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

17 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

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;
```

➔ Analyse de la requête

1.1.3 2 requêtes impliquant plus de 2 tables

1.2 Description textuelles des requêtes de suppression

(2 requêtes impliquant 1 table, 2 requêtes impliquant 2 tables, 2 requêtes impliquant plus de 2 tables)

1.3 Description textuelles des requêtes de consultation

(5 requêtes impliquant 1 table dont 1 avec un group By et une avec un Order By, 5 requêtes impliquant 2 tables avec jointures internes dont 1 externe + 1 group by + 1 tri, 5 requêtes impliquant plus de 2 tables avec jointures internes dont 1 externe + 1 group by + 1 tri)

1.4 Dictionnaire de données MERISE. Pour chaque entité décrire chacune des propriétés

Titre / description / format des données / type / Indentifiant / contraintes

1.5 Description textuelles des associations

Décrire textuellement les associations entre entités

1.6 Définition du Modèle Entité-Association MERISE

(en utilisant le logiciel Poweramc de SYBASE/SAP ou manuellement). Vous devez vous limiter à 10 entités maximum et 5 minimum. Vous devez ici prendre en compte les contraintes identifiées lors de la description du dictionnaire de données. Exemple de liens d'association pour deux entités A et B ayant une liaisons 1 : N ou N-M (exemple UN PILOTE ASSURE 0, 1 ou plusieurs VOL, un VOL est assuré par 1 et 1 PILOTE au plus)

1.7 Définition du modèle logique de Données ou schéma rélationnel

(en utilisant le logiciel Poweramc de SYBASE/SAP ou manuellement) un schéma de données logique en respectant les contraintes d'intégrités d'entité (PRIMARY KEY), de domaine (CHECK, NOT NULL, ...) et de référence(REFERENCES / foreign key). Générable automatiquement avec POWERAMC si vous avez décrites au niveau MCD

1.8 Spécification des traitement avec des packages PLSQL (Modèle de traitements)

Choisir parmi vos tables deux d'entres (A et B) elles sur lesquelles les fonctions suivantes vont être spécifiées puis implémentées : Sur la table A, définir un package plsql ayant le nom de la dite table : - ajouter une nouvelle occurence à A : fonction AInserer ; - supprimer une occurrence à A (Attention : les enregistrements liés dans B doivent aussi être supprimés) : fonction ASupprimer ; - modifier des informations sur de A : fonction AmodifierF1, AmodifierF2 (texte requêtes correspondantes plus haut) ; - lister toutes les occurrences de A : fonction ALister ; - fournir le nombre total des occurrences de A : fonction ATotal ; - Proposer aussi 3 fonctions avec des requêtes de consultation impliquant 2 ou 3 tables au moins (jointure, groupe, tri) : fonction Af1, Af2, Af3. f1, f2, f3 sont des noms à définir AmodifierF2 (texte requêtes correspondantes plus haut) ; Sur la table B, définir un package plsql ayant le nom de la dite table : - ajouter une nouvelle occurence à B : fonction BInserer - supprimer une occurrence à B : fonction BSupprimer ;- modifier des informations sur de B : fonction BModifierF1, BModifierF2 (texte requêtes correspondantes plus haut) ; - lister toutes les occurrences de B pour une occurrence de A donnée : fonction Blister Nota : Seule la partie spécification de chaque package est nécessaire ici. bien choisir les paramètres des méthodes. Bien nommer les méthodes. Remplacer F1 à Fn par des noms appropriés.

1.9 Spécification des trigers

Vous devez définir textuellement aux moins deux triggers. Ces triggers vous permettrant devront vous permettre de gérer aux moins deux règles de gestions qui ne peuvent être prises en compte à travers le schéma de données.