

Livrable 1

Rapport de BD

VACQUIER Hugo, BIRET-TOSCANO Esteban, FERNANDEZ Mickael
21 mars 2023



Sommaire :

I - Schéma relationnel de l'existant :	2
II - Normalisation de la base de données :	4
III - Nouveau schéma relationnel :	13
IV - Nouveau diagramme de classe :	15
V - Nouveau script de la base de données :	16

I - Schéma relationnel de l'existant :

Commandes permettant de visualiser le dictionnaire de données Oracle de la base de données existante :

```
SELECT TABLE_NAME FROM USER_TABLES;
```

```
SELECT      TABLE_NAME,      COLUMN_NAME,      DATA_TYPE,      DATA_LENGTH,      DATA_PRECISION FROM USER_TAB_COLS;
```

```
SELECT * FROM USER_CONSTRAINTS; //car l'identificateur "C" semble incorrect, on réalise une sélection de l'ensemble des attributs disponibles sur cette table
```

```
SELECT TABLE_NAME, CONSTRAINT_NAME, COLUMN_NAME, POSITION FROM USER_CONS_COLUMNS;
```

Schéma relationnel (clés primaires soulignées, clés étrangères suffixées de '#', autres contraintes plus bas) :

COMPOSER (nm#, nj, prej#, nomj#, maillot),

ÉQUIPE (ne, nome, pres, noms, dtns, nats),

JOUEUR (nj, prej, nomj, pst, cap, ne#),

MARQUER (nm#, tps, nj, action, point),

MATCH(nm, jrn, dtm, stade, ville, na, prea, noma, dtna, nata, ne1#, ne2#, scr1, scr2),

SURVENIR (nm#, tps, nj#, evt),

Contraintes uniques :

- EQUIPE.ne,
- EQUIPE.nome,
- JOUEUR.nj,
- MATCH.ne1,
- MATCH.ne2.

Contraintes checks :

- COMPOSER.maillot BETWEEN 1 and 23
- JOUEUR.cap = 'c',
- JOUEUR.pst IN ('Pilier', 'Talonneur', 'Deuxième ligne', 'Troisième ligne aile', 'Troisième ligne centre', 'Demi de mêlée', 'Demi d'ouverture', 'Centre', 'Ailier', 'Arrière'),
- MARQUER CHECK (point = 7 AND action = 'Essais de Pénalité'

OR point = 5 AND action = 'Essais'

OR point = 2 AND action = 'Transformation'

OR point = 3 AND (action = 'Pénalité' OR action = 'Coup de pied tombé'),
- SURVENIR.evt IN ('Carton Jaune', 'Carton Rouge', 'Entrée', 'Sortie'),
- MATCH.ne1, MATCH.ne2 CHECK (ne1 <> ne2),
- MATCH.scr1 >= 0,
- MATCH.scr2 >= 0,
- MATCH.jrn BETWEEN 1 and 5.

Contraintes NOT NULL :

- JOUEUR.ne IS NOT NULL,
- MATCH.ne1 IS NOT NULL,
- MATCH.ne2 IS NOT NULL.

II - Normalisation de la base de données :

Liste des dépendances fonctionnelles :

df1 : ne → nome, noms, pres, dtns, nats

df2 : nome → ne, noms, pres, dtns, nats

df3 : noms, pres → dtns, nats

df4 : nomj, prej → nj, pst, cap, ne

df5 : nj → nomj, prej, pst, cap, ne

df6 : nm → jrn, dtm, stade, ville, na, prea, noma, dtna, nata, ne1, ne2, scr1, scr2

df7 : dtm, ne1, ne2 → nm

df8 : ne1, ne2 → scr1, scr2, nm, jrn, dtm, stade, ville, na, prea, noma, dtna, nata

df9 : stade → ville

df10 : noma, prea → na, dtna, nata

df11 : na → noma, prea

df12 : nm, nj → maillot

df13 : nm, nomj, prej → maillot

df14 : nm, tps, nj → action, point, evt

df15 : nm, tps → point, action

A partir de ces df, nous allons évaluer la qualité de la BD existante.

Pour ce faire, nous allons vérifier dans un premier temps si l'ensemble des dépendances fonctionnelles sont élémentaires. La première étape est d'observer les différentes dépendances fonctionnelles possédant plusieurs attributs à gauche (les df ne contenant qu'un seul attribut dans la partie gauche sont d'office élémentaires) : ici, il s'agit des dépendances fonctionnelles *df3*, *df4*, *df7*, *df8*, *df10*, *df12*, *df13*, *df14* et *df15*.

Ces dépendances ne paraissent pas élémentaires et c'est pourquoi nous allons étudier les différentes fermetures transitives, en fonction de la couverture minimale, des attributs situés à gauche des dépendances fonctionnelles :

Pour $df3$: noms, pres \rightarrow dtns, nats

On va donc calculer les fermetures transitives de noms et pres :

$(\text{noms})^+ = \{\text{noms}\}$

$(\text{pres})^+ = \{\text{pres}\}$

On voit que dtns et nats n'appartiennent pas à $(\text{noms})^+$ ou $(\text{pres})^+$.

On peut donc en déduire que **$df3$ est élémentaire.**

Pour $df4$: nomj, prej \rightarrow nj, pst, cap, ne

On va donc calculer les fermetures transitives de nomj et prej :

$(\text{nomj})^+ = \{\text{nomj}\}$

$(\text{prej})^+ = \{\text{prej}\}$

On voit que nj, pst, cap et ne n'appartiennent pas à $(\text{nomj})^+$ ou $(\text{prej})^+$.

On peut donc en déduire que **$df4$ est élémentaire.**

Pour $df7$: dtm, ne1, ne2 \rightarrow nm

On va donc calculer les fermetures transitives de dtm, ne1 et ne2 :

$(\text{dtm})^+ = \{\text{dtm}\}$

$(\text{ne1})^+ = \{\text{ne1}\}$

$(\text{ne2})^+ = \{\text{ne2}\}$

$(\text{dtm}, \text{ne1})^+ = \{\text{dtm}, \text{ne1}\}$

$(\text{dtm}, \text{ne2})^+ = \{\text{dtm}, \text{ne2}\}$

$(\text{ne1}, \text{ne2})^+ = \{\text{ne1}, \text{ne2}, \text{src1}, \text{scr2}, \text{nm}, \text{jrn}, \text{dtm}, \text{stade}, \text{ville}, \text{na}, \text{prea}, \text{noma}, \text{dtna}, \text{nata}\}$

On voit que nm appartient à $(ne1, ne2)^+$, qui est plus petit que $(dtm, ne1, ne2)^+$.

On peut donc supprimer la df7.

Pour $df8$: $ne1, ne2 \rightarrow scr1, scr2, nm, jrn, dtm, stade, ville, na, prea, noma, dtna, nata$

On va donc calculer les fermetures transitives de $ne1$ et $ne2$:

$$(ne1)^+ = \{ne1\}$$

$$(ne2)^+ = \{ne2\}$$

On voit que $scr1, scr2, nm, jrn, dtm, stade, ville, na, prea, noma, dtna$ et $nata$ n'appartiennent pas à $(ne1)^+$ ou $(ne2)^+$.

On peut donc en déduire que **$df8$ est élémentaire.**

Pour $df10$: $noma, prea \rightarrow na, dtna, nata$

$$(noma)^+ = \{noma\}$$

$$(prea)^+ = \{prea\}$$

Ainsi, $na, dtna$ et $nata \notin (noma)^+$ ou $\notin (prea)^+$, **donc $df10$ est élémentaire.**

Pour $df12$: $nm, nj \rightarrow maillot$

$$(nm)^+ = \{nm, jrn, dtm, stade, ville, na, prea, noma, dtna, nata, ne1, ne2, scr1, scr2\}$$

$$(nj)^+ = \{nj, nomj, prej, pst, cap, ne, nome, noms, pres, dtns, nats\}$$

Ainsi $maillot \notin (nm)^+$ ou $\notin (nj)^+$, **donc $df12$ est élémentaire.**

Pour $df13$: $nm, nomj, prej \rightarrow maillot$

A partir des fermetures transitives déjà calculées pour les dépendances fonctionnelles précédentes, on peut en conclure que l'attribut $maillot \notin (nm)^+$ ou $\notin (nomj)^+$ ou $\notin (prej)^+$.

$$(nm)^+ = \{nm, jrn, dtm, stade, ville, na, prea, noma, dtna, nata, ne1, ne2, scr1, scr2\}$$

$$(\text{nomj})^+ = \{\text{nomj}\}$$
$$(\text{prej})^+ = \{\text{prej}\}$$
$$(\text{nm}, \text{nomj})^+ = \{\text{nm}, \text{nomj}\}$$
$$(\text{nm}, \text{prej})^+ = \{\text{nm}, \text{prej}\}$$
$$\{\text{nomj}, \text{pre}\}^+ = \{\text{nomj}, \text{prej}, \text{nj}, \text{pst}, \text{cap}, \text{ne}\}$$

On voit donc que maillot n'appartient à aucun de ces ensembles, donc **df13 est élémentaire**.

Pour **df14** : $\text{nm}, \text{tps}, \text{nj} \rightarrow \text{action}, \text{point}, \text{evt}$

$$(\text{nm})^+ = \{\text{jrn}, \text{dtm}, \text{stade}, \text{ville}, \text{na}, \text{prea}, \text{noma}, \text{dtna}, \text{nata}, \text{ne1}, \text{ne2}, \text{scr1}, \text{scr2}\}$$
$$(\text{tps})^+ = \{\text{tps}\}$$
$$(\text{nm}, \text{tps})^+ = \{\text{nm}, \text{tps}, \text{point}, \text{action}\}$$
$$(\text{tps}, \text{nj})^+ = \{\text{tps}, \text{nj}\}$$
$$(\text{nm}, \text{nj})^+ = \{\text{nm}, \text{nj}, \text{maillot}\}$$

Ainsi, on voit que action et point appartiennent à $(\text{nm}, \text{tps})^+$, qui est plus petit que $(\text{nm}, \text{tps}, \text{nj})^+$.

On peut donc supprimer action et point de la partie droite de cette df

df14' : $\text{nm}, \text{tps}, \text{nj} \rightarrow \text{evt}$

Pour **df15** : $\text{nm}, \text{tps} \rightarrow \text{point}, \text{action}$

$$(\text{nm})^+ = \{\text{nm}, \text{jrn}, \text{dtm}, \text{stade}, \text{ville}, \text{na}, \text{prea}, \text{noma}, \text{dtna}, \text{nata}, \text{ne1}, \text{ne2}, \text{scr1}, \text{scr2}\}$$
$$(\text{tps})^+ = \{\text{tps}\}$$

On voit que les attributs point et action $\notin (\text{nm})^+$ ou $\notin (\text{tps})^+$. **df15 est également élémentaire**.

La base de données initiale est donc en 1NF, car certaines df ne sont pas élémentaires.

Les nouvelles dépendances fonctionnelles (toutes élémentaires) sont les suivantes :

$df1 : ne \rightarrow nome, noms, pres, dtns, nats$

$df2 : nome \rightarrow ne, noms, pres, dtns, nats$

$df3 : noms, pres \rightarrow dtns, nats$

$df4 : nomj, prej \rightarrow nj, pst, cap, ne$

$df5 : nj \rightarrow nomj, prej, pst, cap, ne$

$df6 : nm \rightarrow jrn, dtm, stade, ville, na, prea, noma, dtna, nata, ne1, ne2, scr1, scr2$

$df8 : ne1, ne2 \rightarrow scr1, scr2, nm, jrn, dtm, stade, ville, na, prea, noma, dtna, nata$

$df9 : stade \rightarrow ville$

$df10 : noma, prea \rightarrow na, dtna, nata$

$df11 : na \rightarrow noma, prea$

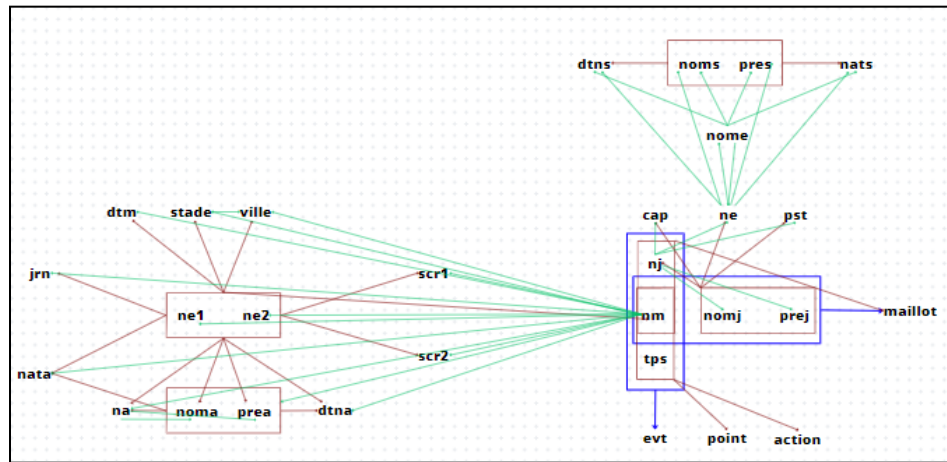
$df12 : nm, nj \rightarrow maillot$

$df13 : nm, nomj, prej \rightarrow maillot$

$df14' : nm, tps, nj \rightarrow evt$

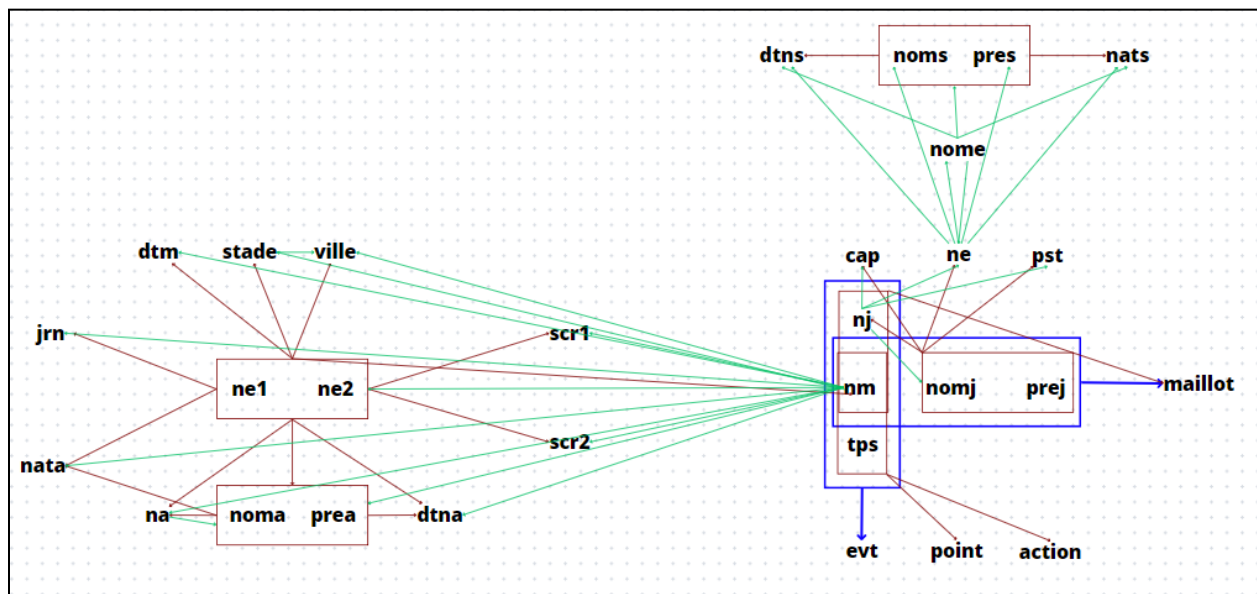
$df15 : nm, tps \rightarrow point, action$

Il nous faut maintenant vérifier que toutes les df soient directes, nous allons construire un graphe pour y voir plus clair :



Un attribut simple qui détermine l'ensemble des attributs d'un groupe équivaut à ce que cet attribut détermine directement le groupe.

Avant de traiter les cas complexes, voyons à quoi le nouveau graphe temporaire ressemble :



Traitons les cas complexes :

- $df5$: $nj \rightarrow nomj, prej, pst, cap, ne$; or $df4$: $nomj, prej \rightarrow nj, pst, cap, ne$.

Ici, nous rencontrons un circuit entre le groupe “nomj, prej” et l'attribut “nj”. Pour des raisons de duplications et de gestion de stockage, nous privilégierons (dans la mesure du possible), le choix d'un attribut simple comme clé primaire : ici nj. Ainsi, toutes les df sont transformées en utilisant la clé en partie gauche :

{ nj, (nomj, prej) }_{4,5} → pst, cap, ne

A noter qu'il ne faut pas oublier à remplacer dans toutes les autres *df*, les attributs du circuit par la clé choisie :

{ nj → nomj, prej, pst, cap, ne ; nomj, prej → nj, pst, cap, ne } = { nj → nomj, prej, pst, cap, ne ; ~~nj → nj, pst, cap, ne~~ } = **{ nj → nomj, prej, pst, cap, ne }**

- *df6* : nm → jrn, dtm, stade, ville, na, prea, noma, dtna, nata, ne1, ne2, scr1, scr2 ; or *df8* : ne1, ne2 → scr1, scr2, nm, jrn, dtm, stade, ville, na, prea, noma, dtna, nata.

De même, nous rencontrons un circuit entre le groupe "ne1, ne2" et l'attribut "nm". Comme déjà évoqué, nous allons choisir un attribut simple comme clé primaire : ici nm. Nous avons alors :

{ nm, (ne1, ne2) }_{6,8} → jrn, dtm, stade, ville, na, prea, noma, dtna, nata, scr1, scr2

Ainsi, remplaçons dans toutes les autres *df*, les attributs du circuit par la clé choisie :

{ nm → jrn, dtm, stade, ville, na, prea, noma, dtna, nata, ne1, ne2, scr1, scr2 ; ne1, ne2 → scr1, scr2, nm, jrn, dtm, stade, ville, na, prea, noma, dtna, nata }
= { nm → jrn, dtm, stade, ville, na, prea, noma, dtna, nata, ne1, ne2, scr1, scr2 ; ~~nm → scr1, scr2, nm, jrn, dtm, stade, ville, na, prea, noma, dtna, nata~~ }
= **{ nm → jrn, dtm, stade, ville, na, prea, noma, dtna, nata, ne1, ne2, scr1, scr2 }**

- *df10* : noma, prea → na, dtna, nata ; or *df11* : na → noma, prea.

Nous avons : **{ na, (noma, prea) }_{10,11} → dtna, nata**

Par la suite : { noma, prea → na, dtna, nata ; na → noma, prea }
= { ~~na → na, dtna, nata~~ ; na → noma, prea } = **{ na → noma, prea, dtna, nata }**

Enfin, un cas plus simple :

- *df1* : ne → nome, noms, pres, dtns, nats ; or *df2* : nome → ne, noms, pres, dtns, nats.

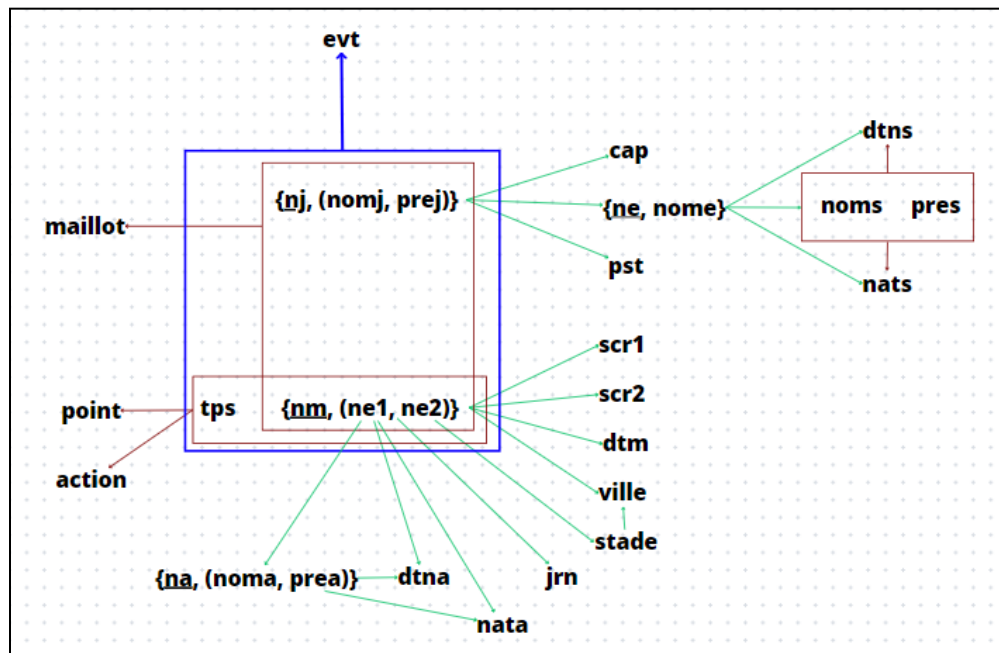
Cette fois-ci, nous avons un circuit entre deux attributs simples : nome et ne. Les deux attributs étant équivalents, le choix s'avère être arbitraire. Ici, nous ferons le choix d'utiliser "ne" comme étant la clé primaire. Nous avons alors : **{ ne, nome }_{1,2} → noms, pres, dtns, nats**

Ainsi : { ne → nome, noms, pres, dtns, nats ; nome → ne, noms, pres, dtns, nats }

= { ne → nome, noms, pres, dtns, nats ; ~~ne → ne, noms, pres, dtns, nats~~ }

= { **ne → nome, noms, pres, dtns, nats** }

Regardons désormais notre nouveau schéma des dépendances fonctionnelles, avec considération des attributs équivalents :



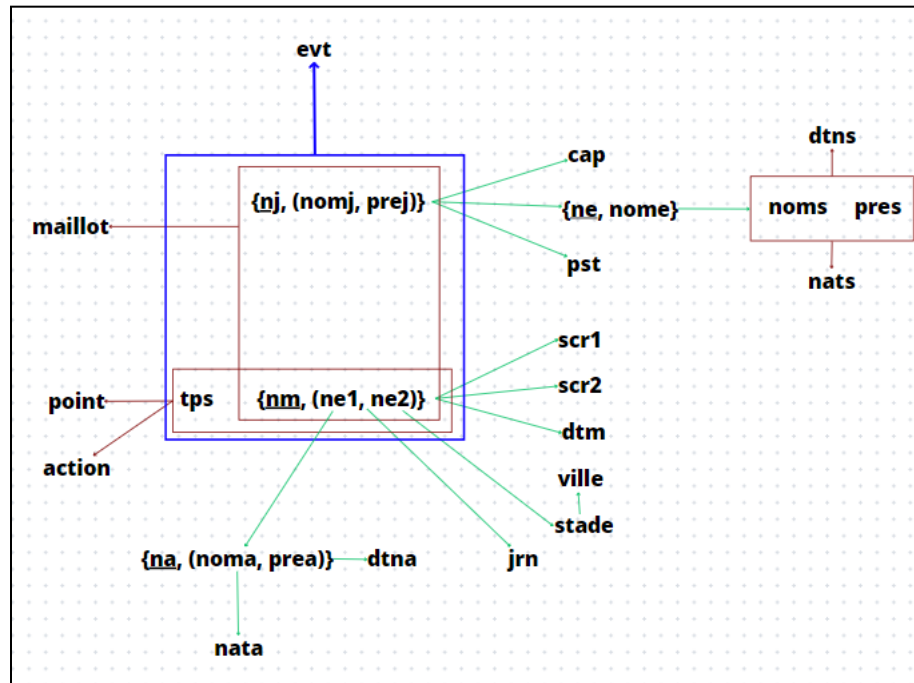
Désormais, bien que de nombreuses dépendances fonctionnelles s'avèrent être transitives, c'est-à-dire le fait de trouver un chemin autre qu'un chemin direct entre deux attributs, celles-ci ne seront pas difficiles à traiter. Ici, nous avons :

- $df1'$: ne → nome, noms, pres, **dtns, nats** ; or $df3$: noms, pres → dtns, nats. Il faut retirer les $df1d'$ et $df1e'$ ce qui donne alors : **$df1'$: ne → nome, noms, pres.**

- $df6'$: nm → jrn, dtm, **stade, ville**, na, prea, noma, dtna, nata, ne1, ne2, scr1, scr2 ; or $df9$: stade → ville. Il faut retirer la $df6d'$ ce qui donne : **$df6'$: nm → jrn, dtm, stade, na, dtna, nata, ne1, ne2, scr1, scr2.**

- $df6'$: $nm \rightarrow jrn, dtm, stade, na, prea, noma, \mathbf{dtna}, \mathbf{nata}, ne1, ne2, scr1, scr2$; or
 $df11'$: $na \rightarrow noma, prea, dtna, nata$. Il faut retirer les $df6g'$ et $df6h'$ ce qui donne :
 $df6'$: $nm \rightarrow jrn, dtm, stade, na, ne1, ne2, scr1, scr2$.

Notre couverture minimale ressemble désormais à la suivante :



Et voici donc la couverture minimale sous forme de df :

- $df1'$: $ne \rightarrow nome, noms, pres$
- $df3$: $noms, pres \rightarrow dtns, nats$
- $df4'$: $nj \rightarrow nomj, prej, pst, cap, ne$
- $df6'$: $nm \rightarrow jrn, dtm, stade, na, ne1, ne2, scr1, scr2$
- $df9$: $stade \rightarrow ville$
- $df11'$: $na \rightarrow noma, prea, dtna, nata$
- $df12$: $nm, nj \rightarrow maillot$

- **df14'** : nm, tps, nj → evt
- **df15** : nm, tps → point, action

III - Nouveau schéma relationnel :

Nous utilisons ici la méthode par synthèse :

Étape 1 : On fait des paquets de df, en regroupant celles ayant exactement la même partie gauche :

- **Hne** = {ne → nome, noms, pres}
- **Hnoms, pres** = {noms, pres → dtns, nats}
- **Hnj** = {nj → nomj, prej, pst, cap, ne}
- **Hnm** = {nm → jrn, dtm, stade, na, ne1, ne2, scr1, scr2}
- **Hstade** = {stade → ville}
- **Hna** = {na → noma, prea, dtna, nata}
- **Hnm, nj** = {nm, nj → maillot}
- **Hnm, tps, nj** = {nm, tps, nj → evt}
- **Hnm, tps** = {nm, tps → point, action}

Étape 2 : Traitement des circuits :

Nous n'avons pas de circuit ici, ils ont été traités lors de la suppression des df transitives.

Étape 3 : Création d'une relation pour chaque groupe :

Les clés primaires sont les attributs de la partie gauche de la df. Pour déterminer les clés étrangères, on regarde si la clé primaire de chaque relation est présente dans d'autres relations.

Voici donc le schéma relationnel de la nouvelle BD en 3NF :

R1 (ne, nome, (noms, pres)#) → Table Équipe

R2 (noms, pres, dtns, nats) → Table Sélectionneur

R3 (nj, nomj, prej, pst, cap, ne#) → Table Joueur

R4 (nm, jrn, dtm, stade#, na#, ne1, ne2, scr1, scr2) → Table Match

R5 (stade, ville) → Table Stade

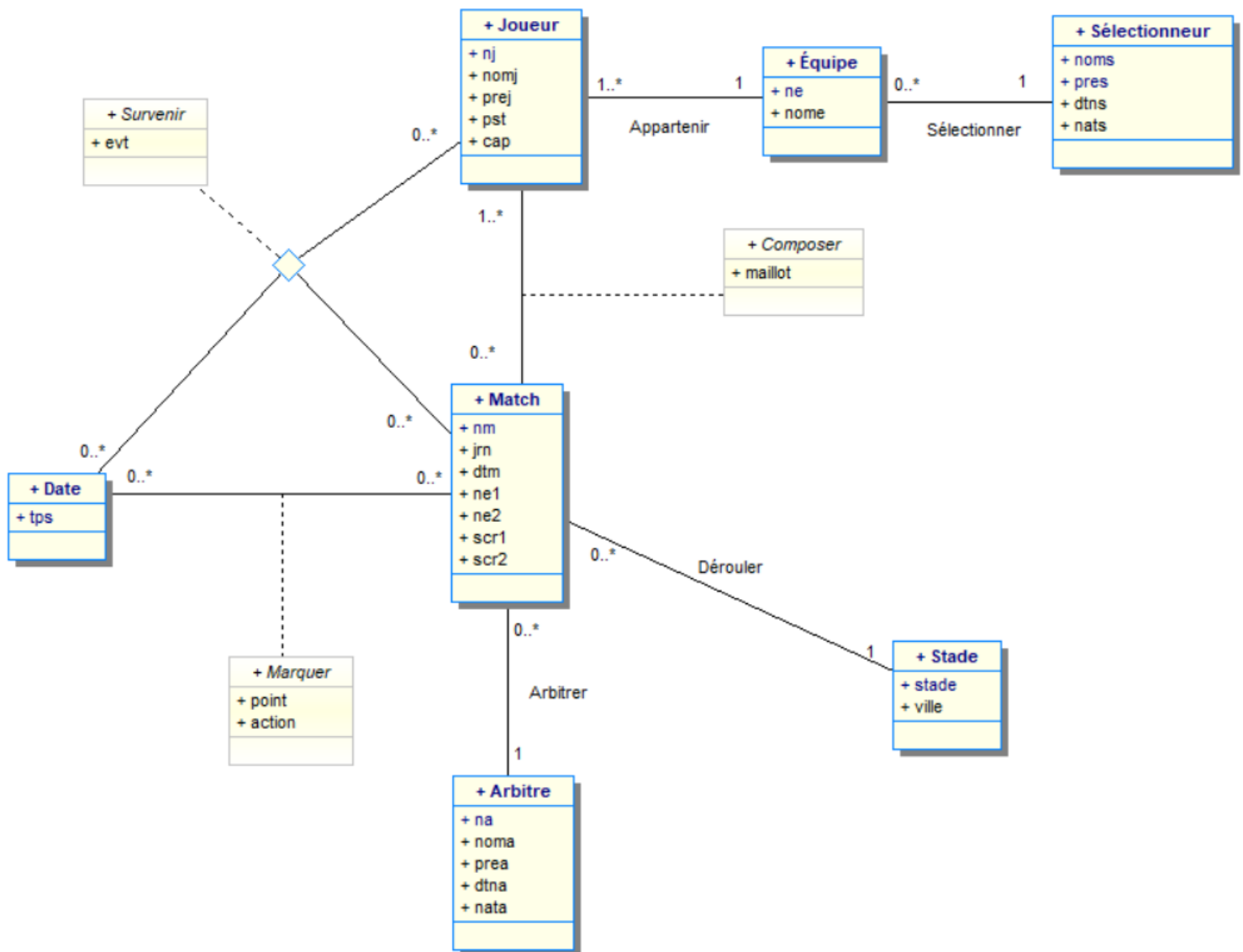
R6 (na, noma, prea, dtna, nata) → Table Arbitre

R7 (nm#, nj#, maillot) → Table Composer

R8 (nm#, nj#, tps, evt) → Table Survenir

R9 (nm#, tps, point, action) → Table Marquer

IV - Nouveau diagramme de classe :




Date (tps) → (ne pas implémenter sous Oracle, uniquement pour le conceptuel).

V - Nouveau script de la base de données :

Voici le script de création des 9 tables, ainsi que toutes les contraintes (clés primaires, étrangères, check, not null et unique) :

```
CREATE TABLE SELECTIONNEURV2 (  
    noms VARCHAR2(30 BYTE),  
    pres VARCHAR2(30 BYTE),  
    dtns DATE,  
    nats VARCHAR2(30 BYTE),  
    CONSTRAINT PK_SELECTIONNEURV2 PRIMARY KEY (noms, pres)  
);
```


```
CREATE TABLE EQUIPEV2 (  
    ne CHAR(3 BYTE),  
    nome VARCHAR2(50 BYTE),  
    noms VARCHAR2(30 BYTE),  
    pres VARCHAR2(30 BYTE),  
    CONSTRAINT PK_EQUIPEV2 PRIMARY KEY (ne) ENABLE,  
    CONSTRAINT UK_EQUIPEV2_NE_NOME UNIQUE (ne, nome),  
    CONSTRAINT FK_EQUIPEV2_SELECTIONNEURV2 FOREIGN KEY (noms, pres)  
    REFERENCES SELECTIONNEURV2 (noms, pres)  
);
```



```
CREATE TABLE JOUEURV2 (  
    nj NUMBER(5,0),  
    nomj VARCHAR2(30 BYTE),  
    prej VARCHAR2(30 BYTE),  
    pst VARCHAR2(30 BYTE),  
    cap CHAR(1 BYTE),  
    ne CHAR(3 BYTE),  
  
    CONSTRAINT PK_JOUEURV2 PRIMARY KEY (nj),  
  
    CONSTRAINT NN_JOUEURV2_NE CHECK (ne IS NOT NULL),  
  
    CONSTRAINT CK_JOUEURV2_PST CHECK (pst IN ('Pilier', 'Talonneur', 'Deuxième ligne',  
'Troisième ligne aile', 'Troisième ligne centre', 'Demi de mêlée', 'Demi d'ouverture', 'Centre', 'Ailier',  
'Arrière'))),  
  
    CONSTRAINT CK_JOUEURV2_CAP CHECK (cap = 'C'),  
  
    CONSTRAINT FK_JOUEURV2_EQUIPEV2 FOREIGN KEY (ne) REFERENCES EQUIPEV2 (ne)  
);
```

```
CREATE TABLE ARBITREV2 (  
    na NUMBER(5,0),  
    noma VARCHAR2(30 BYTE),  
    prea VARCHAR2(30 BYTE),  
    dtna DATE,  
    nata VARCHAR2(30 BYTE),  
  
    CONSTRAINT PK_ARBITREV2 PRIMARY KEY (na)  
);
```

```
CREATE TABLE STADEV2 (
```



```
stade VARCHAR2(30 BYTE),  
ville VARCHAR2(30 BYTE),  
CONSTRAINT PK_STADEV2 PRIMARY KEY (stade)  
);
```

```
CREATE TABLE MATCHV2 (  
    nm NUMBER(5,0),  
    jrn NUMBER(1,0),  
    dtm DATE,  
    stade VARCHAR2(30 BYTE),  
    na NUMBER(5,0),  
    ne1 CHAR(3 BYTE),  
    ne2 CHAR(3 BYTE),  
    scr1 NUMBER(3,0),  
    scr2 NUMBER(3,0),  
    CONSTRAINT PK_MATCHV2 PRIMARY KEY (nm) ENABLE,  
    CONSTRAINT FK_MATCHV2_STADEV2 FOREIGN KEY (stade) REFERENCES STADEV2 (stade),  
    CONSTRAINT FK_MATCHV2_ARBITREV2 FOREIGN KEY (na) REFERENCES ARBITREV2 (na),  
    CONSTRAINT UK_MATCHV2_NE1_NE2 UNIQUE (ne1, ne2),  
    CONSTRAINT NN_MATCHV2_NE1 CHECK (ne1 IS NOT NULL),  
    CONSTRAINT NN_MATCHV2_NE2 CHECK (ne2 IS NOT NULL),  
    CONSTRAINT CK_MATCHV2_NE1_NE2 CHECK (ne1 <> ne2),  
    CONSTRAINT CK_MATCHV2_SCR1 CHECK (scr1 >= 0),  
    CONSTRAINT CK_MATCHV2_SCR2 CHECK (scr2 >= 0),
```

CONSTRAINT CK_MATCHV2_JRN CHECK (jrn BETWEEN 1 AND 5)
);

CREATE TABLE COMPOSERV2 (
 nm NUMBER(5,0),
 nj NUMBER(5,0),
 maillot NUMBER(2,0),
 CONSTRAINT PK_COMPOSERV2 PRIMARY KEY (nm, nj),
 CONSTRAINT FK_COMPOSERV2_MATCHV2 FOREIGN KEY (nm) REFERENCES MATCHV2
 (nm),
 CONSTRAINT FK_COMPOSERV2_JOUEURV2 FOREIGN KEY (nj) REFERENCES JOUEURV2
 (nj),
 CONSTRAINT CK_COMPOSERV2_MAILLOT CHECK (maillot BETWEEN 1 AND 23)
);

CREATE TABLE SURVENIRV2 (
 nm NUMBER(5,0),
 nj NUMBER(5,0),
 tps VARCHAR2(5 BYTE),
 evt VARCHAR2(30 BYTE),
 CONSTRAINT PK_SURVENIRV2 PRIMARY KEY (nm, nj, tps),
 CONSTRAINT FK_SURVENIRV2_MATCHV2 FOREIGN KEY (nm) REFERENCES MATCHV2 (nm),
 CONSTRAINT FK_SURVENIRV2_JOUEURV2 FOREIGN KEY (nj) REFERENCES JOUEURV2 (nj),
 CONSTRAINT CK_SURVENIRV2_EVT CHECK (evt IN ('Carton Jaune','Carton
Rouge','Entrée','Sortie'))

);

```
CREATE TABLE MARQUERV2 (  
    nm NUMBER(5,0),  
    tps VARCHAR2(5 BYTE),  
    point NUMBER(1,0),  
    action VARCHAR2(30 BYTE),  
    CONSTRAINT PK_MARQUERV2 PRIMARY KEY (nm, tps),  
    CONSTRAINT FK_MARQUERV2_MATCHV2 FOREIGN KEY (nm) REFERENCES MATCHV2 (nm),  
    CONSTRAINT CK_MARQUERV2_POINT_ACTION CHECK (point = 7 AND action = 'Essais de  
Pénalité' OR point = 5 AND action = 'Essais' OR point = 2 AND action = 'Transformation' OR point  
= 3 AND (action = 'Pénalité' OR action = 'Coup de pied tombé'))  
);
```

Création des 9 tables avec leurs contraintes sous sqldeveloper :

Table SELECTIONNEURV2 créé(e).

Table ARBITREV2 créé(e).

Table COMPOSERV2 créé(e).

Table EQUIPEV2 créé(e).

Table STADEV2 créé(e).

Table SURVENIRV2 créé(e).

Table JOUEURV2 créé(e).

Table MATCHV2 créé(e).

Table MARQUERV2 créé(e).