Documentation

Projet Base de données

Équipe 7: CHARBONNEL Antoine / KERZREHO Meven / NOBLET Baptiste / PALISSE Léandre / SOUM-FONTEZ Louis

I – ANALYSE DU PROBLEME :

La première étape de ce projet a consisté en une analyse du problème. A partir de la description de l'application, nous avons pu déterminer les propriétés élémentaires du problème ainsi qu'un ensemble de contraintes. Voici les résultats obtenus :

- **Un ensemble de données :**

{ codeNumeroCandidat, titreNumero, resume, dureeNumero, nombreArtistes, artistePrincipal, idArtiste, nom, prenom, dateNaissance, cirqueOrigine, numeroTel, pseudo, theme, evaluation, note, idSpectacle, jourSpectacle, heureDebut, prixSpectacle }

- **Un ensemble de contraintes :**

1. Dépendances fonctionnelles :

   - CodeNumeroCandidat --> titreNumero, theme, resume, dureeNumero, nombreArtistes, artistePrincipal
   - idArtiste --> nom, prenom, dateNaissance, cirqueOrigine, pseudo, numeroTel, theme
   - idSpectacle --> jourSpectacle, heureDebut, prixSpectacle
   - idArtiste, codeNumeroCandidat → evaluation, note

2. Contraintes de valeur :

   - 10min ≤ dureeNumero ≤ 30min
   - nombreArtistes ≥ 1
   - prixSpectacle ≥ 0
   - note ∈ [0,10]
   - heureDebut ∈ {9,14}

3. Contraintes de multiplicité :

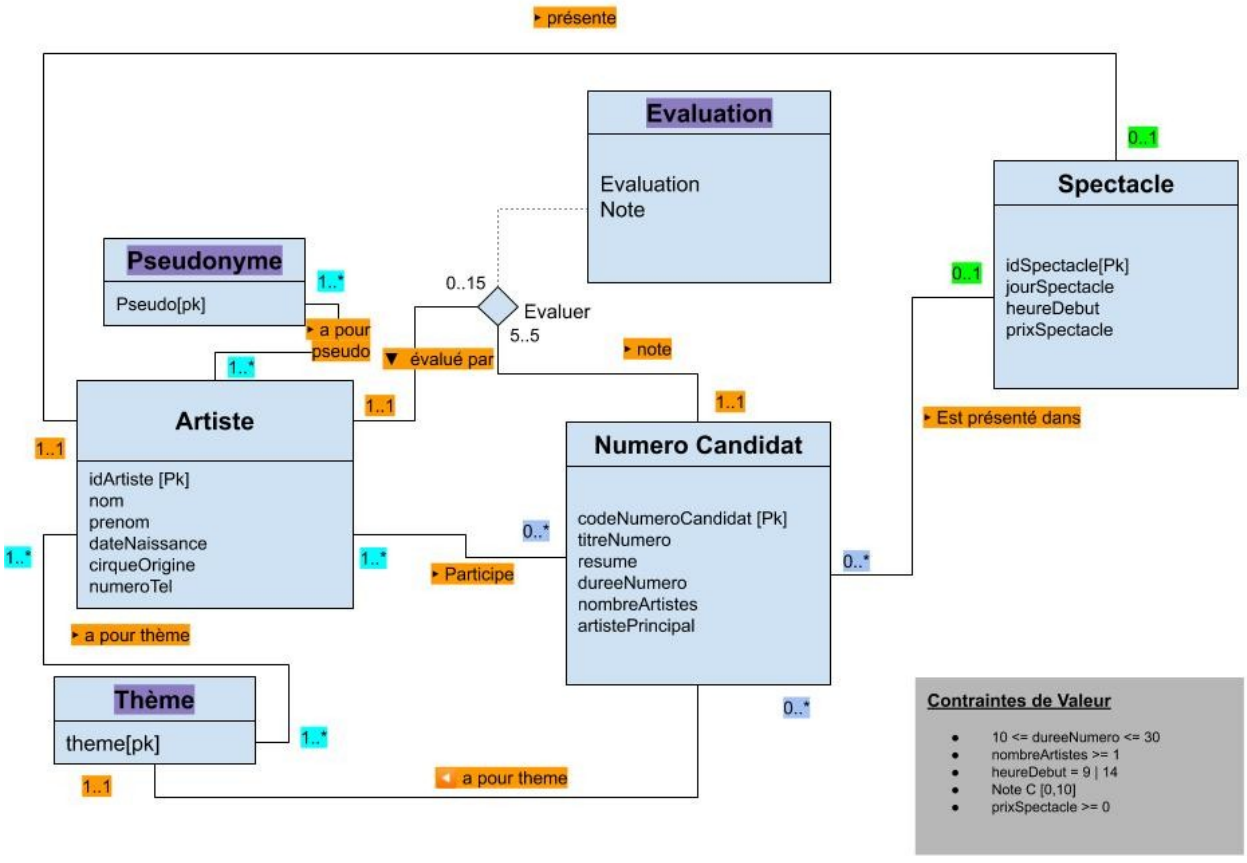
   - " un artiste participe a un ou plusieurs numéros "
   - " un ou plusieurs numéros dans un spectacle "
   - " un artiste a au moins une spécialité "
   - " un artiste a 1 ou plusieurs pseudonymes "
   - " un artiste peut avoir des évaluations, des notes"
   - " un numeroCandidat peut avoir au plus un spectacle"

4. Contraintes contextuelles :

- La somme des durées des numéros d'un spectacle est inférieure ou égale à 180min
- Un expert ne peut pas être du même cirque d'origine que les artistes des numéros qu'il évalue
- Tous les artistes d'un numéro appartiennent au même cirque
- Tous les artistes d'un numéro ont la spécialité du numéro
- Un expert ne peut pas proposer de numéro, donc être l'artiste principal
- Le présentateur d'un spectacle ne peut pas également être un artiste dans un numéro de ce spectacle
- Il y a autant d'évaluations que de notes pour un expert
- 5 experts par numéro
- 3 spécialistes et 2 non spécialistes parmi les 5 experts
- Il ne peut pas y avoir plus de 15 évaluations par expert
- Tous les numéros d'un spectacle ont le même thème

II – Schéma Entités/Associations :

Grâce à l'analyse précédemment effectuée, nous avons pu construire le schéma entités-associations suivant :



III – Schéma relationnel :

On détermine le schéma relationnel à partir des multiplicités indiquées dans le schéma entités-associations :

- artiste [1..*]→Numero Candidat [0..*]

NumeroCandidat(codeNumeroCandidat, titreNumero, themeNumero, resume, dureeNumero, nombreArtistes, artistePrincipal)

A_participeA_nc(idArtiste, codeNumeroCandidat)

- idArtiste non nul et référence Artiste
- Vérifier qu'un numéro candidat ait au moins un artiste

- artiste[1..1]→spectacle[0..1]

Spectacle(idSpectacle, jourSpectacle, heureDebut, prixSpectacle)
Artiste(idArtiste, nom, prenom, dateNaissance, cirqueOrigine, numeroTel)

A_presente_s(idArtiste, id_spectacle)

- idArtiste non nul et référence Artiste
- idSpectacle non nul et référence Spectacle

- numeroCandidat[0..*]→spectacle[0..1]

NC_presenteDans_s(codeNumeroCandidat, idSpectacle)

- codeNumeroCandidat non nul et référence NumeroCandidat
- idSpectacle non nul et référence Spectacle

- Artiste[1..1]→Pseudonyme[1..*]

A_àPour_p(idArtiste, pseudo)

- idArtiste non nul et référence Artiste
- pseudo non et référence pesudonyme
- Vérifier qu'un artiste ait au moins un pseudonyme

- Artiste[0..*]→theme[1..*]

A_àPour_t(idArtiste, specialite)

- idArtiste non nul et référence Artiste
- specialite non nul et référence theme
- Vérifier qu'un artiste ait au moins un thème

- NumeroCandidat[1..*]→theme[1..1]

NumeroCandidat(…, theme)

- theme non nul et référence theme

Evaluer(Artiste, NumeroCandidat, Evaluation, Note])

- artiste référence Artiste
- NumeroCandidat référence NumeroCandidat
- Vérifier qu'un artiste ait au plus 5 évaluations
- Vérifier qu'un numéro candidat ait exactement 5 évaluations

On obtient ainsi le **schéma relationnel** suivant :

- **Artiste**(idArtiste, nom, prenom, dateNaissance, cirqueOrigine, numeroTel)
- **NumeroCandidat**(codeNumeroCandidat, titreNumero, resume, dureeNumero, nombreArtistes, artistePrincipal, theme)
- **Theme**(theme)
- **Spectacle**(idSpectacle, jourSpectacle, heureDebut, prixSpectacle)
- **A_presente_s**(idArtiste, id_spectacle)
- **A_participeA_nc**(CodeNumeroCandidat, idArtiste)
- **NC_presenteDans_s**(codeNumeroCandidat, idSpectacle)
- **A_àPour_p**(idArtiste, pseudo)
- **Pseudo**(pseudo)

- **A__àPour__t**(idArtiste, theme)

- **Evaluer**(idArtiste, codeNumeroCandidat, evaluation, note])

IV – Formes normales des relations :

Toutes les relations obtenues sont **3FNBCK** ce qui implique une absence de redondance. Nous allons le montrer sur quelques exemples types :

- relation **Artiste** :

    - Clé : {{idArtiste}}
    - Attribut clé : {idArtiste}
    - Attributs non clés : {nom, prenom, dateNaissance, cirqueOrigine, numeroTel}

La relation est 2FN car il n'y a qu'un seul attribut clé
La relation est 3FN car il n'y a pas de dépendances fonctionnelles entre les attributs non clés
La relation est 3FNBCK car seul la clé est en partie gauche dans les dépendances fonctionnelles.

- relation **A__aPour__p** :

    - Clés : {{idArtiste, pseudo}}
    - Attribut clé : {idArtiste, pseudo}
    - Attributs non clés : {}

La relation est 2FN car il n'y a pas d'attribut non clé.
La relation est 3FN pour la même raison.
La relation est 3FNBCK car il n'y pas de dépendance fonctionnelle entre idArtiste et pseudo.

- relation **Evaluer** :

    - Clés : {{idArtiste, codeNumeroCandidat}}
    - Attribut clé : {idArtiste, codeNumeroCandidat}
    - Attributs non clés : {evaluation, note}

La relation est 2FN car evaluation et note dépendent de idArtiste et codeNumeroCandidat.
La relation est 3FN car il n'y a pas de dépendance fonctionnelle entre les attributs clés.
La relation est 3FNBCK car seule la clé est en partie gauche dans les dépendances fonctionnelles.

V – Implantation des fonctionnalités :

On liste dans cette partie les requêtes SQL utilisées pour vérifier les contraintes. Il est souvent nécessaire de faire un traitement sur ces requêtes pour obtenir le résultat souhaité.

- Existence d'un numéro candidat dans la table :

```
SELECT COUNT(*) FROM NUMEROCANDIDAT WHERE codeNumeroCandidat =
codeNumeroCandidat
```

- Un numero candidat peut avoir au plus un spectacle :

```
SELECT IDSPECTACLE FROM NC_PRESENTEDANS_S WHERE CODENUMEROCANDIDAT
= id
```

- La somme des durées des NC d'un spectacle est inférieure à 180min :

```
SELECT DUREENUMERO FROM NUMEROCANDIDAT WHERE CODENUMEROCANDIDAT =
id
```

- Tous les NC d'un spectacle ont le même thème :

```
SELECT THEME FROM NUMEROCANDIDAT WHERE CODENUMEROCANDIDAT =
codeNumeroCandidat
```

- Sélectionner les artistes d'un même thème :

```
SELECT DISTINCT IDARTISTE FROM A_APOUR_T WHERE THEME = theme
```

- Déterminer le présentateur potentiel d'un spectacle sachant que le présentateur ne peut pas être également un artiste d'un des numéros du spectacle :

```
SELECT IDARTISTE, NOM, PRENOM FROM ARTISTE  WHERE IDARTISTE NOT IN
(SELECT IDARTISTE  FROM A_PARTICIPEA_NC INNER JOIN NC_PRESENTEDANS_S ON
        A_PARTICIPEA_NC.CODENUMEROCANDIDAT =
        NC_PRESENTEDANS_S.CODENUMEROCANDIDAT
                WHERE IDSPECTACLE = idSpectacle)
```

- Numéros Candidats avec moins de cinq évaluations :

```
SELECT NUMEROCANDIDAT.codeNumeroCandidat FROM NUMEROCANDIDAT JOIN
EVALUER ON NUMEROCANDIDAT.codeNumeroCandidat = EVALUER.codeNumeroCandidat
GROUP BY NUMEROCANDIDAT.codeNumeroCandidat HAVING
COUNT(NUMEROCANDIDAT.codeNumeroCandidat) < 5
```

- Numéros Candidats avec aucune évaluation :

```
SELECT codeNumeroCandidat FROM NUMEROCANDIDAT WHERE codeNumeroCandidat
NOT IN (SELECT codeNumeroCandidat FROM EVALUER)
```

- Les experts avec moins de 15 évaluations :

```
SELECT ARTISTE.idArtiste FROM ARTISTE JOIN EVALUER ON ARTISTE.idArtiste =
EVALUER.idArtiste GROUP BY ARTISTE.idArtiste HAVING COUNT(ARTISTE.idArtiste) < 15
```

- Experts n'évaluent pas les numéros de leurs cirques :

```
SELECT idArtiste FROM ARTISTE WHERE cirqueOrigine = cirque
```

- Experts n'évaluent pas deux fois le même numéro :

```
SELECT idArtiste FROM EVALUER WHERE codeNumeroCandidat = numero
```

VI – Mode d'emploi de l'application :

Le script qui lance l'application est "UserInterface". Dans un premier temps, il permet de créer les tables si elles ne sont pas déjà crées. Un menu permet ensuite de sélectionner l'action désirée :

1. Insertion : ce choix permet l'insertion d'un artiste, d'un numéro candidat et d'un spectacle.

    - Lors de l'insertion d'un **artiste**, il est demandé d'ajouter en plus de son identité son/ses pseudonyme(s) ainsi que son/ses thème(s).

    - Lors de l'insertion d'un **numéro candidat**, il est nécessaire que les artistes participant à ce numéro aient déjà été crées au préalable. On sélectionne en premier l'artiste principal du numéro ainsi que le thème du numéro. Ensuite, il suffit simplement de rentrer les informations concernant le numéro candidat.

- Lors de l'insertion d'un **spectacle**, il est nécessaire d'avoir créé au préalable les numéro candidat présents dans le spectacle et le présentateur du spectacle (qui est un artiste). On rentre ensuite les informations concernant le spectacle.

2. Évaluation des numéros : ce choix permet d'évaluer des numéros candidats, soit qui n'ont pas encore été évalués, soit qui nécessitent encore des évaluations. Les artistes proposés par notre application sont déterminés en tenant compte des contraintes définies dans l'énoncé du problème. Tout artiste qui évalue un numéro est alors un expert.
   Attention, il est nécessaire de rentrer tous les numéros avant de commencer à évaluer afin de respecter la contrainte « Un expert ne peut pas proposer de numéro, donc être l'artiste principal ».

3. Planification des spectacles : ce choix permet d'afficher pour chaque thème la liste des numéros avec leur évaluation.

4. Quitter : permet de fermer l'application

VII – Bilan du projet:

Structure :

La base de données permet aux organisateurs du festival d'ajouter des artistes, des numéros, des évaluations de ces numéros et de consulter ces évaluations afin de préparer leur prochain spectacle. La base de données permet de centraliser toutes ces données. L'utilisation de la base de données est explicite, les attributs nécessaires étant tous nommés et l'erreur étant permise sans pour autant faire planter le programme ou mener à la perte de données. Pour assurer ce dernier point, un système de transactions à base de commits et de rollback a été mise au point.

Organisation :

L'organisation de ce projet s'est faite en utilisant un dépôt Git grâce à la plateforme Gitlab. Jusqu'à la fin de l'analyse des contraintes et des dépendances

fonctionnelles, nous avons divisé notre groupe en deux afin d'avancer deux parties en parallèle : les dépendances fonctionnelles et les différentes contraintes. L'établissement de ces contraintes nous a posé quelques soucis, une majeure partie de nos contraintes de valeurs se révélant être en fait des contraintes de multiplicité. Réaliser cette erreur et l'ajuster a été crucial dans la création de notre schéma Entité-associations.

Un groupe s'est ensuite concentré sur la création du schéma relationnel tandis que l'autre commençait la partie Java du projet. Bien que nous pensions partir sur une interface graphique au début du projet, des contraintes de temps nous ont imposé le choix d'une interface textuelle dans la console.

Une fois le schéma relationnel terminé, nous avons pu véritablement commencer la partie Java, avec une création propre des tables (que l'on peut remplir avec des données semi-aléatoires pour tester).

L'insertion suivant les contraintes s'est révélé être la partie la plus longue du projet, celles-ci étant nombreuses. Il nous tenait à cœur de pouvoir afficher les numéros à évaluer et les artistes disponibles. Cela a donc été une autre facette importante du travail effectué afin de rendre le tout lisible.

Finalement, malgré nos difficultés, l'ensemble du projet répond au cahier des charges posé par le sujet.