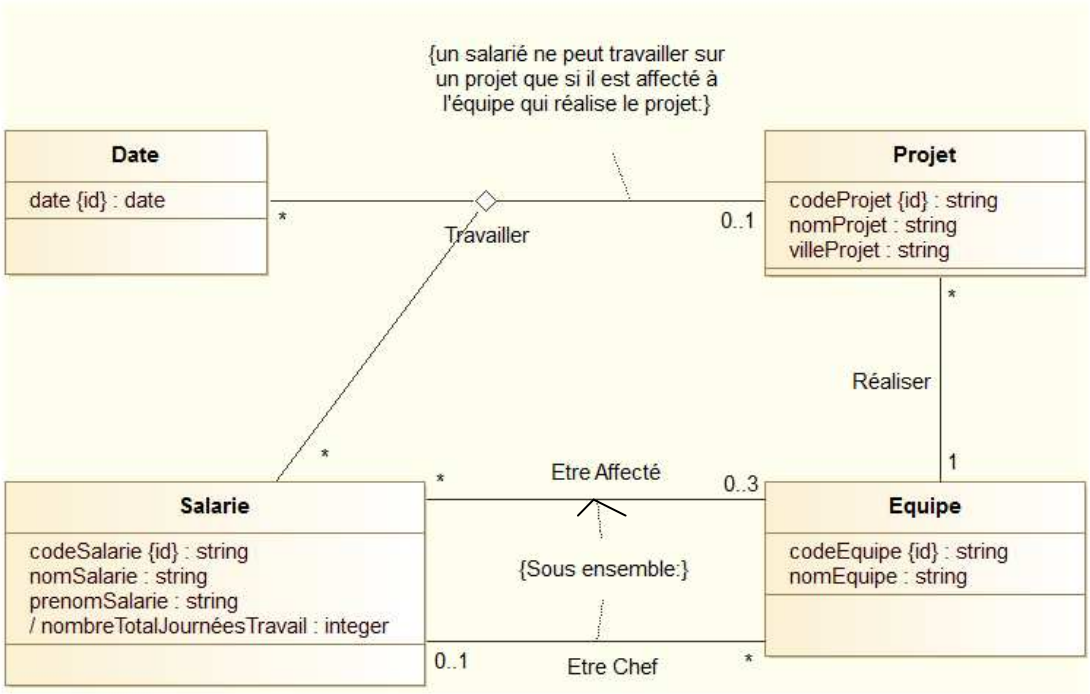
❾ UNIDEV

Dans cet exercice, on s'intéresse à l'ESN (Entreprise de Services du Numérique) UNIDEV™. Plus exactement, on souhaite implanter une base de données qui permettra de gérer les ressources humaines de cette entreprise afin de savoir à quelles équipes sont affectés les différents salariés et sur quels projets ils ont travaillé.

Lors de la phase d'analyse, on a obtenu le diagramme de classes conceptuel (ou diagramme des classes persistantes) suivant :



A partir de ce diagramme de classes, on a déduit le schéma relationnel suivant :

SALARIES (codeSalarie, nomSalarie, prenomSalarie, *nbTotalJourneesTravail*)
EQUIPES (codeEquipe, nomEquipe, codeSalarieChef#)
PROJETS (codeProjet, nomProjet, villeProjet, codeEquipe#)
ETREAFFECTE (codeSalarie#, codeEquipe#)
TRAVAILLER (codeSalarie#, dateTravail, codeProjet#)

Dans la relation *Salaries,* l'attribut `nbTotalJourneesTravail` (qui est indiqué en italique) est calculable à partir d'une requête SQL. Il engendre donc de la redondance dans la base de données, mais il a tout de même été maintenu car il permet d'accélérer certaines requêtes qui sont fréquemment exécutées.

1) Quelles sont les contraintes qui sont exprimées dans le diagramme de classes mais qui ont été perdues lors du passage vers le schéma relationnel ?
2) Pour chacune des contraintes que l'on a perdues, indiquer la ou les solutions qu'il faudrait programmer sous Oracle pour pouvoir la récupérer.

3) Aller sur le Moodle et copier le fichier « Unidev.sql » qui vous permettra de générer les tables de la base de données et d'insérer quelques tuples dans ces tables.

Première partie – Gestion des contraintes avec le SQL et les procédures stockées :

4) Mise à jour automatique d'un attribut calculé.
   - On souhaite que l'attribut `nbTotalJourneesTravail` de la table `Salaries` soit mis à jour automatiquement lorsque on insère une nouvelle ligne dans la table `Travailler` (pour simplifier, on ne gèrera pas encore les cas où on modifie ou supprime des lignes de cette table).

     Pour cela, au lieu de faire directement des insertions dans la table `Travailler`, on utilisera une procédure `AjouterJourneeTravail` que l'on vous demande d'écrire et qui doit avoir la signature suivante :

```
PROCEDURE AjouterJourneeTravail (
                      p_codeSalarie Travailler.codeSalarie%TYPE,
                        p_codeProjet Travailler.codeProjet%TYPE,
                      p_dateTravail Travailler.dateTravail%TYPE)
```

   - Résultat attendu :

```
CALL AjouterJourneeTravail('S2','P3', '10/01/2014');

SELECT nbTotalJourneesTravail
FROM Salaries
WHERE codeSalarie = 'S2';

NBTOTALJOURNEESTRAVAIL
----------------------
8
```

5) Gestion des multiplicités maximales des associations.
   - On souhaite maintenant programmer la contrainte qui empêche un salarié d'être affecté à plus de trois équipes lorsqu'on insère une nouvelle ligne dans la table `EtreAffecte` (pour simplifier on ne gèrera pas encore les cas où on modifie des lignes de cette table).

     Pour cela, au lieu de faire directement des insertions dans la table `EtreAffecte`, on utilisera une procédure `AffecterSalarieEquipe` que l'on vous demande d'écrire et dont la signature doit être la suivante :

```
PROCEDURE AffecterSalarieEquipe (
                     p_codeSalarie EtreAffecte.codeSalarie%TYPE,
                       p_codeEquipe EtreAffecte.codeEquipe%TYPE)
```

     Si le salarié dont le code est passé en paramètre n'a pas déjà été affecté à trois équipes, cette procédure l'affecte à l'équipe qui est passée en paramètre. Par contre, s'il est déjà affecté à trois équipes, cette procédure lève une exception avec l'instruction suivante :

```
RAISE_APPLICATION_ERROR(-20001, 'Le salarié est déjà affecté à
                                          au moins 3 équipes');
```

   - Résultat attendu :

```
CALL AffecterSalarieEquipe('S1', 'E3');

SELECT *
FROM EtreAffecte
WHERE codeSalarie = 'S1'
AND codeEquipe = 'E3';

CODESALARIE  CODEEQUIPE
------------ -----------
S1           E3

CALL AffecterSalarieEquipe('S8', 'E1');
Le salarié est déjà affecté à au moins 3 équipes


SELECT *
FROM EtreAffecte
WHERE codeSalarie = 'S8'
AND codeEquipe = 'E1';

aucune ligne sélectionnée
```

6) Contraintes d'inclusion entre deux associations.
   - On souhaite programmer la contrainte qui empêche un salarié d'être chef d'une équipe où il n'est pas affecté, notamment lorsqu'on modifie le chef d'une équipe qui existe déjà (mais il faudrait que cette contrainte soit aussi respectée quand on crée une nouvelle équipe ou lorsqu'on modifie des affectations existantes).
     Il se peut que la solution soit plus simple qu'elle n'y parait.
   - Résultat attendu :

     Si on indique que le salarié S3 est chef de l'équipe E4 il ne doit pas y avoir d'erreur. Ensuite, si on exécute la requête suivante, on obtiendra :

```
SELECT codeSalarieChef
FROM Equipes
WHERE codeEquipe = 'E4';

CODESALARIECHEF
---------------
S3
```

     Puis, si on indique que le salarié S4 est chef de l'équipe E3 il doit y avoir une **erreur** (car le salarié S4 n'est pas affecté à l'équipe E3). Ensuite, si on exécute la requête suivante, on obtiendra :

```
SELECT codeSalarieChef
FROM Equipes
WHERE codeEquipe = 'E3';

CODESALARIECHEF
---------------
S3
```

7) Contraintes d'inclusion concernant plus de deux associations.
   - On souhaite programmer la contrainte qui empêche un salarié de travailler sur un projet qui est réalisé par une équipe dans laquelle n'est pas affecté le salarié en question. On veut que cette contrainte soit vérifiée à chaque fois que l'on ajoute une journée de travail d'un salarié (pour le moment, on ne souhaite pas gérer le cas où on modifie une journée de travail déjà existante).
   - Résultat attendu :

     Si on indique que le salarié S2 a travaillé le 11/01/2014 sur le projet P3, il ne doit pas y avoir d'erreur. Ensuite, si on exécute la requête suivante, on obtiendra :

```
SELECT nbTotalJourneesTravail
FROM Salaries
WHERE codeSalarie = 'S2';

NBTOTALJOURNEESTRAVAIL
----------------------
9
```

     Puis, si on indique que le salarié S2 a travaillé le 12/01/2014 sur le projet P5, il doit y avoir une **erreur** (car le salarié S2 n'est pas affecté à l'équipe qui travaille sur le projet P5). Ensuite, si on exécute la requête suivante, on obtiendra :

```
SELECT nbTotalJourneesTravail
FROM Salaries
WHERE codeSalarie = 'S2';

NBTOTALJOURNEESTRAVAIL
----------------------
9
```

Deuxième partie – Gestion des contraintes avec des triggers :

8) Mise à jour automatique d'un attribut calculé avec un trigger lors d'une insertion.
   - On souhaite programmer un trigger afin que l'attribut `nbTotalJourneesTravail` de la table `Salaries` soit mis à jour automatiquement lorsque on insère une nouvelle ligne directement dans la table `Travailler` (pour commencer, on ne gèrera pas les modifications et les suppressions de lignes de cette table).
   - Résultat attendu :

```
INSERT INTO Travailler VALUES ('S1', '10/01/2014', 'P1');

SELECT nbTotalJourneesTravail
FROM Salaries
WHERE codeSalarie = 'S1';

NBTOTALJOURNEESTRAVAIL
----------------------
7
```

9) Gestion des multiplicités maximales des associations avec un trigger.
   - On souhaite programmer un trigger qui empêche un salarié d'être affecté à plus de trois équipes lorsqu'on insère une nouvelle ligne directement dans la table `EtreAffecte` (pour commencer on ne gèrera pas les modifications de lignes dans cette table).
   - Résultat attendu :

```
INSERT INTO EtreAffecte VALUES ('S2', 'E4');

SELECT *
FROM EtreAffecte
WHERE codeSalarie = 'S2'
AND codeEquipe = 'E4';

CODESALARIE  CODEEQUIPE
------------ -----------
S2           E4

INSERT INTO EtreAffecte VALUES ('S7', 'E4');
Le salarié est déjà affecté à au moins 3 équipes

SELECT *
FROM EtreAffecte
WHERE codeSalarie = 'S7'
AND codeEquipe = 'E4';

aucune ligne sélectionnée
```

10) Mise à jour des attributs calculés lors des modifications, insertions et suppressions.
   - On souhaite modifier le trigger de la question 8, afin que l'attribut `nbTotalJourneesTravail` de la table `Salaries` soit mis à jour automatiquement lorsque on insère une nouvelle ligne dans la table `Travailler`, mais aussi lorsqu'on modifie ou on supprime des lignes de cette table.
   - Résultat attendu :

```
UPDATE Travailler
SET codeSalarie = 'S5'
WHERE codeSalarie = 'S1'
AND dateTravail = '10/01/2014';

SELECT nbTotalJourneesTravail
FROM Salaries
WHERE codeSalarie = 'S1';

NBTOTALJOURNEESTRAVAIL
----------------------
6

SELECT nbTotalJourneesTravail
FROM Salaries
WHERE codeSalarie = 'S5';

NBTOTALJOURNEESTRAVAIL
----------------------
9

DELETE Travailler
WHERE codeSalarie = 'S5'
AND dateTravail = '10/01/2014';

SELECT nbTotalJourneesTravail
FROM Salaries
WHERE codeSalarie = 'S5';

NBTOTALJOURNEESTRAVAIL
----------------------
8
```

11) Trigger et contraintes d'inclusion concernant plus de deux associations.
   - On souhaite programmer un trigger qui empêche un salarié de travailler sur un projet qui est réalisé par une équipe dans laquelle n'est pas affecté le salarié en question. On veut que ce trigger se déclenche chaque fois que l'on ajoute une journée de travail d'un salarié (si vous avez du temps, vous pourrez ensuite programmer le trigger pour qu'il se déclenche aussi lorsqu'on modifie une journée de travail déjà existante).
   - Résultat attendu :

```
INSERT INTO Travailler VALUES ('S1', '11/01/2014', 'P1');

SELECT nbTotalJourneesTravail
FROM Salaries
WHERE codeSalarie = 'S1';

NBTOTALJOURNEESTRAVAIL
----------------------
7

INSERT INTO Travailler VALUES ('S1', '12/01/2014', 'P5');

Un salarié ne peut pas travailler sur un projet qui est réalisé
par une équipe dans laquelle il n'est pas affecté

SELECT nbTotalJourneesTravail
FROM Salaries
WHERE codeSalarie = 'S1';

NBTOTALJOURNEESTRAVAIL
----------------------
7
```

Troisième partie – Vues et Triggers :

12) Création d'une vue multitables.

- Créer une vue multitables *Affectations* qui contient toutes les affectations des salariés dans les différentes équipes ; avec pour chaque affectation, le code, le nom et le prénom du salarié ainsi que le code et le nom de l'équipe. Cette vue doit avoir la structure suivante :
  AFFECTATIONS (codeSalarie, nomSalarie, prenomSalarie, codeEquipe, nomEquipe)
- Vérifier que votre vue affiche bien ce qu'il faut. Puis essayer d'insérer des données à travers cette vue. Que se passe-t-il ? Pourquoi ?

```
INSERT INTO Affectations
VALUES ('S9', 'Zétofrais', 'Mélanie', 'E5', 'Indigo');
```

13) Trigger INSTEAD OF

- A l'aide d'un trigger INSTEAD OF, faites qu'il soit possible d'insérer des données à travers la vue multitables précédente. Si le salarié de l'affectation qui est inséré à travers la vue n'existe pas, il doit être créé dans la table *Salariés* (avec un nombre de journées de travail égal à 0). De même, si l'équipe de l'affectation qui est insérée à travers la vue n'existe pas, elle doit être elle aussi créée dans la table *Equipes* (avec un chef d'équipe égal à NULL). Dans tous les cas, une ligne doit être ajoutée dans la table *EtreAffecte*.
- Résultat attendu :

```
INSERT INTO Affectations
VALUES ('S9', 'Zétofrais', 'Mélanie', 'E5', 'Indigo');

INSERT INTO Affectations
VALUES ('S9', 'Zétofrais', 'Mélanie', 'E4', 'Mars');

INSERT INTO Affectations
VALUES ('S5', 'Umule', 'Jacques', 'E6', 'Europa');

INSERT INTO Affectations
VALUES ('S10', 'Zeblouse', 'Agathe', 'E7', 'Galileo');

SELECT *
FROM EtreAffecte

CODESALARIE  CODEEQUIPE
------------ -----------
…            …
S10          E7
…            …
S5           E6
…            …
S9           E4
S9           E5
```

- Modifier votre trigger INSTEAD OF pour qu'une vérification soit effectuée sur la cohérence des données insérées à travers votre vue. Ainsi, si le code du salarié passé en paramètre existe dans la table *Salariés* mais que son nom et son prénom ne correspondent pas à ce qu'il y a dans cette table, rien ne doit être inséré dans la base de données (ni salarié, ni équipe, ni affectation) et une exception est levée. De même, si le code de l'équipe passé en paramètre existe dans la table *Equipes* mais que son nom ne correspond pas à ce qu'il y a dans cette table, rien ne doit être inséré dans la base de données (ni salarié, ni équipe, ni affectation) et une exception est levée.
- Résultat attendu :

```
INSERT INTO Affectations
VALUES ('S9', 'Ouzy', 'Jacques', 'E6', 'Europa');
Les données sur le salarié S9 sont fausses

INSERT INTO Affectations
VALUES ('S9', 'Zétofrais', 'Mélanie', 'E6', 'Galileo');
Les données sur l'équipe E6 sont fausses
```

Quatrième partie pour les plus rapides – Requêtes SQL faciles :

R0 : le nom et le prénom des salariés qui ont travaillé le 4 janvier 2014 et le 8 janvier 2014 (ils doivent avoir travaillé à ces deux dates).

```
NOMSALARIE  PRENOMSALARIE
----------- --------------
Deuf        John
Umule       Jacques
```

R1 : le nom et le prénom des salariés qui partagent le même prénom qu'un autre salarié.

```
NOMSALARIE  PRENOMSALARIE
----------- --------------
Titouplin   Jean
Menage      Jean
```

R2 : le nombre de salariés affectés à chaque équipe.

```
NOMEQUIPE  NBSALARIES
---------- -----------
Luna       6
Galileo    1
Ganymède   3
Europa     1
Juno       5
Indigo     1
Mars       6
```

R3 : le nom de l'équipe qui possède le plus de salariés affectés.

```
NOMEQUIPE
----------
Luna
Mars
```

R4 : le nom et le prénom des salariés qui sont affectés dans tous les projets où le salarié Jean Menage est affecté.

```
NOMSALARIE  PRENOMSALARIE
----------- --------------
Gator       Ali
Stiké       Sophie
```

R5 : pour chacune des villes où il y a un ou des projets réalisés, indiquer le nom du projet qui possède le plus de salariés affectés à son équipe.

```
VILLEPROJET  NOMPROJET
------------ --------------------------
Montpellier  Logiciel Sécurité Sociale
Béziers      Outil CRM de la Poste
Nimes        Site du club de billes
```