

SAE S4.03 - Implémentation des Modèles de Bases de Données

1. Modèle Relationnel (SQL)

1.1. Création de la base de données relationnelle

```
CREATE TABLE Piece (  
    ID_PIECE INT PRIMARY KEY,  
    nom VARCHAR(255),  
    description VARCHAR(255),  
    prix DECIMAL(10,2)  
);  
  
CREATE TABLE Composition (  
    ID_PIECE_COMPOSEE INT,  
    ID_PIECE_COMPOSANT INT,  
    quantite INT,  
    PRIMARY KEY (ID_PIECE_COMPOSEE, ID_PIECE_COMPOSANT),  
    FOREIGN KEY (ID_PIECE_COMPOSEE) REFERENCES Piece(ID_PIECE),  
    FOREIGN KEY (ID_PIECE_COMPOSANT) REFERENCES Piece(ID_PIECE)  
);
```

1.2. Requêtes SQL

1.2.2. Liste des composants directs d'une pièce composée

Dans l'exemple avec : 'Panneau de fuselage'

```
SELECT P.nom, C.quantite  
FROM Composition C JOIN Piece P ON C.ID_PIECE_COMPOSANT = P.ID_PIECE  
WHERE C.ID_PIECE_COMPOSEE = (SELECT ID_PIECE FROM Piece WHERE nom =  
    'Panneau de fuselage');
```

1.2.3. Liste des pièces composées triées par nombre de composants

```
SELECT P.nom, COUNT(C.ID_PIECE_COMPOSANT) AS nombre_composants  
FROM Piece P JOIN Composition C ON P.ID_PIECE = C.ID_PIECE_COMPOSEE  
GROUP BY P.nom  
ORDER BY nombre_composants DESC;
```

1.2.4. Nombre de pièces composées

```
SELECT COUNT(DISTINCT ID_PIECE_COMPOSEE) AS nombre_pieces_composees
FROM Composition;
```

1.2.5. Coût total de chaque pièces composées

```
SELECT C.ID_PIECE_COMPOSEE, P.nom, SUM(C.quantite * Pc.prix) AS
cout_total
FROM Composition C JOIN Piece Pc ON C.ID_PIECE_COMPOSANT = Pc.ID_PIECE
JOIN Piece P ON C.ID_PIECE_COMPOSEE = P.ID_PIECE
GROUP BY C.ID_PIECE_COMPOSEE, P.nom;
```

1.2.6. Pièces nécessaires directement ou indirectement

```
SELECT P.nom AS piece_composee, Pc.nom AS composant
FROM Piece P JOIN Composition C ON P.ID_PIECE = C.ID_PIECE_COMPOSEE JOIN
Piece Pc ON C.ID_PIECE_COMPOSANT = Pc.ID_PIECE
CONNECT BY PRIOR C.ID_PIECE_COMPOSANT = C.ID_PIECE_COMPOSEE
ORDER BY P.ID_PIECE;
```

2. Modèle Objet-Relationnel (Oracle)

2.1. Définition des types complexes

```
-- Création de l'objet IndicesQualite
CREATE OR REPLACE TYPE IndicesQualite AS OBJECT (
    nom VARCHAR2(50),
    valeur INT,
    poids INT
);
/

CREATE TYPE ListeIndices AS TABLE OF IndicesQualite;
/

-- Création de l'objet Equipe
CREATE OR REPLACE TYPE Equipe AS OBJECT (
    nom VARCHAR2(100),
    fonction VARCHAR2(100)
);
/

CREATE TYPE EquipeTabT as table of Equipe;
/
```

```
CREATE TABLE PieceObj (  
    num_piece INT PRIMARY KEY,  
    nom_piece VARCHAR(255),  
    date_debut DATE,  
    date_fin DATE,  
    equipe EquipeTabT,  
    indices_qualite ListeIndices  
) NESTED TABLE equipe STORE AS Equipe_Table  
NESTED TABLE indices_qualite STORE AS Indices_Table;
```

2.2. Requêtes

2.2.1. Pour chaque pièce, le nombre de personnes de l'équipe, par fonction

```
SELECT p.nom_piece, e.fonction, COUNT(e.nom) AS nb_personnes  
FROM PieceObj p, TABLE(p.equipe) e  
GROUP BY p.nom_piece, e.fonction;
```

2.2.2. Nombre de pièces par mécanicien

```
SELECT e.nom, COUNT(p.num_piece) AS nb_pieces  
FROM PieceObj p, TABLE(p.equipe) e  
WHERE e.fonction = 'Mecanicien'  
GROUP BY e.nom;
```

2.2.3. Pour chaque pièce, l'impact de chaque indice de qualité

```
SELECT p.nom_piece, i.nom, i.valeur * i.poids AS impact  
FROM PieceObj p, TABLE(p.indices_qualite) i;
```

2.2.4. Pour chaque indice de qualité, son impact moyen

```
SELECT i.nom, AVG(i.valeur * i.poids) AS impact_moyen  
FROM PieceObj p, TABLE(p.indices_qualite) i  
GROUP BY i.nom;
```

3. Modèle Logique (Datalog)

Utilisation de l'archive *AbcDatalog-0.7.2.jar* disponible sur Célène

Commande pour lancer le programme : `java -jar AbcDatalog-0.7.2.jar`

3.1. Prédicats extensionnels

...

3.2. Liste des pièces nécessaires pour une pièce composée

...

3.3. Analysez votre requête pour répondre à la question 1(6)**.

Faites le parallèle de la requête relationnelle (question 1) avec celle que vous venez d'écrire en Datalog (item 2). Pouvez-vous associer certaines parties de la requête relationnelle aux règles de votre programme Datalog ? Comment ?

Réponse :

Réponse bientôt ici

3.4. Liste des pièces composées ne contenant aucun composant à 300 euros

...

4. Modèle Graphe (Neo4J)

PROF

4.1. Modélisation

Noeuds :

- Piece: Représente une pièce avec les propriétés :
 - id (identifiant unique)
 - nom
 - description
 - prix
 - type (pour distinguer pièces de base et composées)

Relations :

- COMPOSE_DE: Relation entre une pièce composée et ses composants
 - Propriété : quantité (quantité nécessaire)

Justification des choix :

- Le modèle graphe est particulièrement adapté pour représenter les relations de composition hiérarchique
- La relation COMPOSE_DE permet de modéliser directement les liens entre pièces
- La propriété quantite sur la relation capture l'information de quantité nécessaire
- Le modèle permet facilement de naviguer dans les relations parent-enfant

4.2. Implémentation de la base de données en Neo4J

...

4.3. Requête Cypher correspondant à la requête 6 que vous avez écrite en SQL (section 3.1)