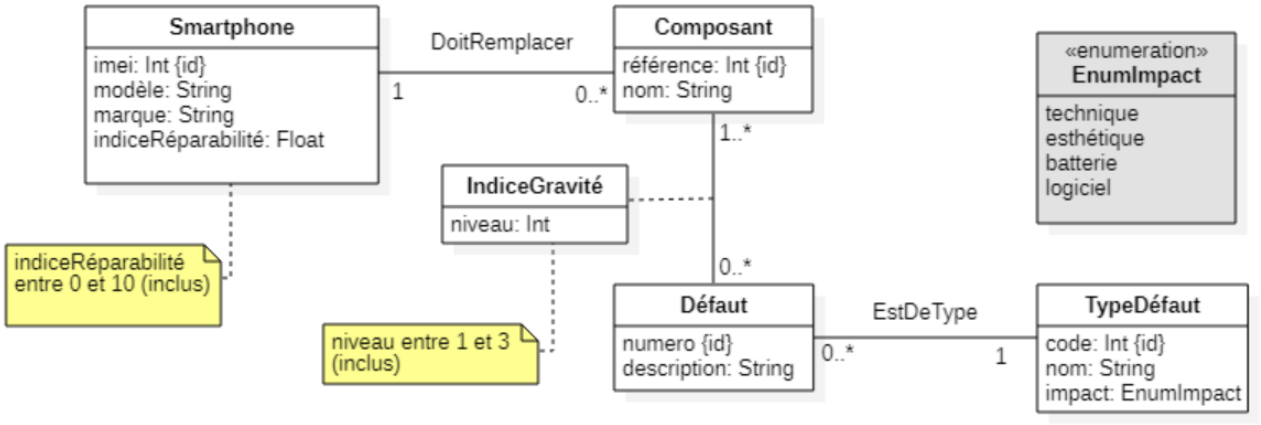
Examen INF403 - Mai 2023

Durée : 2 Heures / Document autorisé : une feuille A4, recto-verso manuscrite
Deux parties à faire dans des feuilles à part

A propos de Smartphones reconditionnés

Le diagramme UML ci-dessous modélise une première version d'un système d'information pour gérer le reconditionnement (réparations) de smartphones de manière simplifiée. *L'état actuel de la BD est fourni en annexe.*



Un début de traduction en relationnel est le suivant :

Smartphones (imei_smartphone, modele_smartphone, marque_smartphone, indice_reparabilite_smartphone)
/ (i, mo, ma, i) ∈ Smartphones ⟺ le smartphone identifié par un numéro imei (international mobile equipment identity), a un modèle mo, une marque ma et il a un indice de réparabilité entre 0 et 10 (le meilleur étant 10). */*
Composants (reference_composant, nom_composant, imei_smartphone)
/ (r, n, i) ∈ Composants ⟺ le composant identifié par un numéro de référence r a un nom n et appartient à un smartphone avec imei i */*
TypesDefauts (code_type_defaut, nom_type_defaut, impact_type_defaut)
/ (c, n, i) ∈ TypesDefauts ⟺ le type défaut t est identifié par un code. Il possède un nom n et il est caractérisé par un impact de type "technique", "esthétique", "batterie" ou "logiciel" */*
Defauts (numero_defaut, description_defaut, code_type_defaut)
/ (n, d, c) ∈ Defauts ⟺ le défaut identifié par un numéro n est caractérisé par une description d et lié à un type de code c) */*
IndicesGravites (reference_composant, numero_defaut, niveau_indice_gravite)
/ (r, nu, ni) ∈ IndicesGravites ⟺ le composant avec référence r peut être affecté par un défaut avec numéro nu avec un indice de gravité 1, 2 ou 3 (3 étant le plus grave) */*

1 Partie 1 (dans une feuille à part)

1.1 Compréhension de modèles

Question 1 (1,5 points) :
Compléter le modèle relationnel avec les **contraintes d'intégrité référentielle** (ex. R[x] ⊆ T[y]) et indiquer si certains attributs ne peuvent pas être vides (**not null**)
Les contraintes d'intégrité référentielle :

Composants[imei_smartphone] ⊆ Smartphones[imei_smartphone]
Defauts[code_type_defaut] ⊆ TypesDefauts[code_type_defaut]
IndicesGravites[numero_defaut] = Defauts[numero_defaut]
IndicesGravites[reference_composant] ⊆ Composants[reference_composant]

Les attributs clés ne peuvent pas être vides. imei_smartphone ne peut pas être vide dans la relation Composants (cardinalité 1

Question 2 (1,5 points) :
Expliquer les problèmes potentiels des tables ci-dessous par rapport au modèle UML précédant et à la problématique traitée.

1. **Composants** (reference_composant, nom_composant, imei_smartphone)

   Inapproprié, car ici, un smartphone (identifié par son imei) peut juste être lié à un composant. Par exemple, le smartphone 001, peut apparaitre une seule fois dans la table car imei_smartphone est la clé.

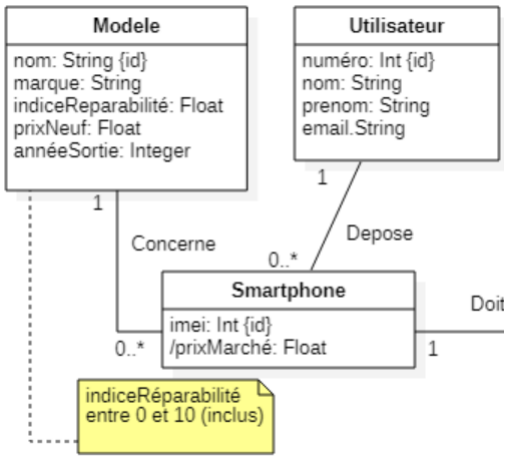
2. **Defauts** (numero_defaut, description_defaut, code_type_defaut, nom_type_defaut, impact_type_defaut)

   Inapproprié, car on peut avoir de la redondance concernant le nom_type_defaut, qui sera répété pour chaque numero_defaut. D'où l'intérêt de faire une relation TypeDefaut, qui fera le lien entre le code et le nom et l'impact qu'une seule fois.

1.2 Modélisation UML

Question 3 (2 points) :
Proposer une extension du modèle UML qui étend le modèle de l'énoncé. **Modéliser seulement les parties concernées par l'extension**. Indiquer les classes, attributs, cardinalités, noms des associations et possibles contraintes avec soin.

- Supposons que la marque ainsi que l'indice de réparabilité dépendent du modèle du smartphone. Un même modèle peut concerner plusieurs smartphones. Pour un modèle, on peut aussi indiquer son prix neuf et son année de sortie.
- Pour un smartphone donné, on veut indiquer le *prix de marché* qui sera calculé en fonction de l'année de sortie du modèle, son prix neuf ainsi que le nombre de défauts et leur gravité (les détails du calcul ne nous intéressent pas ici).
- Un utilisateur de la plateforme est identifié par un numéro. Il a un nom un prénom et un email. Un utilisateur peut déposer plusieurs smartphones mais un smartphone ne peut être déposé que par un seul utilisateur.



Question 4 (2 points) :
Proposer une traduction en relationnel à partir du modèle UML proposé dans la question précédente. Donner toutes les contraintes possibles sauf celles des domaines.

Modeles (nom_modele, marque_modele, indice_reparabilite_modele, prix_neuf_modele, annee_sortie_modele)
Utilisateurs (numero_utilisateur, nom_utilisateur, prenom_utilisateur, email_utilisateur)
Smartphones_base (imei_smartphone, nom_modele, numero_utilisateur)*/* nom_modele et numero_utilisateur not null */*
(View) **Smartphones** (imei_smartphone, nom_modele, numero_utilisateur, prix_marche_smartphone)

Les contraintes d'intégrité référentielle :

Smartphones_base[nom_modele] $\subseteq$ Modeles[nom_modele]
Smartphones_base[numero_utilisateur] $\subseteq$ Utilisateurs[numero_utilisateur]

2 Partie 2 (dans une feuille à part)

2.1 Compréhension de requêtes en SQL

Question 5 (3 points) :
En considérant les relations fournies en annexe, donner **le résultat retourné** sous forme de tableau par chacune des requêtes ci-dessous (considérer le système Oracle vu en TP) ainsi qu'**une phrase qui décrit le but de la requête** (seulement si elle est correcte). S'il agit d'une erreur d'exécution, expliquer cette dernière.

1. ```
   SELECT reference_composant, MIN(niveau_indice_gravite) AS minIG
   FROM IndicesGravites
   WHERE reference_composant = '2005'
   ORDER BY reference_composant;
   ```

   Erreur. Manque un group by pour pouvoir faire la projection avec l'agrégation

2. ```
   SELECT MAX(indice_reparabilite_smartphone) AS maxIR
   FROM Smartphones JOIN Composants USING (imei_smartphone)
                    JOIN IndicesGravites USING (reference_composant)
   WHERE marque_smartphone = 'Pixel';
   ```

   Donner l'indice de réparabilité maximum des Pixels qui ont au moins un défaut

   ```
   maxIR
   -----
   6.0
   ```

3. ```
   SELECT imei_smartphone
   FROM Smartphones
   MINUS
   SELECT imei_smartphone
   FROM Smartphones JOIN Composants USING (imei_smartphone)
                    JOIN IndicesGravites USING (reference_composant);
   ```

   Donner les smarphones sans defauts.

   ```
   imei_smartphone
   ---------------
   002
   003
   008
   009
   ```

2.2 Expression de requêtes en SQL et création d'une vue

Question 6 (6 points) :
Exprimer en SQL les requêtes ci-dessous. Les requêtes devront construire des résultats sans répétition de valeurs, la clause DISTINCT ne sera utilisée que lorsque nécessaire.

1. Donner la référence et le nom des composants 'processeur' qui ont un défaut avec un niveau de gravité plus que 1 (ex. 2006, processeur)

   ```
   SELECT DISTINCT reference_composant, nom_composant
   FROM Composants JOIN IndicesGravites USING (reference_composant)
   WHERE nom_composant = 'processeur' AND niveau_indice_gravite > 1;
   ```

2. Donner le type de défaut (code_type_defaut) qui n'affecte aucun smartphone (ex. 201)
   ```

```
SELECT code_type_defaut
FROM TypesDefauts
MINUS
SELECT code_type_defaut
FROM Defauts;
```

3. Donner l'imei, le modèle et la marque des smartphones avec le meilleur indice de réparabilité contenant au moins un défaut dans un de leurs composants (ex. 001, 4, Fairphone)

```
SELECT DISTINCT imei_smartphone,
                modele_smartphone,
                marque_smartphone
FROM Smartphones JOIN Composants USING (imei_smartphone)
                 JOIN IndicesGravites USING (reference_composant)
WHERE indice_reparabilite_smartphone IN (
   SELECT MAX (indice_reparabilite_smartphone)
   FROM Smartphones
);
```

4. Donner l'impact du type de défaut le plus répandu. C'est-à-dire, celui qui affecte le plus de composants (ex. technique).

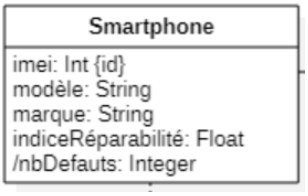
```
WITH ImpactsDefautsComposants AS (
   SELECT impact_type_defaut, COUNT(reference_composant) AS nbComposants
   FROM TypesDefauts JOIN Defauts USING (code_type_defaut)
                     JOIN IndicesGravites USING (numero_defaut)
   GROUP BY impact_type_defaut
)
SELECT impact_type_defaut
FROM ImpactsDefautsComposants
WHERE nbComposants IN (
   SELECT MAX(nbComposants)
   FROM ImpactsDefautsComposants
);
```

Question 7 (2 points) :
Donner le code SQL permettant de **créer** la table *IndicesGravites* (CREATE) et aussi le code SQL permettant d'**éliminer** correctement **toutes** les tables (DROP).

```
-- Création de la table
CREATE TABLE IndicesGravites (
   reference_composant INTEGER,
   numero_defaut INTEGER,
   niveau_indice_gravite INTEGER,
   CONSTRAINT pk_indice_gravite PRIMARY KEY (reference_composant, numero_defaut),
   CONSTRAINT fk_indice_gravite1 FOREIGN KEY (reference_composant)
      REFERENCES Composants(reference_composant),
   CONSTRAINT fk_indice_gravite2 FOREIGN KEY (numero_defaut)
      REFERENCES Defauts(numero_defaut),
   CONSTRAINT ck_indice_gravite CHECK (niveau_indice_gravite IN (1,2,3))
);

-- Elimination de tables (IF EXISTS optionnel)
DROP TABLE IF EXISTS IndicesGravites;
DROP TABLE IF EXISTS Defauts;
DROP TABLE IF EXISTS TypesDefauts;
DROP TABLE IF EXISTS Composants;
DROP TABLE IF EXISTS Smartphones;
```

Question 8 (2 points) :
Proposer le **code SQL de la vue** correspondant à l'extension ci-dessous du modèle UML (attribut calculé) : la vue donne toutes les informations relatives à **tous** les smartphones ainsi que le nombre de défauts par smartphone.




```
-- On utilise ici le patron vu en cours concernant la traduction
-- des classes avec attributs calculés.

--Sans jointure externe (préférable car vu en cours)
CREATE VIEW Smartphones AS
   SELECT imei_smartphone,
      modele_smartphone,
      marque_smartphone,
      indice_reparabilite_smartphone,
      COUNT (DISTINCT numero_defaut) AS nb_defauts_smartphone
   FROM Smartphones_base JOIN Composants USING (imei_smartphone)
                         JOIN IndicesGravites USING (reference_composant)
   GROUP BY imei_smartphone,
            modele_smartphone,
            marque_smartphone,
            indice_reparabilite_smartphone
   UNION
   SELECT imei_smartphone,
          modele_smartphone,
          marque_smartphone,
          indice_reparabilite_smartphone,
          0
   FROM Smartphones_base
   MINUS
   SELECT imei_smartphone,
          modele_smartphone,
          marque_smartphone,
          indice_reparabilite_smartphone,
          0
   FROM Smartphones_base JOIN Composants USING (imei_smartphone)
                         JOIN IndicesGravites USING (reference_composant)
;


--Avec jointure externe (pas vu en cours)

CREATE VIEW Smartphones AS
   SELECT imei_smartphone,
          modele_smartphone,
```

```
          marque_smartphone,
          indice_reparabilite_smartphone,
          COUNT (DISTINCT numero_defaut) AS nb_defauts_smartphone
   FROM Smartphones_base LEFT JOIN Composants USING (imei_smartphone)
                         LEFT JOIN IndicesGravites USING (reference_composant)
   GROUP BY imei_smartphone,
            modele_smartphone,
            marque_smartphone,
            indice_reparabilite_smartphone
;
```

3 Annexe

Smartphones

imei_smartphone	modele_smartphone	marque_smartphone	indice_reparabilite_smartphone
001	4	Fairphone	9.3
002	3+	Fairphone	8.7
003	4	Fairphone	9.3
004	14 Pro	Iphone	7.0
005	X	Iphone	4.8
006	12	Iphone	6.0
007	6	Pixel	6.0
008	6a	Pixel	6.5
009	Galaxy s23	Samsung	8.2
010	Galaxy s23	Samsung	8.2

Composants

reference_composant	nom_composant	imei_smartphone
1001	écran	001
1004	camera	004
1005	capteur GPS	005
1006	camera	006
1007	écran	007
1010	mémoire	010
2001	antenne	001
2004	haut-parleur	004
2005	batterie	005
2006	processeur	006
2007	coque	007
2010	batterie	010
3005	processeur	005

IndicesGravites

reference_composant	numero_defaut	niveau_indice_gravite
1001	1	1
2001	2	3
1004	3	2
2004	4	1
1005	5	1
2005	6	3
2005	7	3
3005	7	3
1006	8	1
2006	8	2
1007	9	1
2007	9	1
1010	10	3
2010	11	3

Defauts

numero_defaut	description_defaut	code_type_defaut
1	écran cassé	100
2	antenne capte plus	101
3	caméra tout noir	101
4	haut-parleurs sans son	100
5	capteur GPS en défaut	101
6	accumulateur batterie en défaut	300
7	surchauffe générale	101
8	couvert de poussière	202
9	rayures profondes	200
10	virus ransomware	402
11	batterie surchauffe et gonfle	301

TypesDefauts

code_type_defaut	nom_type_defaut	impact_type_defaut
100	casse	technique
101	court-circuit	technique
200	rayure	esthétique
201	tâche	esthétique
202	saleté	esthétique
300	cycles dépassés	batterie
301	explosive	batterie
401	obsolescence	logiciel
402	virus	logiciel