Université Cote d'Azur / FDS

UE3 - Immersion BD et SQL Oracle

Projet : Gestion d'un cabinet médical

Etudiants :

- **PIERRE** Bob Charlemagne
- **DOUILLY** Rodely
- **MILORME** Pierre Rubens
- **SURLIN** Djimy

Professeur : WEDTER Jérome

28 février 2024

Table des matières

Description du sujet

Cette application pour un cabinet médical permet la gestion complète des patients, des médecins, des rendez-vous, des consultations, des prescriptions, des examens et de la facturation. Les patients peuvent être enregistrés avec leurs détails personnels, tandis que les médecins sont répertoriés avec leur spécialité respective. Les rendez-vous entre patients et médecins sont programmés, enregistrés dans la table RENDEZ-VOUS. Chaque consultation est consignée dans la table CONSULTATION, associée à un patient, un médecin et une facture. Les prescriptions médicales sont enregistrées dans la table PRESCRIPTION, liées à la consultation correspondante. Les détails des examens sont stockés dans la table EXAMEN, également liés à la consultation. Chaque consultation génère une facture, enregistrée dans la table FACTURE avec le montant total à payer. Des contraintes de clé étrangère garantissent l'intégrité des données et les relations entre les tables.

En résumé, cette application fournit un système complet pour gérer les opérations quotidiennes d'un cabinet médical, optimisant le suivi des patients et la gestion des consultations et des facturations.

1 Spécification, Analyse et conception

1.1 Description textuelle des requêtes de mise à jour

1.1.1 2 requêtes impliquant 1 table

1.
```
UPDATE PATIENT
SET DATE_NAISSAINCE = TO_DATE('22-AUG-1986','DD-MON-YYYY')
WHERE EMAIL = 'thomas.leclerc@email.com';
```

➔ Analyse de la requête

Cette requête SQL est composée de trois parties principales :

- **UPDATE PATIENT :** cette partie indique que la requête souhaite modifier des données dans la table PATIENT.
- **SET DATE_NAISSAINCE = TO_DATE('22-AUG-1986', 'DD-MON-YYYY') :** cette partie définit les modifications à apporter. La colonne DATE_NAISSAINCE sera mise à jour avec la valeur 22-AUG-1986 formatée en date selon le format DD-MON-YYYY.
- **WHERE EMAIL = 'thomas.leclerc@email.com' :** cette partie précise les lignes de la table PATIENT qui seront affectées par la modification. Seules les lignes où la colonne EMAIL est égale à thomas.leclerc@email.com seront mises à jour.

➔ Fonctionnement de la requête

La requête recherche d'abord toutes les lignes de la table PATIENT où la colonne EMAIL est égale à thomas.leclerc@email.com. Pour chaque ligne trouvée, la valeur de la colonne DATE_NAISSAINCE est remplacée par la date 22-AUG-1986 formatée selon le format DD-MON-YYYY. Le nombre de lignes affectées par la modification est ensuite renvoyé.

➔ Résumé des effets de la requête

Cette requête modifie la date de naissance du patient dont l'adresse mail est thomas.leclerc@email.com. La nouvelle date de naissance sera 22-AUG-1986. Seules les lignes correspondant à l'adresse mail spécifiée seront affectées.

2.
```
UPDATE PATIENT
SET ADRESSE = '90, DELMAS 75'
WHERE ID_PATIENT_ = 1;
```

➔ Analyse de la requête

Cette requête SQL est composée de trois parties principales :

- **UPDATE PATIENT :** cette partie indique que la requête souhaite modifier des données dans la table PATIENT.
- **SET ADRESSE = '90, DELMAS 75' :** cette partie définit les modifications à apporter. La colonne ADRESSE sera mise à jour avec la valeur '90, DELMAS 75'.
- **WHERE ID_PATIENT_ = 1 :** cette partie précise les lignes de la table PATIENT qui seront affectées par la modification. Seule la ligne où la colonne ID_PATIENT_ est égale à 1 sera mise à jour.

➔ Fonctionnement de la requête :

La requête recherche d'abord la ligne de la table PATIENT où la colonne ID_PATIENT_ est

égale à 1. La valeur de la colonne ADRESSE de cette ligne est remplacée par la valeur '90, DELMAS 75'. Le nombre de lignes affectées par la modification est ensuite renvoyé (ici, 1).
➔ Résumé des effets de la requête :
Cette requête modifie l'adresse du patient dont l'identifiant est 1. La nouvelle adresse sera '90, DELMAS 75'. Seule la ligne correspondant à l'identifiant spécifié sera affectée.

1.1.2 2 requêtes impliquant 2 tables

1. ```
   UPDATE (SELECT P.ID_PATIENT_, P.NOM, P.PRENOM,
   R.DATE_RENDEZ_VOUS FROM PATIENT P
   JOIN RENDEZ_VOUS R
   ON P.ID_PATIENT_ = R.ID_PATIENT_
   WHERE R.DATE_RENDEZ_VOUS BETWEEN '14-FEB-2024' AND '18-FEB-2024') X
   SET X.DATE_RENDEZ_VOUS = '01-MAR-24';
   ```

   ➔ Analyse de la requête
   Cette requête SQL est composée de plusieurs parties imbriquées :

   — **UPDATE :** Cette partie indique que la requête souhaite modifier des données.
   — **(SELECT P.ID_PATIENT_, P.NOM,P.PRENOM, R.DATE_RENDEZ_VOUS FROM PATIENT P JOIN RENDEZ_VOUS R ON P.ID_PATIENT_ = R.ID_PATIENT_ WHERE R.DATE_RENDEZ_VOUS BETWEEN '14-FEB-2024' AND '18-FEB-2024') :** Cette partie définit la table et les colonnes à modifier, ainsi que les lignes concernées par la modification.
   — **SET DATE_RENDEZ_VOUS = '01-MAR-24' :** Cette partie définit la nouvelle valeur à attribuer à la colonne DATE_RENDEZ_VOUS.

   ➔ Fonctionnement de la requete
   La requête exécute d'abord la sous-requête entre parenthèses. Cette sous-requête sélectionne les colonnes ID_PATIENT_, NOM, PRENOM et DATE_RENDEZ_VOUS des tables PATIENT et RENDEZ_VOUS pour les patients dont le rendez-vous est entre le 14-FEB-2024 et le 18-FEB-2024. La requête utilise ensuite le résultat de la sous-requête comme table virtuelle pour la mise à jour.

   Pour chaque ligne de la table virtuelle, la valeur de la colonne DATE_RENDEZ_VOUS est mise à jour avec la valeur 01-MAR-24.

   Le nombre de lignes affectées par la modification est ensuite renvoyé.

   ➔ Résumé des effets de la requête :

   Cette requête modifie la date de rendez-vous des patients dont le rendez-vous est prévu entre le 14-FEB-2024 et le 18-FEB-2024. La nouvelle date de rendez-vous sera le 01-MAR-24. Seules les lignes correspondant aux dates spécifiées seront affectées.

   ➔ Remarques

   La sous-requête permet de filtrer les patients dont la date de rendez-vous est dans la période spécifiée. La clause WHERE de la sous-requête est obligatoire pour limiter les lignes à modifier. La jointure entre les tables PATIENT et RENDEZ_VOUS est nécessaire pour obtenir les informations des deux tables.
   ```

En résumé, cette requête SQL permet de modifier la date de rendez-vous des patients dont le rendez-vous est prévu entre le 14-FEB-2024 et le 18-FEB-2024 dans la table virtuelle issue de la jointure entre PATIENT et RENDEZ_VOUS. La nouvelle date de rendez-vous sera le 01-MAR-24.

2.
```
UPDATE (SELECT * FROM FACTURE F , PATIENT P
WHERE F.ID_PATIENT_ = P.ID_PATIENT_
AND F.MONTANT_TOTAL < 200)
SET MONTANT_TOTAL = MONTANT_TOTAL + MONTANT_TOTAL * 0.1;
```

➔ Description

Cette requête vise à mettre à jour le montant total de certaines factures dans une base de données.

➔ Fonctionnement

— **Sélection des factures :**
La requête commence par la sous-requête SELECT * FROM FACTURE F, PATIENT P. Cela signifie qu'elle sélectionne toutes les colonnes des tables FACTURE (F) et PATIENT (P). La clause WHERE associée à la sous-requête précise les conditions de sélection : F.ID_PATIENT_ = P.ID_PATIENT_ : cette condition garantit que seules les factures et les patients correspondants sont sélectionnés. En d'autres termes, la facture doit appartenir au patient identifié. F.MONTANT_TOTAL ¡ 200 : cette condition filtre les factures dont le montant total est inférieur à 200. Seules ces factures seront affectées par la mise à jour.

— **Mise à jour du montant total :**
La clause UPDATE intervient ensuite. Elle indique que l'on souhaite modifier des données dans une table, mais la table à modifier n'est pas explicitement mentionnée ici. En effet, la sous-requête précédente agit comme une table virtuelle contenant les factures sélectionnées. La clause SET définit la modification à apporter. Dans ce cas, MONTANT_TOTAL = MONTANT_TOTAL + MONTANT_TOTAL * 0.1 signifie que le montant total de chaque facture sélectionnée sera augmenté de 10%. L'augmentation est calculée en multipliant le montant total d'origine par 1.1 (100% + 10%).

En résumé, cette requête parcourt toutes les factures dont le montant total est inférieur à 200 et pour lesquelles un patient correspondant existe. Pour chacune de ces factures, elle augmente le montant total de 10%.

1.1.3 2 requêtes impliquant plus de 2 tables

1.
```
UPDATE PRESCRIPTION R
SET R.DETAILS_PRESCRIPTION = 'Zinoboost'
WHERE R.ID_CONSULTATION_ IN (
    SELECT C.ID_CONSULTATION_
    FROM CONSULTATION C
    WHERE DATE_CONSULTATION = TO_DATE('05/02/2024','DD/MM/YYYY')
        AND C.ID_PATIENT_ IN (
        SELECT P.ID_PATIENT_
```

```
        FROM PATIENT P
        WHERE P.ID_PATIENT_ = 1
    )
);
```

➔ Description

Cette requête SQL met à jour les enregistrements dans la table PRESCRIPTION.

Voici une explication de la requête :

- **UPDATE PRESCRIPTION R :** Cela indique que nous allons mettre à jour des données dans la table PRESCRIPTION et nous l'identifions avec l'alias R.
- **SET R.DETAILS_PRESCRIPTION = 'Zinoboost' :** Cela indique que nous allons mettre à jour la colonne DETAILS_PRESCRIPTION de la table PRESCRIPTION avec la valeur 'Zinoboost'.
- **WHERE R.ID_CONSULTATION_ IN (...) :** C'est une clause de restriction qui spécifie que la mise à jour ne s'applique qu'aux enregistrements dans la table PRESCRIPTION où la colonne ID_CONSULTATION_ correspond à certaines valeurs.
- **(SELECT C.ID_CONSULTATION_ FROM CONSULTATION C WHERE DATE_CONSULTATION = TO_DATE('05/02/2024','DD/MM/YYYY') AND C.ID_PATIENT_ IN (...)) :** C'est une sous-requête qui sélectionne les identifiants de consultation (ID_CONSULTATION_) de la table CONSULTATION où la date de consultation (DATE_CONSULTATION) est égale au 5 février 2024 et où l'identifiant de patient (ID_PATIENT_) correspond à certaines valeurs.
- **(SELECT P.ID_PATIENT_ FROM PATIENT P WHERE P.ID_PATIENT_ = 1) :** C'est une autre sous-requête qui sélectionne l'identifiant de patient (ID_PATIENT_) de la table PATIENT où l'identifiant de patient est égal à 1.

En résumé, cette requête met à jour la colonne DETAILS_PRESCRIPTION de la table PRESCRIPTION pour les enregistrements associés à des consultations de patients ayant un identifiant de patient égal à 1 et où la date de consultation est le 5 février 2024. La nouvelle valeur de la colonne DETAILS_PRESCRIPTION sera 'Zinoboost'.

2.
```
UPDATE EXAMEN  R
SET R.DETAILS_EXAMEN  = 'HAC1'
WHERE R.ID_CONSULTATION_ IN (
    SELECT C.ID_CONSULTATION_
    FROM CONSULTATION C
    WHERE DATE_CONSULTATION = TO_DATE('05/02/2024','DD/MM/YYYY')
    AND C.ID_PATIENT_ IN (
        SELECT P.ID_PATIENT_
        FROM PATIENT P
        WHERE P.ID_PATIENT_ = 1
    )
);
```

➔ Description :
Cette requête SQL Oracle met à jour les enregistrements dans la table EXAMEN.

Voici une explication de la requête :

— **UPDATE EXAMEN R :** Cela indique que nous allons mettre à jour des données dans la table EXAMEN et nous l'identifions avec l'alias R.

— **SET R.DETAILS_EXAMEN =** 'HAC1' : Cela indique que nous allons mettre à jour la colonne DETAILS_EXAMEN de la table EXAMEN avec la valeur 'HAC1'.

— **WHERE R.ID_CONSULTATION_ IN (...) :** C'est une clause de restriction qui spécifie que la mise à jour ne s'applique qu'aux enregistrements dans la table EXAMEN où la colonne ID_CONSULTATION_ correspond à certaines valeurs.

— **(SELECT C.ID_CONSULTATION_ FROM CONSULTATION C WHERE DATE_CONSULTATION =**
TO_DATE('05/02/2024',
'DD/MM/YYYY') AND C.ID_PATIENT_ IN (...)) : C'est une sous-requête qui sélectionne les identifiants de consultation (ID_CONSULTATION_) de la table CONSULTATION où la date de consultation (DATE_CONSULTATION) est égale au 5 février 2024 et où l'identifiant de patient (ID_PATIENT_) correspond à certaines valeurs.

— (SELECT P.ID_PATIENT_ FROM PATIENT P WHERE P.ID_PATIENT_ = 1) : C'est une autre sous-requête qui sélectionne l'identifiant de patient (ID_PATIENT_) de la table PATIENT où l'identifiant de patient est égal à 1.

En résumé, cette requête met à jour la colonne 'DETAILS_EXAMEN' de la table 'EXAMEN' pour les enregistrements associés à des consultations de patients ayant un identifiant de patient égal à 1 et où la date de consultation est le 5 février 2024. La nouvelle valeur de la colonne DETAILS_EXAMEN sera 'HAC1'.

1.2 Description textuelle des requêtes de suppression

1.2.1 2 requêtes impliquant 1 table

1.
```
DELETE EXAMEN
WHERE DATE_EXAMEN = TO_DATE('05-02-24','DD-MM-YY');
```

➔ Description
Cette requête SQL est une commande DELETE qui supprime des lignes de la table EXAMEN en fonction d'un critère spécifique.

➔ Voici une explication détaillée de chaque élément de la requête :

— **DELETE :** Il s'agit du mot-clé qui indique à Oracle que vous souhaitez supprimer des données d'une table.

— **EXAMEN :** C'est le nom de la table à partir de laquelle vous souhaitez supprimer des données.

— **WHERE :** C'est une clause conditionnelle qui spécifie les conditions que les lignes à supprimer doivent satisfaire. Seules les lignes qui correspondent à cette condition seront supprimées.

— **DATE_EXAMEN :** C'est le nom de la colonne sur laquelle la condition est basée. Dans ce cas, la condition porte sur la colonne DATE_EXAMEN.

— **TO_DATE('05-02-24','DD-MM-YY') :** C'est une fonction Oracle qui convertit une chaîne de caractères en un type de données DATE. Dans cet exemple, la chaîne '05-02-24' est convertie en une date au format 'DD-MM-YY', ce qui signifie que la date est le 5 février 2024.

En résumé, cette requête supprime toutes les lignes de la table EXAMEN où la valeur de la colonne DATE_EXAMEN est égale à la date du 5 février 2024.

2.
```
DELETE FROM PRESCRIPTION
WHERE DATE_PRESCRIPTION  = TO_DATE('05-02-24','DD-MM-YY');
```

➔ Description

Cette requête SQL est une commande DELETE qui supprime des lignes de la table PRESCRIPTION en fonction d'un critère spécifique.

➔ Voici une explication détaillée de chaque élément de la requête :

— **DELETE :** Il s'agit du mot-clé qui indique à Oracle que vous souhaitez supprimer des données d'une table.

— **FROM PRESCRIPTION :** PRESCRIPTION est le nom de la table à partir de laquelle vous souhaitez supprimer des données. Le mot-clé FROM est utilisé pour spécifier la table à partir de laquelle les données seront supprimées.

— **WHERE :** C'est une clause conditionnelle qui spécifie les conditions que les lignes à supprimer doivent satisfaire. Seules les lignes qui correspondent à cette condition seront supprimées.

— **DATE_PRESCRIPTION :** C'est le nom de la colonne sur laquelle la condition est basée. Dans ce cas, la condition porte sur la colonne DATE_PRESCRIPTION.

— **TO_DATE('05-02-24','DD-MM-YY') :** C'est une fonction Oracle qui convertit une chaîne de caractères en un type de données DATE. Dans cet exemple, la chaîne '05-02-24' est convertie en une date au format 'DD-MM-YY', ce qui signifie que la date est le 5 février 2024.

En résumé, cette requête supprime toutes les lignes de la table PRESCRIPTION où la valeur de la colonne DATE_PRESCRIPTION est égale à la date du 5 février 2024.

1.2.2 2 requêtes impliquant 2 tables

1.
```
DELETE FROM RENDEZ_VOUS
WHERE ID_PATIENT_ IN (
    SELECT ID_PATIENT_ FROM PATIENT
    WHERE EMAIL = 'thomas.leclerc@email.com'
) ;
```

➔ Description

Cette requête SQL Oracle est une commande DELETE qui supprime des lignes de la table RENDEZ_VOUS en fonction d'un critère spécifique.

➔ Voici une explication détaillée de chaque élément de la requête :

— **DELETE :** Il s'agit du mot-clé qui indique à Oracle que vous souhaitez supprimer des données d'une table.

— **FROM RENDEZ_VOUS :** RENDEZ_VOUS est le nom de la table à partir de laquelle vous souhaitez supprimer des données. Le mot-clé FROM est utilisé pour spécifier la table à partir de laquelle les données seront supprimées.

— **WHERE :** C'est une clause conditionnelle qui spécifie les conditions que les lignes à supprimer doivent satisfaire. Seules les lignes qui correspondent à cette condition seront supprimées.

— **ID_PATIENT_ IN (SELECT ID_PATIENT_ FROM PATIENT WHERE EMAIL = 'thomas.leclerc@email.com') :** C'est une sous-requête qui sélectionne les identifiants de patients à partir de la table "PATIENT" où l'adresse e-mail est égale à 'thomas.leclerc@email.com'. Cette sous-requête est utilisée comme condition pour la suppression des rendez-vous. Seuls les rendez-vous des patients dont l'adresse e-mail correspondante est 'thomas.leclerc@email.com' seront supprimés.

En résumé, cette requête supprime tous les rendez-vous des patients dont l'adresse e-mail est 'thomas.leclerc@email.com'.

2.
```
DELETE FROM RENDEZ_VOUS
WHERE ID_PATIENT_ IN (
    SELECT ID_PATIENT_ FROM PATIENT
    WHERE ID_PATIENT_ = 3
) ;
```

➔ Description

Cette requête SQL est une commande DELETE qui supprime des lignes de la table RENDEZ_VOUS en fonction d'un critère spécifique.

➔ Voici une explication détaillée de chaque élément de la requête :

— **DELETE :** Il s'agit du mot-clé qui indique à Oracle que vous souhaitez supprimer des données d'une table.

— **FROM RENDEZ_VOUS :** RENDEZ_VOUS est le nom de la table à partir de laquelle vous souhaitez supprimer des données. Le mot-clé FROM est utilisé pour spécifier la table à partir de laquelle les données seront supprimées.

— **WHERE :** C'est une clause conditionnelle qui spécifie les conditions que les lignes à supprimer doivent satisfaire. Seules les lignes qui correspondent à cette condition seront supprimées.

— **ID_PATIENT_ IN (SELECT ID_PATIENT_ FROM PATIENT WHERE ID_PATIENT_ = 3) :** C'est une sous-requête qui sélectionne les identifiants de patients à partir de la table "PATIENT" où l'identifiant de patient est égal à 3. Cette sous-requête est utilisée comme condition pour la suppression des rendez-vous. Seuls les rendez-vous associés au patient dont l'identifiant est 3 seront supprimés.

En résumé, cette requête supprime tous les rendez-vous du patient dont l'identifiant est 3.

1.2.3 2 requêtes impliquant plus de 2 tables

1. ```
   DELETE FROM EXAMEN
       WHERE ID_CONSULTATION_ IN (
           SELECT C.ID_CONSULTATION_
           FROM CONSULTATION C
           WHERE DATE_CONSULTATION = TO_DATE('05/02/2024','DD/MM/YYYY')
           AND C.ID_PATIENT_ IN (
               SELECT P.ID_PATIENT_
               FROM PATIENT P
               WHERE P.ID_PATIENT_ = 1
       )
   );
   ```

   ➔ Description

   Cette requête SQL Oracle est une commande DELETE qui supprime des lignes de la table EXAMEN en fonction de certains critères spécifiques.

   ➔ Voici une explication détaillée de chaque élément de la requête :

   - **DELETE :** Il s'agit du mot-clé qui indique à Oracle que vous souhaitez supprimer des données d'une table.
   - **FROM EXAMEN :** EXAMEN est le nom de la table à partir de laquelle vous souhaitez supprimer des données. Le mot-clé FROM est utilisé pour spécifier la table à partir de laquelle les données seront supprimées.
   - **WHERE :** C'est une clause conditionnelle qui spécifie les conditions que les lignes à supprimer doivent satisfaire. Seules les lignes qui correspondent à cette condition seront supprimées.
   - **ID_CONSULTATION_ IN (...) :** C'est une condition qui filtre les lignes à supprimer en se basant sur les identifiants de consultation. La sous-requête entre parenthèses sélectionne les identifiants de consultation à partir de la table CONSULTATION.
   - **SELECT C.ID_CONSULTATION_ FROM CONSULTATION C WHERE DATE_CONSULTATION =**
     TO_DATE('05/02/2024','DD/MM/YYYY')
     AND C.ID_PATIENT_ IN (...) : C'est une sous-requête qui sélectionne les identifiants de consultation à partir de la table "CONSULTATION" en filtrant les consultations qui ont eu lieu le 5 février 2024 et qui sont associées à un patient spécifique. La sous-requête imbriquée sélectionne d'abord l'identifiant du patient à partir de la table "PATIENT".
   - **TO_DATE('05/02/2024','DD/MM/YYYY') :** C'est une fonction Oracle qui convertit une chaîne de caractères en un type de données DATE. Dans cet exemple, la chaîne '05/02/2024' est convertie en une date au format 'DD/MM/YYYY', ce qui correspond au 5 février 2024.

   En résumé, cette requête supprime tous les examens associés à une consultation qui a eu lieu le 5 février 2024 et qui est liée à un patient dont l'identifiant est 1.

2. ```
   DELETE FROM EXAMEN
   ```

```
WHERE ID_CONSULTATION_ IN (
    SELECT C.ID_CONSULTATION_
    FROM CONSULTATION C
    WHERE DATE_CONSULTATION = TO_DATE('06/02/2024','DD/MM/YYYY')
    AND C.ID_PATIENT_ IN (
        SELECT P.ID_PATIENT_
        FROM PATIENT P
        WHERE P.ID_PATIENT_ = 2
    )
);
```

➔ Description

Cette requête SQL Oracle est une commande DELETE qui supprime des lignes de la table EXAMEN en fonction de certains critères spécifiques.

➔ Voici une explication détaillée de chaque élément de la requête :

- **DELETE :** Il s'agit du mot-clé qui indique à Oracle que vous souhaitez supprimer des données d'une table.
- **FROM EXAMEN :** EXAMEN est le nom de la table à partir de laquelle vous souhaitez supprimer des données. Le mot-clé FROM est utilisé pour spécifier la table à partir de laquelle les données seront supprimées.
- **WHERE :** C'est une clause conditionnelle qui spécifie les conditions que les lignes à supprimer doivent satisfaire. Seules les lignes qui correspondent à cette condition seront supprimées.
- **ID_CONSULTATION_ IN (...) :** C'est une condition qui filtre les lignes à supprimer en se basant sur les identifiants de consultation. La sous-requête entre parenthèses sélectionne les identifiants de consultation à partir de la table CONSULTATION.
- **SELECT C.ID_CONSULTATION_ FROM CONSULTATION C WHERE DATE_CONSULTATION = TO_DATE('06/02/2024','DD/MM/YYYY') AND C.ID_PATIENT_ IN (...) :** C'est une sous-requête qui sélectionne les identifiants de consultation à partir de la table CONSULTATION en filtrant les consultations qui ont eu lieu le 6 février 2024 et qui sont associées à un patient spécifique. La sous-requête imbriquée sélectionne d'abord l'identifiant du patient à partir de la table PATIENT.
- **TO_DATE('06/02/2024','DD/MM/YYYY') :** C'est une fonction Oracle qui convertit une chaîne de caractères en un type de données DATE. Dans cet exemple, la chaîne '06/02/2024' est convertie en une date au format 'DD/MM/YYYY', ce qui correspond au 6 février 2024.

En résumé, cette requête supprime tous les examens associés à une consultation qui a eu lieu le 6 février 2024 et qui est liée à un patient dont l'identifiant est 2.

1.3 Description textuelle des requêtes de consultation

1.3.1 5 requêtes impliquant 1 table dont 1 avec un group By et une avec un Order By

1. `SELECT * FROM PATIENT;`

   ➔ Description
   Cette requête SQL Oracle est une commande SELECT qui récupère toutes les lignes et toutes les colonnes de la table PATIENT.

   Voici une explication détaillée de chaque élément de la requête :

   — **SELECT * :** C'est une clause SELECT qui spécifie les colonnes que vous souhaitez récupérer dans la requête. L'astérisque (*) est un opérateur de sélection qui signifie "toutes les colonnes". Ainsi, cette partie de la requête récupère toutes les colonnes de la table PATIENT.

   — **FROM PATIENT :** PATIENT est le nom de la table à partir de laquelle vous souhaitez récupérer les données. Le mot-clé FROM est utilisé pour spécifier la table à partir de laquelle les données seront extraites.

   En résumé, cette requête récupère toutes les informations (toutes les colonnes) stockées dans la table PATIENT, ce qui signifie qu'elle retournera toutes les lignes de cette table.

2. `SELECT * FROM CONSULTATION;`

   ➔ Description
   Cette requête SQL Oracle est une commande SELECT qui récupère toutes les lignes et toutes les colonnes de la table CONSULTATION.

   Voici une explication détaillée de chaque élément de la requête :

   — **SELECT * :** C'est une clause SELECT qui spécifie les colonnes que vous souhaitez récupérer dans la requête. L'astérisque (*) est un opérateur de sélection qui signifie "toutes les colonnes". Ainsi, cette partie de la requête récupère toutes les colonnes de la table CONSULTATION.

   — **FROM CONSULTATION :** CONSULTATION est le nom de la table à partir de laquelle vous souhaitez récupérer les données. Le mot-clé FROM est utilisé pour spécifier la table à partir de laquelle les données seront extraites.

   En résumé, cette requête récupère toutes les informations (toutes les colonnes) stockées dans la table CONSULTATION, ce qui signifie qu'elle retournera toutes les lignes de cette table.

3. ```
   SELECT COUNT(*), DETAILS_EXAMEN
   FROM EXAMEN
   GROUP BY DETAILS_EXAMEN;
   ```

   ➔ Description
   Cette requête SQL Oracle est une commande SELECT qui récupère le nombre de lignes pour chaque valeur unique de la colonne DETAILS_EXAMEN de la table EXAMEN.

   Voici une explication détaillée de chaque élément de la requête :
   ```

— **SELECT COUNT(*) :** C'est une fonction d'agrégation qui compte le nombre de lignes dans chaque groupe résultant de la requête GROUP BY. L'opérateur * signifie que toutes les lignes sont prises en compte dans le comptage. La virgule (,),c'est simplement un séparateur pour indiquer que nous voulons sélectionner une autre colonne dans notre résultat.

— **DETAILS_EXAMEN :** C'est le nom de la colonne pour laquelle nous voulons obtenir le compte des occurrences de chaque valeur. FROM EXAMEN : EXAMEN est le nom de la table à partir de laquelle les données sont extraites.

— **GROUP BY DETAILS_EXAMEN :** C'est une clause GROUP BY qui groupe les lignes en fonction des valeurs uniques dans la colonne DETAILS_EXAMEN. En d'autres termes, cela signifie que les lignes avec la même valeur dans la colonne DETAILS_EXAMEN seront regroupées ensemble.

Ainsi, cette requête retournera le nombre de lignes pour chaque valeur unique de la colonne "DETAILS_EXAMEN" de la table "EXAMEN".

4.
```
SELECT COUNT(*), DETAILS_EXAMEN
FROM EXAMEN
GROUP BY DETAILS_EXAMEN
ORDER BY DETAILS_EXAMEN ;
```

➔ Description

Cette requête SQL Oracle est une commande SELECT qui récupère le nombre de lignes pour chaque valeur unique de la colonne DETAILS_EXAMEN de la table EXAMEN.

Voici une explication détaillée de chaque élément de la requête :

— **SELECT COUNT(*) :** C'est une fonction d'agrégation qui compte le nombre de lignes dans chaque groupe résultant de la requête GROUP BY. L'opérateur * signifie que toutes les lignes sont prises en compte dans le comptage. La virgule (,)C'est simplement un séparateur pour indiquer que nous voulons sélectionner une autre colonne dans notre résultat.

— **DETAILS_EXAMEN :** C'est le nom de la colonne pour laquelle nous voulons obtenir le compte des occurrences de chaque valeur.

— **FROM EXAMEN :** EXAMEN est le nom de la table à partir de laquelle les données sont extraites.

— **GROUP BY DETAILS_EXAMEN :** C'est une clause GROUP BY qui groupe les lignes en fonction des valeurs uniques dans la colonne DETAILS_EXAMEN. En d'autres termes, cela signifie que les lignes avec la même valeur dans la colonne DETAILS_EXAMEN seront regroupées ensemble. Ainsi, cette requête retournera le nombre de lignes pour chaque valeur unique de la colonne "DETAILS_EXAMEN" de la table "EXAMEN".

5.
```
SELECT ID_PATIENT_, COUNT(*), SUM(MONTANT_TOTAL)
FROM FACTURE
GROUP BY ID_PATIENT_
ORDER BY ID_PATIENT_ ;
```

➔ Description

Cette requête SQL Oracle est une commande SELECT qui agrège les données de la table FACTURE en fonction de la colonne ID_PATIENT_.

Voici une explication détaillée de chaque élément de la requête :

- **SELECT ID_PATIENT_, COUNT(*), SUM(MONTANT_TOTAL) :** Cette partie de la requête spécifie les colonnes que nous voulons afficher dans le résultat de la requête.
- **ID_PATIENT_ :** C'est la colonne pour laquelle nous voulons grouper les données.
- **COUNT(*) :** C'est une fonction d'agrégation qui compte le nombre de lignes pour chaque groupe de données résultant du GROUP BY.
- **SUM(MONTANT_TOTAL) :** C'est une autre fonction d'agrégation qui calcule la somme des valeurs de la colonne MONTANT_TOTAL pour chaque groupe de données résultant du GROUP BY.
- **FROM FACTURE :** FACTURE est le nom de la table à partir de laquelle les données sont extraites.
- **GROUP BY ID_PATIENT_ :** C'est une clause GROUP BY qui groupe les lignes en fonction des valeurs uniques dans la colonne ID_PATIENT_. En d'autres termes, cela signifie que les lignes avec la même valeur dans la colonne ID_PATIENT_ seront regroupées ensemble.
- **ORDER BY ID_PATIENT_ :** C'est une clause ORDER BY qui trie les résultats de la requête par ordre croissant des valeurs de la colonne "ID_PATIENT_".

En résumé, cette requête retournera le nombre de factures (COUNT(*)) et la somme des montants totaux (SUM(MONTANT_TOTAL)) pour chaque patient (ID_PATIENT_), avec les résultats triés par ID_PATIENT_ dans l'ordre croissant.

1.3.2 5 requêtes impliquant 2 tables avec jointures internes dont 1 externe + 1 group by + 1 tri

1.
```
SELECT * FROM CONSULTATION C
INNER JOIN PATIENT P
ON C.ID_PATIENT_ = P.ID_PATIENT_;
```

➔ Description

Cette requête SQL Oracle est une commande SELECT qui effectue une jointure entre deux tables, CONSULTATION (aliasée par C) et PATIENT (aliasée par P), en utilisant la clé étrangère ID_PATIENT_ comme critère de jointure.

Voici une explication détaillée de chaque élément de la requête :

- **SELECT * :** Cette clause spécifie les colonnes que vous souhaitez récupérer dans le résultat de la requête. L'astérisque (*) signifie toutes les colonnes. Par conséquent, cette requête récupérera toutes les colonnes de la table CONSULTATION et de la table PATIENT après la jointure.
- **FROM CONSULTATION C :** CONSULTATION est le nom de la première table à partir de laquelle les données sont extraites. L'alias C est utilisé pour identifier la table CONSULTATION dans la requête et faciliter la référence à ses colonnes.

— **INNER JOIN PATIENT P :** C'est la clause de jointure qui spécifie que nous voulons joindre la table PATIENT à la table CONSULTATION. PATIENT est le nom de la deuxième table que nous voulons joindre, et P est l'alias utilisé pour identifier cette table dans la requête.

— **ON C.ID_PATIENT_ = P.ID_PATIENT_ :** C'est la condition de jointure. Elle spécifie quelles colonnes des deux tables doivent être comparées pour déterminer les correspondances. Dans ce cas, nous comparons la colonne ID_PATIENT_ de la table CONSULTATION avec la colonne ID_PATIENT_ de la table PATIENT. Cela signifie que nous récupérerons uniquement les lignes où les valeurs de ces colonnes correspondent dans les deux tables.

En résumé, cette requête récupère toutes les colonnes des tables CONSULTATION et PATIENT pour lesquelles les valeurs de la colonne ID_PATIENT_ correspondent dans les deux tables. Cela permet de récupérer des informations sur les consultations ainsi que sur les patients associés à ces consultations.

2.
```
SELECT * FROM FACTURE F
INNER JOIN PATIENT P
ON F.ID_PATIENT_ = P.ID_PATIENT_;
```

➔ Description

Cette requête SQL Oracle est une commande SELECT qui effectue une jointure interne entre deux tables, FACTURE (aliasée par F) et PATIENT (aliasée par P), en utilisant la clé étrangère ID_PATIENT_ comme critère de jointure.

Voici une explication détaillée de chaque élément de la requête :

— **SELECT * :** Cette clause spécifie les colonnes que vous souhaitez récupérer dans le résultat de la requête. L'astérisque (*) signifie toutes les colonnes. Par conséquent, cette requête récupérera toutes les colonnes de la table "FACTURE" et de la table "PATIENT" après la jointure.

— **FROM FACTURE F :** FACTURE est le nom de la première table à partir de laquelle les données sont extraites. L'alias F est utilisé pour identifier la table FACTURE dans la requête et faciliter la référence à ses colonnes.

— **INNER JOIN PATIENT P :** C'est la clause de jointure qui spécifie que nous voulons joindre la table PATIENT à la table FACTURE. PATIENT est le nom de la deuxième table que nous voulons joindre, et P est l'alias utilisé pour identifier cette table dans la requête.

— **ON F.ID_PATIENT_ = P.ID_PATIENT_ :** C'est la condition de jointure. Elle spécifie quelles colonnes des deux tables doivent être comparées pour déterminer les correspondances. Dans ce cas, nous comparons la colonne ID_PATIENT_ de la table FACTURE avec la colonne ID_PATIENT_ de la table PATIENT. Cela signifie que nous récupérerons uniquement les lignes où les valeurs de ces colonnes correspondent dans les deux tables.

En résumé, cette requête récupère toutes les colonnes des tables FACTURE et PATIENT pour lesquelles les valeurs de la colonne ID_PATIENT_ correspondent dans les deux tables. Cela permet de récupérer des informations sur les factures ainsi que sur les patients associés

à ces factures.

3. ```
   SELECT * FROM CONSULTATION C
   INNER JOIN PATIENT P
   ON C.ID_PATIENT_ = P.ID_PATIENT_
   ORDER BY C;
   ```

   ➔ Description

   Cette requête SQL est une instruction SELECT qui récupère des données à partir de deux tables, CONSULTATION et PATIENT, en utilisant une jointure interne (INNER JOIN) entre elles.

   Voici une explication détaillée de la requête :

   — **SELECT * FROM CONSULTATION C :** Cela sélectionne toutes les colonnes de la table CONSULTATION et les renomme avec l'alias C. L'alias est utilisé pour référencer la table de manière abrégée dans le reste de la requête.

   — **INNER JOIN PATIENT P ON C.ID_PATIENT_ = P.ID_PATIENT_ :** C'est la clause de jointure. Elle relie les lignes des tables CONSULTATION et PATIENT où les valeurs de la colonne ID_PATIENT_ sont égales dans les deux tables. Cela relie les consultations aux patients correspondants en fonction de leur ID_PATIENT_.

   — **ORDER BY C.DATE_CONSULTATION :** Cette clause est utilisée pour trier les résultats de la requête par ordre croissant de la colonne DATE_CONSULTATION de la table CONSULTATION.

   En analyse, cette requête récupérera toutes les données des consultations (avec leurs patients correspondants) et les ordonnera par date de consultation. Cela peut être utile pour obtenir une liste chronologique des consultations et des détails des patients associés à chacune d'elles.

4. ```
   SELECT F.ID_PATIENT_, P.EMAIL, SUM(F.MONTANT_TOTAL) MONTANT_TOTAL
   FROM FACTURE F
   INNER JOIN PATIENT P
   ON F.ID_PATIENT_ = P.ID_PATIENT_
   GROUP BY F.ID_PATIENT_, P.EMAIL
   ORDER BY F.ID_PATIENT_, P.EMAIL;
   ```

   ➔ Description

   Cette requête SQL récupère des données à partir de deux tables, FACTURE et PATIENT, en utilisant une jointure interne (INNER JOIN) entre elles. Ensuite, elle effectue une agrégation en regroupant les données par ID_PATIENT_ et EMAIL du patient, puis elle calcule la somme des montants totaux de factures pour chaque patient.

   Voici une explication détaillée de la requête :

   — **SELECT F.ID_PATIENT_, P.EMAIL, SUM(F.MONTANT_TOTAL) AS MONTANT_TOTAL :** Cette partie de la requête sélectionne trois colonnes. F.ID_PATIENT_ est l'identifiant du patient à partir de la table FACTURE, P.EMAIL est l'adresse e-mail du patient à partir de la table PATIENT, et SUM(F.MONTANT_TOTAL) calcule la somme des montants totaux des factures pour chaque patient. L'alias MONTANT_TOTAL est utilisé pour désigner cette somme calculée.

— **FROM FACTURE F INNER JOIN PATIENT P ON F.ID_PATIENT_ = P.ID_PATIENT_ :** Cette clause spécifie les tables à partir desquelles les données seront sélectionnées. Elle indique également comment les données des deux tables doivent être jointes. Ici, les lignes de la table FACTURE sont jointes aux lignes correspondantes de la table PATIENT en utilisant les ID_PATIENT_ correspondants.

— **GROUP BY F.ID_PATIENT_, P.EMAIL :** Cette clause est utilisée pour regrouper les données en fonction de la colonne ID_PATIENT_ de la table FACTURE et de la colonne EMAIL de la table PATIENT. Cela signifie que la somme des montants totaux sera calculée pour chaque combinaison unique de ID_PATIENT_ et EMAIL.

— **ORDER BY F.ID_PATIENT_, P.EMAIL :** Cette clause ordonne les résultats de la requête en fonction des colonnes ID_PATIENT_ et EMAIL, dans l'ordre croissant.

En analyse, cette requête récupère la somme totale des montants des factures pour chaque patient, en les groupant par ID_PATIENT_ et EMAIL. Cela peut être utile pour obtenir une vue consolidée des montants facturés à chaque patient.

5.
```
SELECT * FROM CONSULTATION C
RIGHT JOIN PATIENT P
ON C.ID_PATIENT_ = P.ID_PATIENT_
ORDER BY C.DATE_CONSULTATION;
```

➔ Description

Cette requête SQL utilise une jointure externe droite (RIGHT JOIN) pour récupérer toutes les lignes de la table PATIENT et les lignes correspondantes de la table CONSULTATION lorsque la condition de jointure est satisfaite.

Voici une explication détaillée de la requête :

— **SELECT * FROM CONSULTATION C RIGHT JOIN PATIENT P ON C.ID_PATIENT_ = P.ID_PATIENT_ :** Cela sélectionne toutes les colonnes de la table CONSULTATION et de la table PATIENT. La jointure RIGHT JOIN est utilisée pour inclure toutes les lignes de la table PATIENT, même si elles n'ont pas de correspondance dans la table CONSULTATION. Les lignes de la table CONSULTATION qui ont une correspondance avec la table PATIENT sont également incluses dans le résultat, basées sur la condition de jointure où les valeurs de ID_PATIENT_ sont égales dans les deux tables.

— **ORDER BY C.DATE_CONSULTATION :** Cette clause est utilisée pour trier les résultats de la requête par ordre croissant de la colonne DATE_CONSULTATION de la table CONSULTATION.

En analyse, cette requête retournera toutes les données de la table PATIENT ainsi que les données correspondantes de la table CONSULTATION, si elles existent. Si un patient n'a pas de consultation associée, les colonnes de la table CONSULTATION pour ce patient seront NULL dans le résultat. Les résultats seront triés par date de consultation dans l'ordre croissant.

1.3.3 5 requêtes impliquant plus de 2 tables avec jointures internes dont 1 externe + 1 group by + 1 tri

1. ```
   SELECT C.*, P.*, E.*
   FROM CONSULTATION C
   INNER JOIN PATIENT P
   ON C.ID_PATIENT_ = P.ID_PATIENT_
   INNER JOIN EXAMEN E
   ON C.ID_CONSULTATION_  = E.ID_CONSULTATION_;
   ```

   Cette requête SQL Oracle est une commande SELECT qui effectue une jointure entre trois tables : "CONSULTATION" (aliasée par "C"), "PATIENT" (aliasée par "P"), et "EXAMEN" (aliasée par "E"). Voici une explication détaillée de chaque élément de la requête :

   - **SELECT C.*, P.*, E.* :** Cette clause spécifie les colonnes que vous souhaitez récupérer dans le résultat de la requête. Les préfixes "C.*", "P.*", et "E.*" signifient "toutes les colonnes" pour les tables "CONSULTATION", "PATIENT", et "EXAMEN", respectivement. Par conséquent, cette requête récupérera toutes les colonnes de ces trois tables.
   - **FROM CONSULTATION C :** "CONSULTATION" est le nom de la première table à partir de laquelle les données sont extraites. L'alias "C" est utilisé pour identifier la table "CONSULTATION" dans la requête et faciliter la référence à ses colonnes.
   - **INNER JOIN PATIENT P ON C.ID_PATIENT_ = P.ID_PATIENT_ :** C'est la première clause de jointure qui spécifie que nous voulons joindre la table "PATIENT" à la table "CONSULTATION". La condition de jointure "ON C.ID_PATIENT_ = P.ID_PATIENT_" indique que nous devons trouver des correspondances entre les valeurs de la colonne "ID_PATIENT_" de la table "CONSULTATION" et de la colonne "ID_PATIENT_" de la table "PATIENT".
   - **INNER JOIN EXAMEN E ON C.ID_CONSULTATION_ = E.ID_CONSULTATION_ :** C'est la deuxième clause de jointure qui spécifie que nous voulons joindre la table "EXAMEN" à la table "CONSULTATION". La condition de jointure "ON C.ID_CONSULTATION_ = E.ID_CONSULTATION_" indique que nous devons trouver des correspondances entre les valeurs de la colonne "ID_CONSULTATION_" de la table "CONSULTATION" et de la colonne "ID_CONSULTATION_" de la table "EXAMEN".

   En résumé, cette requête récupère toutes les colonnes des tables "CONSULTATION", "PATIENT", et "EXAMEN", pour lesquelles les valeurs des colonnes "ID_PATIENT_" et "ID_" correspondent dans les tables respectives. Cela permet de récupérer des informations sur les consultations, les patients associés à ces consultations, ainsi que les examens réalisés pour ces consultations.

2. ```
   SELECT  F.*, P.*, C.*
   FROM FACTURE F
   INNER JOIN PATIENT P
   ON F.ID_PATIENT_ = P.ID_PATIENT_
   INNER JOIN CONSULTATION C
   ON F.ID_FACTURE_ = C.ID_FACTURE_;
   ```

Cette requête SQL Oracle est une commande SELECT qui effectue une jointure entre trois tables : "FACTURE" (aliasée par "F"), "PATIENT" (aliasée par "P"), et "CONSULTATION" (aliasée par "C").

Voici une explication détaillée de chaque élément de la requête :

- **SELECT F.*, P.*, C.* :** Cette clause spécifie les colonnes que vous souhaitez récupérer dans le résultat de la requête. Les préfixes "F.*", "P.*", et "C.*" signifient "toutes les colonnes" pour les tables "FACTURE", "PATIENT", et "CONSULTATION", respectivement. Par conséquent, cette requête récupérera toutes les colonnes de ces trois tables.
- **FROM FACTURE F :** "FACTURE" est le nom de la première table à partir de laquelle les données sont extraites. L'alias "F" est utilisé pour identifier la table "FACTURE" dans la requête et faciliter la référence à ses colonnes.
- **INNER JOIN PATIENT P ON F.ID_PATIENT_ = P.ID_PATIENT_ :** C'est la première clause de jointure qui spécifie que nous voulons joindre la table "PATIENT" à la table "FACTURE". La condition de jointure "ON F.ID_PATIENT_ = P.ID_PATIENT_" indique que nous devons trouver des correspondances entre les valeurs de la colonne "ID_PATIENT_" de la table "FACTURE" et de la colonne "ID_PATIENT_" de la table "PATIENT".
- **INNER JOIN CONSULTATION C ON F.ID_FACTURE_ = C.ID_FACTURE_ :** C'est la deuxième clause de jointure qui spécifie que nous voulons joindre la table "CONSULTATION" à la table "FACTURE". La condition de jointure "ON F.ID_FACTURE_ = C.ID_FACTURE_" indique que nous devons trouver des correspondances entre les valeurs de la colonne "ID_FACTURE_" de la table "FACTURE" et de la colonne "ID_FACTURE_" de la table "CONSULTATION".

En résumé, cette requête récupère toutes les colonnes des tables "FACTURE", "PATIENT", et "CONSULTATION", pour lesquelles les valeurs des colonnes "ID_PATIENT_" et "ID_FACTURE_" correspondent dans les tables respectives. Cela permet de récupérer des informations sur les factures, les patients associés à ces factures, ainsi que les consultations liées à ces factures.

3.
```
SELECT C.*,P.*,E.*
FROM CONSULTATION C
INNER JOIN PATIENT P
ON C.ID_PATIENT_ = P.ID_PATIENT_
INNER JOIN EXAMEN E
ON C.ID_CONSULTATION_  = E.ID_CONSULTATION_
ORDER BY C.DATE_CONSULTATION;
```

Ce code est une requête SQL qui extrait des données de trois tables différentes : CONSULTATION (C), PATIENT (P), et EXAMEN (E). Voici une explication détaillée de la requête :

- **SELECT C.*, P.*, E.* :**** Cela signifie que nous voulons sélectionner toutes les colonnes de chaque table impliquée dans la requête. Cela est indiqué par C.*, P.*, et E.* où C, P, et E sont des alias pour les tables CONSULTATION, PATIENT, et EXAMEN respectivement.
- **FROM CONSULTATION C INNER JOIN PATIENT P ON C.ID_PATIENT_ = P.ID_PATIENT_ :**** Cette partie de la requête spécifie à partir de quelles tables

les données doivent être extraites et comment ces tables sont liées. Les mots-clés "INNER JOIN" indiquent que seules les lignes ayant des correspondances dans les deux tables seront incluses dans le résultat. Dans ce cas, nous joignons la table CONSULTATION (aliasée par C) avec la table PATIENT (aliasée par P) en utilisant la condition de jointure "ON C.ID_PATIENT_ = P.ID_PATIENT_". Cela signifie que nous joignons les lignes où l'ID du patient dans la table CONSULTATION correspond à l'ID du patient dans la table PATIENT.

— **INNER JOIN EXAMEN E ON C.ID_CONSULTATION_ = E.ID_CONSULTATION_ :**** Cette partie ajoute une autre jointure. Nous joignons la table EXAMEN (aliasée par E) à la précédente jointure entre CONSULTATION et PATIENT. Nous utilisons également la condition de jointure "ON C.ID_CONSULTATION_ = E.ID_CONSULTATION_", ce qui signifie que nous joignons les lignes où l'ID de consultation dans la table CONSULTATION correspond à l'ID de consultation dans la table EXAMEN.

— **ORDER BY C.DATE_CONSULTATION ;**** Cette partie de la requête spécifie l'ordre dans lequel les résultats doivent être triés. Ici, nous ordonnons les résultats par la colonne DATE_CONSULTATION de la table CONSULTATION (aliasée par C), ce qui signifie que les résultats seront affichés par date croissante de consultation.

En résumé, cette requête récupère toutes les colonnes des tables CONSULTATION, PATIENT et EXAMEN, en combinant les données des trois tables en fonction des conditions de jointure spécifiées, et les trie par date de consultation.

4.
```
SELECT F.ID_PATIENT_ ,P.EMAIL,SUM(F.MONTANT_TOTAL) MONTANT_TOTAL,
F.DATE_FACTURE, C.DATE_CONSULTATION
FROM FACTURE F
INNER JOIN PATIENT P
ON F.ID_PATIENT_ = P.ID_PATIENT_
INNER JOIN CONSULTATION C
ON F.ID_FACTURE_ = C.ID_FACTURE_
GROUP BY F.ID_PATIENT_,P.EMAIL,F.DATE_FACTURE, C.DATE_CONSULTATION
ORDER BY F.ID_PATIENT_, P.EMAIL,F.DATE_FACTURE, C.DATE_CONSULTATION;
```

Cette requête SQL est conçue pour extraire des informations spécifiques sur les factures associées aux consultations et aux patients. Voici une explication détaillée de chaque partie de la requête :

— **SELECT F.ID_PATIENT_, P.EMAIL, SUM(F.MONTANT_TOTAL) AS MONTANT_TOTAL, F.DATE_FACTURE, C.DATE_CONSULTATION :**** Cette partie spécifie les colonnes que nous voulons sélectionner dans les résultats de la requête. F.ID_PATIENT_ : Identifiant du patient associé à la facture. P.EMAIL : Adresse e-mail du patient associé à la facture. SUM(F.MONTANT_TOTAL) AS MONTANT_TOTAL : C'est une agrégation. Ici, nous calculons la somme totale des montants des factures (MONTANT_TOTAL) associées à chaque patient. Nous utilisons la fonction SUM() pour agréger les montants des factures. F.DATE_FACTURE : Date à laquelle la facture a été émise. C.DATE_CONSULTATION : Date de la consultation associée à la facture.

— **FROM FACTURE F INNER JOIN PATIENT P ON F.ID_PATIENT_ = P.ID_PATIENT_ INNER JOIN CONSULTATION C ON F.ID_FACTURE_ = C.ID_FACTURE_ :** Cette partie spécifie les tables à partir desquelles les données sont extraites et comment elles sont liées. Nous joignons les tables FACTURE (aliasée par F) et PATIENT (aliasée par P) en utilisant l'ID du patient pour trouver les correspondances entre les deux tables. Ensuite, nous joignons la table FACTURE (aliasée par F) et la table CONSULTATION (aliasée par C) en utilisant l'ID de facture pour trouver les correspondances entre ces deux tables.

— **GROUP BY F.ID_PATIENT_, P.EMAIL, F.DATE_FACTURE, C.DATE_CONSULTATION :** Cette partie indique comment les résultats doivent être regroupés. Nous regroupons les résultats en fonction de l'ID du patient, de l'adresse e-mail du patient, de la date de la facture et de la date de la consultation. Cela signifie que toutes les lignes ayant les mêmes valeurs pour ces colonnes seront regroupées ensemble.

— **ORDER BY F.ID_PATIENT_, P.EMAIL, F.DATE_FACTURE, C.DATE_CONSULTATION :** Cette partie spécifie l'ordre dans lequel les résultats doivent être triés. Les résultats seront triés par ordre croissant d'ID du patient, d'adresse e-mail du patient, de date de facture, puis de date de consultation.

En résumé, cette requête récupère des informations sur les factures, les patients et les consultations, agrège les montants des factures par patient, et les trie par ordre croissant d'ID du patient, d'adresse e-mail du patient, de date de facture, puis de date de consultation.

5.
```
SELECT C.*,E.*,P.*
FROM CONSULTATION C
INNER JOIN EXAMEN E
ON C.ID_CONSULTATION_  = E.ID_CONSULTATION_
RIGHT JOIN PATIENT P
ON C.ID_PATIENT_ = P.ID_PATIENT_
ORDER BY C.DATE_CONSULTATION;
```

Cette requête SQL récupère des informations à partir des tables CONSULTATION (aliasée par C), EXAMEN (aliasée par E), et PATIENT (aliasée par P). Voici une explication détaillée de chaque partie de la requête :

— **SELECT C.*, E.*, P.* :** Cette partie spécifie les colonnes que nous voulons sélectionner dans les résultats de la requête. C.* : Sélectionne toutes les colonnes de la table CONSULTATION. E.* : Sélectionne toutes les colonnes de la table EXAMEN. P.* : Sélectionne toutes les colonnes de la table PATIENT.

FROM CONSULTATION C INNER JOIN EXAMEN E ON C.ID_CONSULTATION_ = E.ID_CONSULTATION_ : Cette partie spécifie les tables à partir desquelles les données sont extraites et comment elles sont liées. Nous joignons la table CONSULTATION (aliasée par C) avec la table EXAMEN (aliasée par E) en utilisant l'ID de consultation pour trouver les correspondances entre les deux tables.

RIGHT JOIN PATIENT P ON C.ID_PATIENT_ = P.ID_PATIENT_ : Cette partie effectue une jointure de type RIGHT JOIN entre la table PATIENT (aliasée par P) et le résultat de la jointure précédente (CONSULTATION et EXAMEN). Cela signifie

que toutes les lignes de la table PATIENT seront incluses dans le résultat final, même si elles n'ont pas de correspondance dans les tables CONSULTATION et EXAMEN. Les correspondances seront faites sur la base de l'ID du patient.

ORDER BY C.DATE_CONSULTATION : Cette partie spécifie l'ordre dans lequel les résultats doivent être triés. Les résultats seront triés par ordre croissant de la colonne DATE_CONSULTATION de la table CONSULTATION.

En résumé, cette requête récupère toutes les colonnes des tables CONSULTATION, EXAMEN et PATIENT, en combinant les données des tables CONSULTATION et EXAMEN en fonction de l'ID de consultation, puis joint la table PATIENT en fonction de l'ID du patient. Les résultats sont triés par date de consultation.

1.4 Dictionnaire de données MERISE

Table 1: Dictionnaire de données MERISE

Titre	Caractéristiques	
CONSULTATION	Description :	Enregistre les détails d'une consultation médicale entre un médecin et un patient.
	Format de données :	— ID_CONSULTATION_ : Entier — ID_FACTURE_ : Entier — ID_MEDECIN_ : Entier — ID_PATIENT_ : Entier — DATE_CONSULTATION : Date
	Type :	— ID_CONSULTATION_ : NUMBER(38) — ID_FACTURE_ : NUMBER(38) — ID_MEDECIN_ : NUMBER(38) — ID_PATIENT_ : NUMBER(38) — DATE_CONSULTATION : DATE
	Identifiant :	ID_CONSULTATION_
	Contraintes :	— ID_CONSULTATION_ IS NOT NULL — ID_FACTURE_ IS NOT NULL — ID_MEDECIN_ IS NOT NULL — ID_PATIENT_ IS NOT NULL — DATE_CONSULTATION IS NOT NULL — Clé primaire (PK_CONSULTATION) sur ID_CONSULTATION_ — Contrainte de clé étrangère FK_CONSULTA_EFFECTUER_MEDECIN) sur ID_MEDECIN_ référençant MEDECIN(ID_MEDECIN_) — Contrainte de clé étrangère (FK_CONSULTA_INCLURE_FACTURE) sur ID_FACTURE_ référençant FACTURE(ID_FACTURE_) — Contrainte de clé étrangère (FK_CONSULTA_PASSER_PATIENT) sur ID_PATIENT_ référençant PATIENT(ID_PATIENT_)

Continued on next page

Table 1: Dictionnaire de données MERISE (Continued)

Titre	Caractéristiques	
EXAMEN	Description :	Enregistre les détails des examens médicaux effectués lors d'une consultation.
	Format de données :	— ID_EXAMEN_ : Entier — ID_CONSULTATION_ : Entier — DETAILS_EXAMEN : Chaîne de caractères — DATE_EXAMEN : Date
	Type :	— ID_EXAMEN_ : NUMBER(38) — ID_CONSULTATION_ : NUMBER(38) — DETAILS_EXAMEN : VARCHAR2(100) — DATE_EXAMEN : DATE
	Identifiant :	ID_EXAMEN_
	Contraintes :	— ID_EXAMEN_ IS NOT NULL — ID_CONSULTATION_ IS NOT NULL — DETAILS_EXAMEN IS NOT NULL — DATE_EXAMEN IS NOT NULL — Clé primaire (PK_EXAMEN) sur ID_EXAMEN_ — Contrainte de clé étrangère (FK_EXAMEN_CONTENIR_CONSULTA) sur ID_CONSULTATION référençant CONSULTATION(ID_CONSULTATION_)

Continued on next page

Table 1: Dictionnaire de données MERISE (Continued)

Titre	Caractéristiques	
FACTURE	Description :	Enregistre les détails des factures générées suite à une consultation.
	Format de données :	— ID_FACTURE_ : Entier — ID_PATIENT_ : Entier — MONTANT_TOTAL : Nombre décimal — DATE_FACTURE : Date
	Type :	— ID_FACTURE_ : NUMBER(38) — ID_PATIENT_ : NUMBER(38) — MONTANT_TOTAL : NUMBER(15,2) — DATE_FACTURE : DATE
	Identifiant :	ID_FACTURE_
	Contraintes :	— ID_FACTURE_ IS NOT NULL — ID_PATIENT_ IS NOT NULL — MONTANT_TOTAL IS NOT NULL — DATE_FACTURE IS NOT NULL — Clé primaire (PK_FACTURE) sur ID_FACTURE_ — Contrainte de clé étrangère (FK_FACTURE_RECEVOIR_PATIENT) sur ID_PATIENT_ référençant PATIENT(ID_PATIENT_)

Continued on next page

Table 1: Dictionnaire de données MERISE (Continued)

Titre	Caractéristiques	
MEDECIN	Description :	Enregistre les détails des médecins.
	Format de données :	— ID_MEDECIN_ : Entier — NOM : Chaîne de caractères — PRENOM : Chaîne de caractères — SPECIALITE : Chaîne de caractères — TELEPHONE : Chaîne de caractères — EMAIL : Chaîne de caractères
	Type :	— ID_MEDECIN_ : NUMBER(38) — NOM : VARCHAR2(50) — PRENOM : VARCHAR2(50) — SPECIALITE : VARCHAR2(50) — TELEPHONE : VARCHAR2(8) — EMAIL : VARCHAR2(50)
	Identifiant :	ID_MEDECIN_
	Contraintes :	— ID_MEDECIN_ IS NOT NULL — NOM IS NOT NULL — PRENOM IS NOT NULL — SPECIALITE IS NOT NULL — TELEPHONE IS NOT NULL — EMAIL IS NOT NULL — Clé primaire (PK_MEDECIN) sur ID_MEDECIN_

Continued on next page

Table 1: Dictionnaire de données MERISE (Continued)

Titre	Caractéristiques	
PATIENT	Description :	Enregistre les détails des patients.
	Format de données :	— ID_PATIENT_ : Entier — NOM : Chaîne de caractères — PRENOM : Chaîne de caractères — ADRESSE : Chaîne de caractères — EMAIL : Chaîne de caractères — DATE_NAISSAINCE : Date
	Type :	— ID_PATIENT_ : NUMBER(38) — NOM : VARCHAR2(50) — PRENOM : VARCHAR2(50) — ADRESSE : VARCHAR2(100) — EMAIL : VARCHAR2(50) — DATE_NAISSAINCE : DATE
	Identifiant :	ID_PATIENT_
	Contraintes :	— ID_PATIENT_ IS NOT NULL — NOM IS NOT NULL — PRENOM IS NOT NULL — DATE_NAISSAINCE IS NOT NULL — Clé primaire (PK_PATIENT) sur ID_PATIENT_

Continued on next page

Table 1: Dictionnaire de données MERISE (Continued)

Titre	Caractéristiques	
PRESCRIPTION	Description :	Enregistre les détails des prescriptions médicales effectuées lors d'une consultation.
	Format de données :	— ID_PRESCRIPTION_ : Entier — ID_CONSULTATION_ : Entier — DETAILS_PRESCRIPTION : Chaîne de caractères — DATE_PRESCRIPTION : Date
	Type :	— ID_PRESCRIPTION_ : NUMBER(38) — ID_CONSULTATION_ : NUMBER(38) — DETAILS_PRESCRIPTION : VARCHAR2(100) — DATE_PRESCRIPTION : DATE
	Identifiant :	ID_PRESCRIPTION_
	Contraintes :	— ID_PRESCRIPTION_ IS NOT NULL — ID_CONSULTATION_ IS NOT NULL — DETAILS_PRESCRIPTION IS NOT NULL — DATE_PRESCRIPTION IS NOT NULL — Clé primaire (PK_PRESCRIPTION) sur ID_PRESCRIPTION_ — Contrainte de clé étrangère (FK_PRESCRIP_CONTENIR_CONSULTA) sur ID_CONSULTATION référençant CONSULTATION(ID_CONSULTATION_)

Continued on next page

Table 1: Dictionnaire de données MERISE (Continued)

Titre	Caractéristiques	
RENDEZ_VOUS	Description :	Enregistre les rendez-vous pris entre un patient et un médecin.
	Format de données :	— ID_PATIENT_ : Entier — ID_MEDECIN_ : Entier — DATE_RENDEZ_VOUS : Date
	Type :	— ID_PATIENT_ : NUMBER(38) — ID_MEDECIN_ : NUMBER(38) — DATE_RENDEZ_VOUS : DATE
	Identifiant :	(ID_PATIENT_, ID_MEDECIN_)
	Contraintes :	— ID_PATIENT_ IS NOT NULL — ID_MEDECIN_ IS NOT NULL — DATE_RENDEZ_VOUS IS NOT NULL — Clé primaire composée (PK_RENDEZ_VOUS) sur (ID_PATIENT_, ID_MEDECIN_) — Contrainte de clé étrangère (FK_RENDEZ_V_RENDEZ_VO_MEDECIN) sur (ID_MEDECIN_) référençant MEDECIN (ID_MEDECIN_) — Contrainte de clé étrangère (FK_RENDEZ_V_RENDEZ_VO_PATIENT) sur (ID_PATIENT_) référençant PATIENT (ID_PATIENT_)

1.5 Description textuelle des associations

Sont décrites ci-dessous les associations entres les différentes entités du modèle.

1.5.1 Association "Effectuer" entre MEDECIN et CONSULTATION

Cette association indique que chaque consultation est effectuée par un médecin. Un médecin peut effectuer plusieurs consultations, mais une consultation est effectuée par un seul médecin à la fois.

1.5.2 Association "Inclure" entre FACTURE et CONSULTATION

Cette association représente le fait qu'une facture inclut une consultation. Chaque consultation peut être incluse dans une seule facture, mais une facture peut inclure plusieurs consultations.

1.5.3 Association "Passer" entre PATIENT et CONSULTATION

Cette association signifie qu'un patient passe une consultation. Chaque consultation est passée par un seul patient, mais un patient peut passer plusieurs consultations.

1.5.4 Association "Contenir" entre CONSULTATION et EXAMEN

Cette association indique que chaque consultation peut contenir plusieurs examens. Chaque examen est associé à une seule consultation.

1.5.5 Association "Recevoir" entre PATIENT et FACTURE

Cette association représente le fait qu'un patient peut recevoir une facture. Chaque facture est destinée à un seul patient, mais un patient peut recevoir plusieurs factures.

1.5.6 Association "Contenir" entre CONSULTATION et PRESCRIPTION

Cette association signifie qu'une consultation peut contenir plusieurs prescriptions. Chaque prescription est associée à une seule consultation.

1.5.7 Association "Rendez-vous" entre PATIENT et MEDECIN

Cette association indique que chaque rendez-vous est entre un patient et un médecin. Chaque rendez-vous est pris par un patient avec un médecin spécifique.

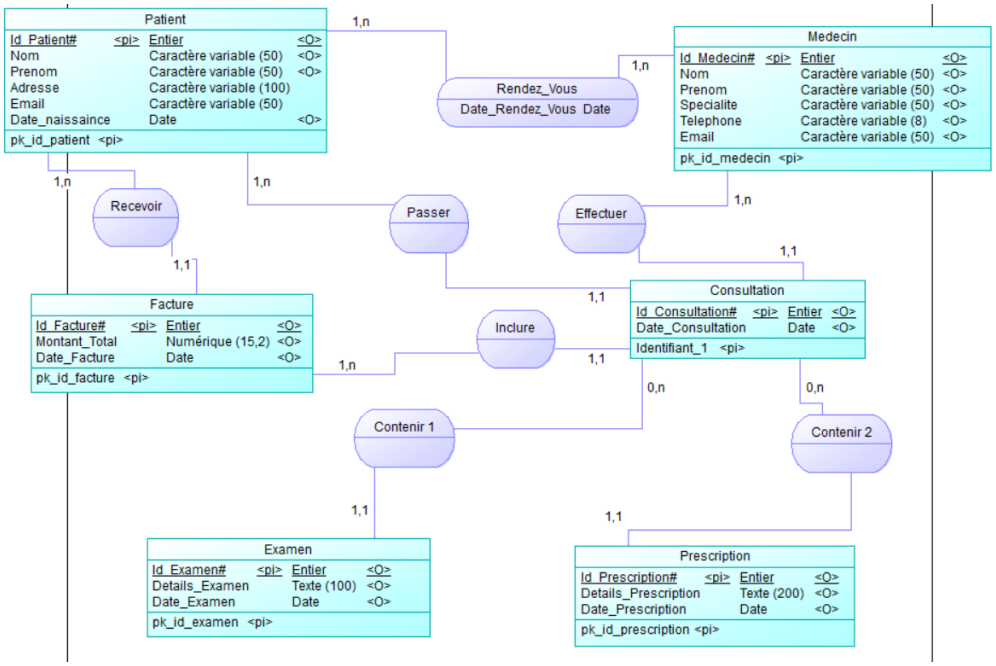
1.6 Définition du Modèle Entité-Association MERISE

La figure 1 présente le modèle Modèle Entité-Association MERISE.

1.7 La définition du modèle logique de Données ou schéma rélationnel

La figure 2 présente le modèle Modèle Entité-Association MERISE.

FIGURE 1 – Définition du Modèle Entité-Association MERISE



1.8 Spécification des traitement avec des packages PLSQL

1.8.1 Table MEDECIN

1.8.2 Table B

1.9 Spécification des triggers

Voici les définitions textuelles des deux triggers :

1.9.1 Trigger "TG_CTRL_RENDEZ_VOUS_DOUBLE"

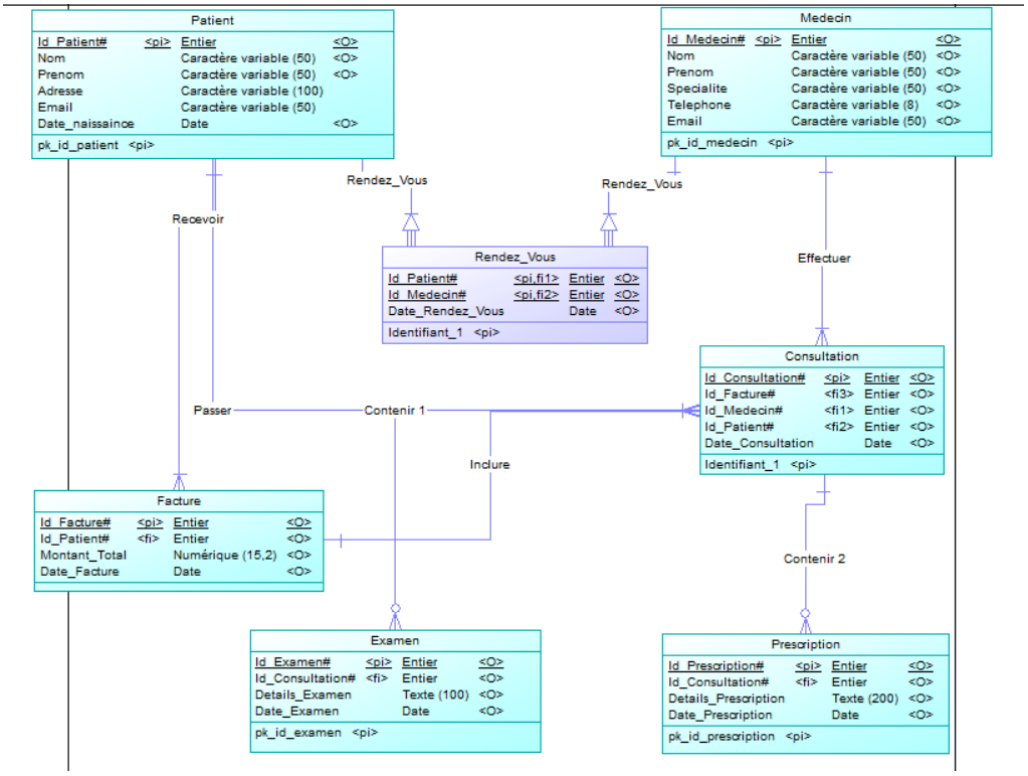
— **Objectif**
Empêcher la création de rendez-vous en doublon pour un même patient avec un même médecin à la même date.

— **Déclenchement**
Avant l'insertion ou la mise à jour dans la table RENDEZ_VOUS.

— **Action**

1. Le trigger vérifie s'il existe déjà un rendez-vous pour le nouveau patient avec le nouveau médecin à la nouvelle date spécifiée.
2. Si un rendez-vous en doublon est trouvé, une erreur est levée pour signaler que le rendez-vous existe déjà pour ce patient avec ce médecin à cette date.

FIGURE 2 – La définition du modèle logique de Données ou schéma rélationnel



1.9.2 Trigger "TG_HISTORIQUE_FACTURE"

— **Objectif**
Maintenir un historique des modifications apportées aux factures.

— **Déclenchement**
Après l'insertion, la mise à jour ou la suppression dans la table FACTURE.

— **Action**

1. Si une nouvelle facture est insérée (INSERTING), les détails de la facture sont insérés dans la table HISTORIQUE_FACTURE avec l'action 'INSERTION'.
2. Si une facture est mise à jour et que la colonne MONTANT_TOTAL est modifiée (UPDATING ('MONTANT_TOTAL')), les anciennes et nouvelles valeurs du montant ainsi que la date de la modification sont insérées dans la table HISTORIQUE_FACTURE avec l'action 'MODIFICATION'.
3. Si une facture est supprimée (DELETING), les détails de la facture avant sa suppression sont insérés dans la table HISTORIQUE_FACTURE avec l'action 'SUPPRESSION'. De plus, toute référence à cette facture dans la table CONSULTATION est supprimée en mettant à jour la colonne ID_FACTURE_ à NULL.