UPDATE, GRANT, SELECT, CREATE USER

Commande SQL	Rôle	LDD	LMD	LCD
	Permet la suppression d'une table tout entière			
	Permet la suppression de droits d'un utilisateur			
	Permet la modification de la structure d'une table			
	Permet l'insertion des données dans une table			
	Permet l'attribution de droits à un utilisateur			
	Permet de mettre à jour les données d'une table			
	Permet la suppression des données d'une table			
	Permet la recherche des données			
	Permet la création de la structure d'une table			

Exercice 4 :

Pour chaque question, cochez la bonne réponse :

1. En SQL, quelle commande utiliser pour sélectionner une ou plusieurs colonnes ?

- ☐ GET
- ☐ FILTER
- ☐ SELECT
- ☐ EXTRACT

2. Que veut dire l'abréviation SQL ?

- ☐ Strong Query Language
- ☐ Strong Question Language
- ☐ Structured Question Language
- ☐ Structured Query Language

3. Vous souhaitez accorder des privilèges à l'utilisateur Ahmed qui permettront à Ahmed de mettre à jour les données de la table EMPLOYES. Quel type de privilèges accorderez-vous à Ahmed ?

- ☐ Privilèges utilisateur
- ☐ Privilèges objet
- ☐ Privilèges système
- ☐ Privilèges d'administrateur

4. Avec SQL, comment sélectionner tous les enregistrements d'une table nommée « Personnes » où la valeur de la colonne « Prénom » commence par un « a » ?

- ☐ SELECT * FROM Personnes WHERE Prénom LIKE 'a%' ;
- ☐ SELECT * FROM Personnes WHERE Prénom='%a%' ;
- ☐ SELECT * FROM Personnes WHERE Prénom LIKE '%a' ;
- ☐ SELECT * FROM Personnes WHERE Prénom='a' ;

5. Avec SQL, comment supprimer les enregistrements dont le « Prénom » est « Ali » dans la table des personnes ?

- ☐ DELETE Prénom='Ali' FROM Personnes ;
- ☐ DELETE ROW Prénom='Ali' FROM Personnes ;
- ☐ DELETE FROM Personnes WHERE Prénom= 'Ali' ;
- ☐ DROP Prénom= 'Ali' FROM Personnes ;

Exercice 5 :

Soit la représentation textuelle d'une base de données simplifiée qui gère les espèces végétales d'une pépinière.

- TYPEPLANTE (IdTyp, LibTyp)
- CATEGORIE (IdCat, LibCat)
- PLANTE (IdPlan, NomPlan, Couleur, Expo, PrixUnit, IdTyp#, IdCat#)
- PARCELLE (IdParc, Surface)
- PLANTER (IdParc#, IdPlan#, Qte)

Écrire les requêtes SQL pour :

1. Afficher les noms des plantes de couleur rouge et dont l'exposition est Mi-Ombre.
2. Afficher toutes les plantes (Nom, couleur et prix) de catégorie 'Plante de jardin'.
3. Afficher les noms par ordre alphabétique des plantes qui se trouvent sur la parcelle dont l'identifiant est 'PA10'.
4. Mettre à jour la table concernée par la livraison de 1000 unités de plantes identifiées par le code 'PL55' à partir de la parcelle d'identifiant 'PA105'.
5. Afficher par couleur (Couleur et quantité totale) des plantes disponibles en quantité totale supérieure ou égale à 100 unités.
6. Ajouter une contrainte d'intégrité de domaine permettant d'autoriser uniquement les valeurs 'O', 'M' ou 'S' dans la colonne Expo de la table PLANTE.
7. Pour éviter le changement des résultats, l'administrateur de la base de données décide de retirer de l'utilisateur User1 le droit de modification sur la table PLANTE. Sachant que l'utilisateur User1 est déjà créé et bénéficie de tous les droits, écrire une requête SQL permettant d'effectuer cette tâche.