Génie Logiciel : Base de données

Christian Ingouff
Pierre-Alexandre Tyndal
Hugo Dos

EISTI

2013/2014
Semestre 1

Table des matières

Chapitre 1

Introduction

Ce projet de gestion de questionnaires consiste, grâce à une base de données et de scripts SQL, à récupérer les différents questionnaires et les différents calculs possibles, propres à chaque type de questionnaire.

Ayant au cours des derniers livrables construit au fur et à mesure la base de données, qui est désormais complète et fonctionnelle sous Oracle 10g, nous avons à notre disposition la base sur laquelle nous allons appliquer le projet que nous avons exposé par le biais du Microsoft Excel du deuxième jalon.

Vous trouverez dans ce rapport le détail des requêtes SQL mises en avant par notre réflexion ainsi que le développement technique et la justification de celles-ci.

Chapitre 2

Modèle conceptuel de données

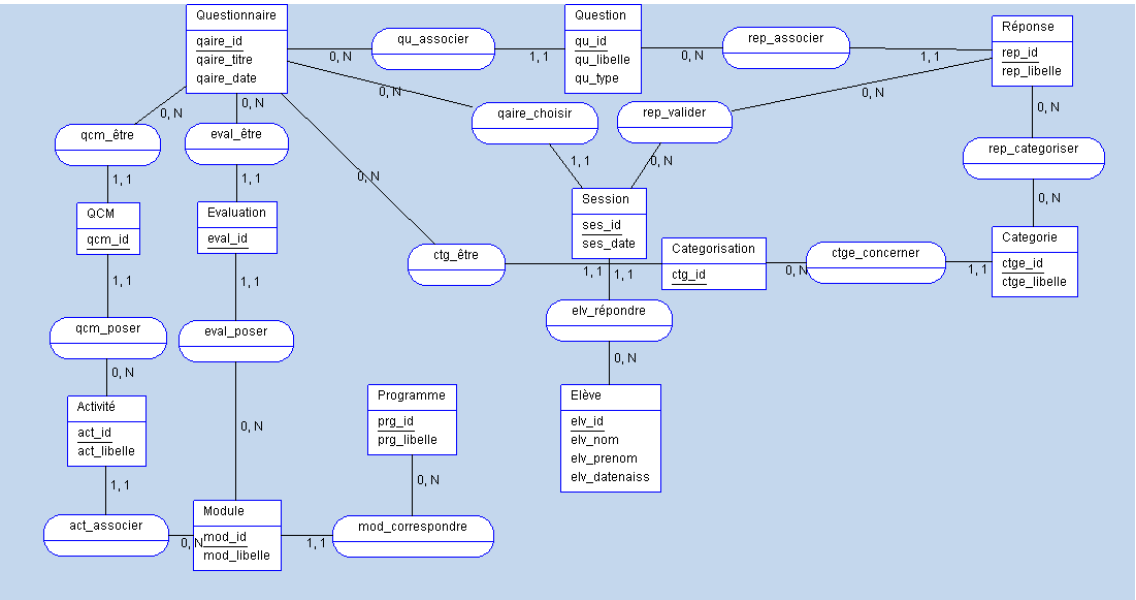

FIGURE 2.1 – Modèle conceptuel de données

Nom ▲	ID	
		AUTO_INCREMENT
act_id	act_id	AUTO_INCREMENT
act_libelle	act_libelle	CHAR
ctg_id	ctg_id	AUTO_INCREMENT
ctge_id	ctge_id	AUTO_INCREMENT
ctge_libelle	ctge_libelle	CHAR
elv_datenaiss	elv_datenaiss	DATE
elv_id	elv_id	AUTO_INCREMENT
elv_nom	elv_nom	CHAR
elv_prenom	elv_prenom	CHAR
eval_id	eval_id	AUTO_INCREMENT
mod_id	mod_id	AUTO_INCREMENT
mod_libelle	mod_libelle	CHAR
prg_id	prg_id	AUTO_INCREMENT
prg_libelle	prg_libelle	CHAR
qaire_date	qaire_date	DATE
qaire_id	qaire_id	AUTO_INCREMENT
qaire_titre	qaire_titre	CHAR
qcm_id	qcm_id	AUTO_INCREMENT
qu_id	qu_id	AUTO_INCREMENT
qu_libelle	qu_libelle	CHAR
qu_type	qu_type	CHAR
rep_id	rep_id	AUTO_INCREMENT
rep_libelle	rep_libelle	CHAR
ses_date	ses_date	DATE
ses_id	ses_id	AUTO_INCREMENT

FIGURE 2.2 – Dictionnaire de données

Nous rappelons ici notre MCD, qui fonctionne de la même manière que dans le dernier livrable. Cependant, nous avons entre temps ajouté une entité Catégorie ainsi que 2 associations connexes pour permettre le traitement de questionnaires de ce type.

2.1 Les entités

- **Questionnaire :** Ensemble des questionnaires, contient un id unique, un titre et une date de création
- **QCM, Catégorisation, Evaluation :** types de questionnaires, contient un id unique.
- **Catégorie :** Elle appartient à une catégorisation, ce sont les résultats possibles pour ce genre de questionnaire. Contient un id unique ainsi que son intitulé.
- **Question :** Ensemble des questions. Ces questions appartiennent à un questionnaire. Contient un id unique, un libellé (le texte de la question) et le type de question (fermée, ouverte, etc...).
- **Réponse :** Ensemble des réponses. Ces réponses sont sous-jacentes à une question. Contient un id unique et le texte correspondant à la réponse.
- **Session :** Définie quand un élève répond à un questionnaire. Contient un id unique et une date de session.
- **Elève, Programme, Module, Activité :** Ces entités représentent comme leur nom l'indique l'élève, les programmes de l'EISTI et enfin les modules et les activités des différents programmes.

2.2 Les associations

- **rep_associer :** Une question proposant des réponses, cette association fait le lien entre la réponse et son unique (cardinalité 1,1) question correspondante. Si une question est ouverte, les réponses qui lui sont associées vont être les réponses entrées par les utilisateurs lors des sessions.
- **qu_associer :** Un questionnaire proposant des questions, cette association fait le lien entre la question et le questionnaire où elle figure.
- **qcm_être, eval_être, ctg_être :** Définition du type de questionnaire
- **qcm_poser, eval_poser :** Définition du contexte dans lequel le questionnaire est posé (dans quelle activité pour les QCM, dans quel module

pour les évaluations)

- **act_associer, mod_correspondre :** Association hiérarchique de données : une activité est contenue dans un module, un module est contenu dans un programme
- **elv_répondre :** Quand un élève répond à un questionnaire, il génère une session. elv_répondre fait le lien entre la session et l'élève.
- **qaire_choisir :** Une session choisit forcément un et un seul questionnaire
- **rep_valider :** Une session enregistrera des réponses choisies
- **ctge_concerner :** Lien entre la catégorie et sa catégorisation
- **rep_catégoriser :** Les réponses qui vont correspondre à une catégorie en particulier vont être reliées par cette association. C'est ainsi que l'on identifiera plusieurs catégories dans un questionnaire de catégorisation.

Chapitre 3

Modèle logique de données

Le modèle conceptuel de données décrit ci-avant se traduit ainsi en modèle logique de données :

- Questionnaire(<u>id</u>, titre, qa_date) ;
- Question(<u>id</u>, libelle, qu_type, #qa_id) ;
- Reponse(<u>id</u>, libelle, #qu_id) ;

- Eleve(<u>id</u>, nom, prenom, datenaiss) ;
- Programme(<u>id</u>, libelle) ;
- Module(<u>id</u>, libelle, #prg_id) ;
- Activite(<u>id</u>, libelle, #mod_id) ;

- QCM(<u>id</u>, #qa_id, #act_id) ;
- Evaluation(<u>id</u>, #qa_id, #mod_id) ;
- Categorisation(<u>id</u>, #qa_id) ;
- Categorie(<u>id</u>, libelle, #ctgn_id) ;
- rep_categoriser(<u>#rep_id, #ctg_id</u>) ;

- Session(<u>id</u>, ses_date, #elv_id, #qa_id) ;
- rep_valider(<u>#ses_id, #rep_id</u>) ;

Pour les références des clés étrangères, on se réfère au préfixe de l'attribut en question, avec :

- qa : Questionnaire
- qu : Question
- ctgn : Catégorisation
- ctg : Catégorie
- prg : Programme
- mod : Module
- act : Activité
- elv : Elève
- ses : Session
- rep : Réponse

Chapitre 4

Opérations SQL

4.1 Utilisation

Nous fournissons dans le dossier *src* de ce livrable quatre fichiers .sql :
- create.sql : Ce fichier permet de créer les tables de notre base de données
- insert.sql : Ce fichier permet d'insérer les données de notre jeu d'essai dans la base de données
- clean.sql : Ce fichier permet de supprimer les tables (et leurs données) de notre base de données
- queries.sql : Ce fichier contient l'ensemble des requêtes SQL développées dans ce livrable.

Nous avons effectué notre travail à l'aide d'Oracle 10g : il est donc recommandé d'utiliser cette version pour utiliser nos fichiers. Dedans, on crée et on remplit la base de données avec ces deux commandes successives :

```
start <chemin du fichier>/create.sql
start <chemin du fichier>/insert.sql
```

On supprime les tables quand on en a plus besoin avec la commande :

```
start <chemin du fichier>/clean.sql
```

Il est recommandé de copier et coller chaque requête présente depuis le fichier queries.sql jusqu'à la console SQL : ainsi, le résultat est facilement visible.

4.2 Requêtes SQL

4.2.1 Requêtes simples

```
SELECT *
FROM Eleve
ORDER BY datenaiss ASC;
```

Simple affichage de la liste des élèves du plus vieux au plus jeune (avec ORDER BY).

```
SELECT *
FROM Eleve
WHERE datenaiss = (
   SELECT min(datenaiss)
   FROM Eleve
);
```

Recherche de l'élève le plus âgé, avec la fonction min() qui cherche le minimum.

```
SELECT count(id)
FROM Questionnaire
WHERE id IN (
    SELECT id
    FROM Categorisation
);
```

Affichage du nombre de questionnaires appartenant au type catégorisation. On note ici le lien entre les deux tables avec une sous-requête et l'utilisation de la fonction count().

```
SELECT qu.libelle as question, r.libelle as reponse
FROM Question qu, Reponse r
WHERE r.qu_id = qu.id
AND qu.id = 14;
```

Cette requête affiche l'énoncé de la question numéro 14. Comme on veut afficher à la fois la question et la réponse, on a recours ici à une jointure plutôt qu'à une sous-requête. On voit également la notion de restriction.

```
SELECT P.libelle AS programme, M.libelle AS module, A.libelle AS activite
FROM Activite A, Module M, Programme P
WHERE A.mod_id = M.id
AND M.prg_id = P.id
ORDER BY P.id ASC;
```

Sous le même principe, on fait la liste des programmes, des modules et des activités. Comme on peut le voir, les jointures ne sont pas limitées à deux tables.

```
SELECT e.nom as eleve, q.libelle as question, r.libelle as reponse
FROM Eleve e, Question q, Reponse r, Session_ s, rep_valider v
WHERE e.id = s.elv_id
AND v.ses_id = s.id
AND v.rep_id = r.id
AND r.qu_id = q.id
AND s.id = 2;
```

Nous avons les réponses enregistrées dans la session 2 avec leurs questions correspondantes. Ceci est un exemple de requête avec de multiples jointures, et il est bien évidemment possible d'en faire davantage.

```
SELECT e.nom, e.prenom, count(s.id) AS nombre
FROM Eleve e LEFT JOIN Session_ s
ON s.elv_id = e.id
GROUP BY e.nom, e.prenom;
```

Affichage du nombre de questionnaires auxquels chaque élève a répondu. On note l'utilisation du LEFT JOIN qui permet d'afficher les élèves qui n'ont répondu à aucun questionnaire. On utilise également GROUP BY avec count() pour compter selon chaque élève.

```
SELECT libelle
FROM Reponse
WHERE qu_id = (
    SELECT id
    FROM Question
    WHERE qu_type = 'ouverte'
    AND qa_id = (
        SELECT id
        FROM Questionnaire
        WHERE id = (
            SELECT qa_id
            FROM Evaluation
            WHERE id = 1
        )
    )
);
```

Liste des remarques en question ouverte itérées pour l'évaluation numéro 1. On note l'utilisation du type de question, qui permet de différencier les questions fermées aux questions ouvertes. Les sous-requêtes à répétition sont aussi une alternative plus optimisée en mémoire que les jointures si on peut se le permettre.

```
SELECT qu.libelle, count(r.id)
FROM Question qu, Reponse r
WHERE r.qu_id = qu.id
AND qu.qa_id = (
    SELECT id
    FROM Questionnaire
    WHERE id = (
        SELECT qa_id
        FROM QCM
        WHERE id = 1
    )
)
GROUP BY qu.libelle;
```

Nombre de réponses, avec les questions, proposées pour chaque question dans le QCM numéro 1. Ceci est un exemple de combinaison de plusieurs procédés vus précédemment.

```
SELECT qu.libelle AS question, r.libelle AS reponse
FROM Question qu, Reponse r
WHERE r.qu_id = qu.id
AND qu.qa_id IN (
    SELECT id
    FROM Questionnaire
    WHERE id IN (
        SELECT qa_id
        FROM Evaluation
        WHERE mod_id IN (
            SELECT id
            FROM Module
            WHERE prg_id = (
                SELECT id
                FROM Programme
                WHERE libelle = 'ING 1'
            )
        )
    )
)
AND r.id IN (
    SELECT rep_id
    FROM rep_valider
    WHERE ses_id IN (
        SELECT id
        FROM Session_
        WHERE elv_id = (
            SELECT id
            FROM Eleve
            WHERE lower(nom) = 'tyndal'
            AND lower(prenom) = 'pierre-alexandre'
        )
    )
);
```

On récupére ici les réponses (avec questions) données par TYNDAL Pierre-Alexandre dans les évaluations du programme ING 1. On peut voir ici le parcours hiérarchique des tables composant notre base de données.

4.2.2 Requêtes techniques

Les requêtes suivantes vont directement traiter de l'énoncé du projet qui nous a été donné. Il est également à mettre en parallèle avec la modélisation que nous avons faite du projet sur Microsoft Excel.

```
SELECT qu.libelle AS question, r.libelle AS reponse, count(rv.rep_id) AS voix
FROM Question qu, Reponse r LEFT JOIN rep_valider rv
ON r.id = rv.rep_id
WHERE r.qu_id = qu.id
AND qu.qa_id = (
    SELECT qa_id
    FROM Evaluation
    WHERE id = 1
)
GROUP BY qu.libelle, r.libelle
ORDER BY qu.libelle;
```

Cette requête permet de visualiser les statistiques de l'évaluation numéro 1, c'est-à-dire combien de personnes ont choisi telle réponse parmi celles disponibles.

```
SELECT categ, score FROM (
    SELECT c.libelle AS categ, count(rc.rep_id) AS score
    FROM Categorie c LEFT JOIN rep_categoriser rc
    ON c.id = rc.ctg_id
    AND rc.rep_id IN (
        SELECT rep_id
        FROM rep_valider
        WHERE ses_id = (
            SELECT id
            FROM Session_
            WHERE elv_id = (
                SELECT id
                FROM Eleve
                WHERE lower(nom) = 'dos'
                AND lower(prenom) = 'hugo'
            )
            AND qa_id = (
                SELECT qa_id
                FROM Categorisation
                WHERE id = 1
            )
        )
    )
    GROUP BY c.libelle
    ORDER BY score DESC
)
WHERE ROWNUM = 1;
```

On obtient la catégorie à laquelle appartient DOS Hugo selon la catégorisation numéro 1. En effet, la catégorie à laquelle cet élève appartient correspond à celle avec laquelle il a le plus d'affinités, c'est-à-dire de réponses communes avec celles enregistrées dans *rep_ categoriser* pour une catégorie. Ainsi, on fait le compte de réponses correspondantes avec toutes les catégories, puis on prend le maximum (ORDER BY décroissant, puis prise du premier résultat).

```
SELECT qu.libelle as question, r.libelle as reponse
FROM Question qu, reponse r
WHERE r.qu_id = qu.id
AND r.id IN (
    SELECT rep_id
    FROM rep_valider
    WHERE ses_id = (
        SELECT id
        FROM Session_
        WHERE elv_id IS NULL
        AND qa_id = (
            SELECT id
            FROM Questionnaire
            WHERE id = (
                SELECT qa_id
                FROM QCM
                WHERE id = 1
            )
        )
    )
)
ORDER BY qu.id ASC;
```

On affiche ici les bonnes réponses du QCM numéro 1. Il est intéressant de noter ici notre gestion des bonnes réponses d'un QCM. En effet, nous générons une session sans id d'élève pour signifier qu'il s'agit d'une solution de QCM et non d'un élève qui y répond.

```
SELECT rep_id
FROM rep_valider
WHERE ses_id IN (
    SELECT id
    FROM Session_
    WHERE elv_id IS NULL
)
INTERSECT
SELECT rep_id
FROM rep_valider
WHERE ses_id IN (
    SELECT id
    FROM Session_
    WHERE elv_id = (
        SELECT id
        FROM Eleve
        WHERE lower(nom) = 'van damme'
        AND lower(prenom) = 'jean-claude'
    )
    AND qa_id = (
        SELECT qa_id
        FROM QCM
        WHERE act_id IN (
            SELECT id
            FROM Activite
            WHERE mod_id = (
                SELECT id
                FROM Module
                WHERE libelle = 'Transverse'
                AND prg_id = (
                    SELECT id
                    FROM Programme
                    WHERE libelle = 'ING 1'
                )
            )
        )
    )
);
```

Ceci est une manière d'obtenir les bonnes réponses de Jean-Claude Van Damme au QCM se rapportant à la Transverse des ING 1. Il utilise l'in-

tersection entre deux résultats : l'ensemble des bonnes réponses de tous les QCM avec l'ensemble des réponses données par l'élève dans le QCM précisé. La comparaison se fait avec l'ensemble des bonnes réponses de tous les QCM pour éviter d'appeler inutilement des tables en plus : si on précise d'un côté les modalités du QCM, l'autre côté sera filtré par l'intersection.

Egalement, on peut obtenir les mauvaises réponses de l'élève si on fait la négation pour la liste des bonnes réponses (donc si on obtient la liste des mauvaises réponses des QCM).

En algèbre relationnelle, la requête se traduit ainsi :

Bonnes réponses des QCM :

$$\begin{array}{lcl} R_1 & = & \pi_{\text{id}}(\sigma_{\text{elv\_id = NULL}}(\text{Session})) \\ R_{\text{réponses}} & = & \pi_{\text{rep\_id}}(\sigma_{\text{ses\_id}=R_1}(\text{rep\_valider})) \end{array}$$

Sélection du bon questionnaire :

$$\begin{array}{lcl} R_1 & = & \pi_{\text{id}}(\sigma_{\text{libelle = 'ING 1'}}(\text{Programme})) \\ R_2 & = & \pi_{\text{id}}(\sigma_{\text{libelle = 'Transverse' ET prg\_id}=R_1}(\text{Module})) \\ R_3 & = & \pi_{\text{id}}(\sigma_{\text{mod\_id}=R_2}(\text{Activité})) \\ R_{\text{QCM}} & = & \pi_{\text{qa\_id}}(\sigma_{\text{act\_id}=R_3}(\text{QCM})) \end{array}$$

Sélection de Jean-Claude Van Damme :

$$R_{\text{JCVD}} = \pi_{\text{id}}(\sigma_{\text{nom = 'Van Damme' ET prenom = 'Jean-Claude'}}(\text{Elève}))$$

Les réponses de Jean-Claude Van Damme :

$$R_{rep\_JCVD} = \pi_{\text{id}}(\sigma_{\text{elv\_id}=R_{JCVD}\text{ETqa\_id}=R_{QCM}}(\text{rep\_valider}))$$

Résultat final :

$$R = R_{\text{réponses}} \cap R_{\text{rep\_JCVD}}$$

```
SELECT e.nom, e.prenom, count(rv.rep_id) AS score
FROM Eleve e, Session_ s, rep_valider rv
WHERE e.id = s.elv_id
AND s.id = rv.ses_id
AND rv.rep_id IN (
    SELECT rep_id
    FROM rep_valider
    WHERE ses_id = (
        SELECT id
        FROM Session_
        WHERE elv_id IS NULL
        AND qa_id = (
            SELECT qa_id
            FROM QCM
            WHERE id = 1
        )
    )
)
GROUP BY e.nom, e.prenom
HAVING count(rv.rep_id) >= (
    SELECT avg(score) FROM (
        SELECT s.elv_id, count(rv.rep_id) AS score
        FROM Session_ s, rep_valider rv
        WHERE s.id = rv.ses_id
        AND rv.rep_id IN (
            SELECT rep_id
            FROM rep_valider
            WHERE ses_id = (
                SELECT id
                FROM Session_
                WHERE elv_id IS NULL
                AND qa_id = (
                    SELECT qa_id
                    FROM QCM
                    WHERE id = 1
                )
            )
        )
        GROUP BY s.elv_id
    )
);
```

Finalement, cette requête renvoie la liste des élèves ayant davantage de bonnes réponses que la moyenne des répondants au QCM numéro 1. Elle fait d'abord la liste des élèves avec leur nombre de bonnes réponses puis filtre avec HAVING pour placer la barre à la moyenne des répondants.

Chapitre 5

Conclusion

C'est ainsi que nous pouvons contempler la finalisation de notre projet sur l'aménagement de questionnaires sur Oracle 10g. Nous avons ainsi constaté qu'avec une base de données à première vue relativement simple, on pouvait effectuer une multitude d'opérations, et ceci avec le SQL.

Ainsi, nous voyons clairement le potentiel d'un tel aménagement dans des projets de plus grande envergure. De plus, nous avons pris conscience qu'une bonne organisation de la base de données était primordiale dans l'aménagement des requêtes. Il va de soi que la tâche n'est pas forcément simple, ni pratique, en notant l'absence d'une réelle interface pratique.

Néanmoins, cela a développé notre esprit débrouillard et notre esprit d'équipe. Il ne reste plus qu'à résumer nos efforts dans ce dernier livrable qui portera de la synthèse et du parachèvement de notre projet.