Courses automobiles

On désire automatiser la gestion de courses automobiles.

Le propriétaire (*propEc*) d'une écurie de course (*codeEc, nomEc*) engage des pilotes (*codePi, nomPi*) pour courir sur les voitures de leur marque. Ce même propriétaire choisit, parmi ses pilotes, un pilote responsable chargé de faire le lien entre lui et l'ensemble des pilotes pour des problèmes particuliers.

Des groupes industriels (*codeGr, nomGr*) sponsorisent les écuries ou les pilotes en consacrant un certain budget publicitaire (*budget*) proportionnel à la taille du logo (*tailleLogo*) apposé sur la voiture (dans le cas d'un sponsoring d'écurie) ou sur le pilote (dans le cas d'un sponsoring de pilote). Comme la demande est forte, un même groupe industriel ne peut sponsoriser qu'une écurie ou qu'un pilote.

Parmi les pilotes, on distingue les pilotes d'essai et les pilotes titulaires, les pilotes titulaires pouvant occasionnellement faire le travail de pilotes d'essai. Les pilotes d'essai sont rémunérés avec un salaire horaire (*salaireHoraire*) alors que les pilotes titulaires ont un salaire fixe (*salaire*) et un montant de prime négocié et versé à chaque victoire de course (*prime*). Les pilotes d'essai effectuent des essais sur des circuits, à une certaine date (*dateEssai*) et avec un certain temps (*tempsEssai*). Chaque circuit (*codeCi, nomCi*) est situé dans une ville (*villeCi*) et un pays *(paysCi*). Les pilotes titulaires participent à des qualifications pour des courses (*codeCo, nomCo*) avec un certain résultat (*tempsQualif* et *placeQualif*). Grâce à ces qualifications, ils participent ensuite (sans condition de résultat) aux courses elles-mêmes (on conserve également le résultat de la course : *temps*, *place*). Il est impossible de participer à la course sans avoir participé aux qualifications pour cette course. Chaque circuit peut abriter plusieurs courses dans une saison à des dates différentes (*dateCo*).

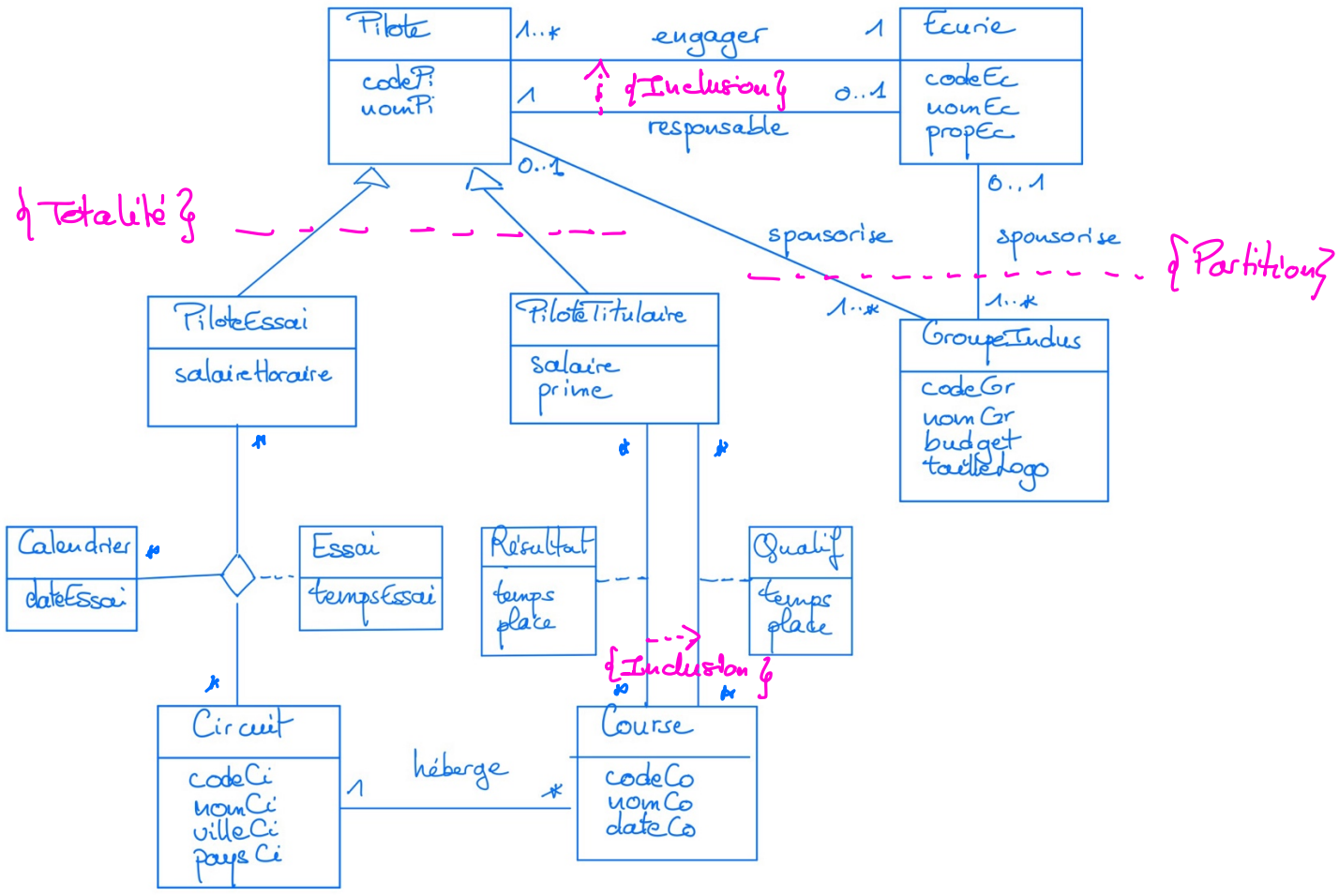
Q1 – A partir de la modélisation UML proposée en page 2 et du texte précédent, ajouter les contraintes nécessaires au diagramme de classes.

Q2 – Donner le modèle relationnel associé

Q3 – Donner le schéma physique associé

Diagramme de classe (schéma conceptuel) à compléter

Etape 1 - Niveau Conceptuel - Diagramme de classes avec ses contraintes



Etape 2 - Niveau logique : Schéma relationnel

On commence par regarder les différentes traductions de l'héritage pour choisir la meilleure possible.

① Traduction par distinction

Pilote (<u>codePi</u>, nomPi, #codeEc)
PiloteEssai (<u>#codePi</u>, salaireHoraire)
PiloteTitulaire (<u>#codePi</u>, salaire, prime)

Ecurie (<u>codeEc</u>, nomEc, propEc, #codePi)

traduction de Engager

traduction de responsable

[Dans cette traduction, les contraintes d'héritage ne sont pas prises en compte.]

② Traduction descendante (avec totalité)

PiloteEssai (<u>codePi</u>, nomPi, salaireHoraire, #codeEc)
PiloteTitulaire (<u>codePi</u>, nomPi, salaire, prime, #codeEc)
Ecurie (<u>codeEc</u>, nomEc, propEc, #codePi)

traduction de Engager

traduction de Responsable

Pb: Comment faire la référence vers la PK (Primary Key ?)
Il faudrait faire 2 références => impossible

Solution: On pourrait enlever la FK (Foreign Key) et la simuler avec un trigger qui vérifierait que la valeur existe bien dans l'une des 2 tables PiloteEssai / PiloteTitulaire

Autre solution possible: (..., #codePiEssai, #codePiTitulaire
Mais il faut vérifier que les 2 ne soient pas remplies et qu'au moins 1 est remplie
=> CHECK

③ Traduction ascendante

Pilote (codePi, nomPi, salaireHoraire, salaire, prime, #codeEc)

Ecurie (codeEc, nomEc, propEc, #codePi)

Dans cette traduction, les contraintes d'héritage ne sont pas prises en compte.

Pb ici :
Traduction compliquée ensuite de Qualif, Essai & Résultat, pour lesquelles il faudrait des triggers pour gérer les clés étrangères.

⇒ On garde la traduction par distinction, et on traduit maintenant le reste du schéma.

Pilote (coPi, nomPi, #codeEc)

PiloteEssai (#coPi, salaireHoraire)

PiloteTitulaire (#coPi, salaire, prime)

Ecurie (codeEc, nomEc, propEc, #codePi)

GroupeEtudes (codeGr, nomGr, budget, tailleEsp, #codePi, #codeEc)

Circuit (codeCi, nomCi, villeCi, paysCi)

Course (codeCo, nomCo, dateCo, #codeCi)

Qualif (#codePi, #codeCo, temps, place)

Résultat (#codePi, #codeCo, temps, place)

Essai (#CodePi, #codeCi, ~~#~~dateEssai, tempsEssai)

Calendrier (dateEssai)

Etape 3. Schéma physique => Contraintes sur les tables + triggers + procédures stockées

Pilote (codePi, nomPi, #codeEc)

↳ PK, FK, NOT NULL sur CodeEc

PiloteEssai (#codePi, salaireHoraire)

↳ PK, FK

PiloteTitulaire (#codePi, salaire, prime)

↳ PK, FK

+ Triggers & procédures pour gérer la totalité

Ecurie (codeEc, nomEc, respEc, #codePi)

↳ PK, FK, NOT NULL sur CodePi

- Gestion de l'inclusion entre Responsable et Engagé

Sol. 1 : Trigger sur Ecurie qui vérifie lors de l'insertion / mise à jour d'un couple (codeEc, codePi) que ce couple existe dans Pilote.

Sol 2 : FK(codeEc, codePi) de Ecurie qui référence Pilote(codeEc, codePi) + UNIQUE sur (codeEc, codePi) dans Pilote pour que la FK soit possible.

- "Un pilote ne peut être responsable que d'une seule écurie"
  => Il faut rajouter une contrainte UNIQUE (codePi)

GroupeIndus (codeGr, nomGr, budget, tailleLogo, #codePi, #codeEc)

↳ PK, 2 FK

+CHECK ((codePi IS NULL AND codeEc IS NOT NULL) OR (codePi is not null and codeEc is NULL))

Course (codeCo, nomCo, #codeCi)

↳ PK, FK, NOT NULL (codeCi)

Circuit (codeCi, nomCi, villeCi, paysCi)

↳ PK

Qualif (#codePi, #codeCo, tempsQualif, placeQualif)

↳ PK, & FK

Resultat (#codePi, #codeCo, temps, place)

↳ PK, 2 FK

Inclusion : FK (codePi, codeCo) de Résultat qui pointe vers PK (codePi, codeCo) de Qualif

Essai (#codePi, #codeCi, dateEssai, tempsEssai)

↳ PK, 2 FK

Que reste-il à traduire une fois les tables et leurs contraintes créés ?

Pilote (codePi, nomPi, #codeEc)
Essai (#codePi, salaireHoraire)
Titulaire (#codePi, salaire, prime)

Pour traduire l'héritage :
Contrainte B: il n'existe pas de Pilote, ni Essai ni Titulaire
Contrainte C: un pilote peut être essai & titulaire

Contrainte B:

. 2 procédures d'insertion :
→ insertionEssai : va insérer dans Pilote s'il n'existe pas déjà, puis insère dans PiloteEssai
→ insertionTitulaire : _____ PiloteTitulaire.

Pour forcer le passage par les procédures d'insertion pour peupler les tables, il faut enlever les droits d'insertion directe sur les tables Pilote, PiloteEssai et PiloteTitulaire et ne laisser que les droits d'exécution des procédures

. 2 procédures de suppression :
→ suppressionEssai : va supprimer dans PiloteEssai et s'il n'est pas dans piloteTitulaire, va supprimer dans Pilote
→ suppressionTitulaire : _____ PiloteTitulaire _____ piloteEssai _____

+ enlever les droits de suppression directe pour forcer le passage par les procédures de suppression

. 1 trigger de MAJ
Si on modifie CodePi dans Pilote, il faut répercuter la modification dans PiloteTitulaire & PiloteEssai

Contrainte C: ∅