ÉCOLE NATIONALE D'INGÉNIEURS DE BRST

UCI UNION CYCLISTE INTERNATIONALE

ENIB Semester S3P
Computer Science - Relational Databases

Union Cycliste Internationale: Managing Riders in Different Teams

---

Florian Pasco S3P-A

Version 2.0 - May 10, 2022

Contents

List of Figures

1 Initial Specifications

The Union Cycliste Internationale has requested our IT services to develop an Information System to manage the riders belonging to the various teams of the federation based on the following specifications:

"Riders are individuals for whom we record their first and last name, height, date of birth, and the team they belong to. A team is identified by its name, has a budget, and a sports director who is a person with a known first and last name and date of birth. Teams are funded by sponsors, which may vary each year, and for each sponsor, we record their name, address, and field of activity.

A race corresponds to a race name (e.g., 'Tour de France'), and we know the total distance to be covered. It can include one or more stages, each identified by its sequence number (e.g., 'Stage 3'), date, type (e.g., 'Individual Time Trial'), starting city, and finishing city.

For each rider who participated in a stage of a race, we record the ranking they achieved in that stage. For each race, we track the final winner and the team they belong to.

For every race, teams employ soigneurs, who are individuals for whom we store their first and last name, date of birth, and nationality. Additionally, at each stage, we record which doses of which product(s) a soigneur administered to a rider.

A product is identified by a product number, has a name, an indication (e.g., 'muscle pain'), a contraindication (e.g., 'do not administer to individuals under 20 years old'), and a dosage recommendation (e.g., '1 tablet per day').

In this production database, only current information (related to the ongoing edition) about the race, riders, teams, etc., is stored."

2 Entity Identification

Based on these specifications, we identify the most important entities and the related information.

- **A rider** is **a person** for whom we record their first and last name, height, date of birth, and the **team** they belong to.
- **A team** is identified by its name, has a budget, a **sports director**, and is funded by **sponsors**.
- **A sports director** is **a person** for whom we store their first and last name and date of birth.
- **A sponsor** is identified by its name, address, and field of activity.
- **A race** corresponds to a race name, and we know the total distance to be covered.
- **A stage** has an order number (e.g., "Stage 3"), a date, a type, a starting city, and a finishing city. A ranking determines a rider's position during a stage of a race.
- **A soigneur** is **a person** for whom we record their first and last name, date of birth, and nationality.
- **A product** is identified by a product number and has a name, an indication, a contraindication, and a dosage recommendation.
- **A ranking** obtained by **a rider** during a stage is recorded.

3 Identification of Associations

Associations link entities together. We identify associations from subject-verb-object sentences in the specifications. The verb represents the association between two entities (Subject, Object).

3.1 Entity-Association

Based on the Union Cycliste Internationale specifications, we extract key sentences that connect entities through verbs:

- A person **is** a rider.
- A person **is** a sports director.
- A person **is** a soigneur.
- A rider **belongs to** a team.
- A soigneur **treats** a rider.
- A stage **is won by** a rider.
- A ranking **determines the position of** a rider.

- A ranking **is achieved during** a stage.
- A race **includes** stages.
- A stage **is won by** a team.
- A soigneur **provides treatment during** a race.
- A soigneur **treats at the request of** a team.

3.2 UML Modeling (Version 1.1)

From these associations between entities, we can propose an initial version of the database structure using UML formalism.

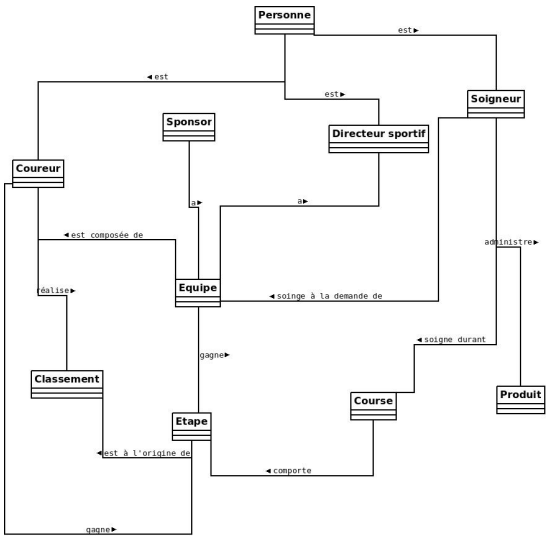


Figure 1: Model: Entity-Association

4 Identification of Cardinalities

From these sentences in the specifications, we can also identify the cardinality of associations (one-to-one, one-to-many, many-to-many, etc.). For example, a treatment is administered during one or multiple races, and a team is funded by at least one sponsor.

4.1 Association-Cardinality

Identifying cardinalities is crucial to managing referential constraints between entities. These cardinalities determine which entity (table) should hold a reference (foreign key) to another entity (table) in an association or whether a new entity is required to link the two entities. This is especially necessary for many-to-many (M:N) relationships.

- **A** person **can be (but is not necessarily)** a rider, a sports director, or a soigneur.
- **A** team consists of **multiple** riders.
- **A** rider **may or may not** win stages.
- **A** rider **may or may not** achieve rankings.
- **A** stage results in **at least one** ranking.
- **A** race consists of **at least one** stage.
- **A** team **may or may not** win stages.
- **Multiple** soigneurs provide treatment during **multiple** races.
- **Multiple** soigneurs provide treatment at the request of **multiple** teams.
- **Multiple** soigneurs administer **multiple** products.
- **A** team **must have exactly one** sports director.
- **A** team has **at least one** sponsor.

4.2 UML Modeling (Version 1.2)

Based on the identified cardinalities, we can propose a second version of the UML model.

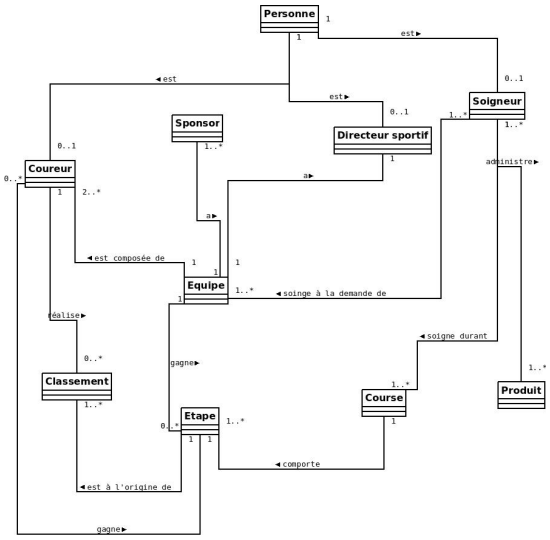


Figure 2: Model: Association Cardinalities

4.3 UML Modeling (Version 1.3)

The previous UML model reveals several many-to-many relationships, which must systematically be broken down into two one-to-many associations. We can express this refinement more precisely by introducing associative tables (entities) and redefining the associations between these new tables and the two original entities.

First:

- **Multiple** soigneurs provide treatment during **multiple** races.

becomes:

- **A single** treatment is administered during **multiple** races.
- **Multiple** soigneurs provide **a single** treatment.

Then:

- **Multiple** soigneurs provide treatment at the request of **multiple** teams.

becomes:

- **Multiple** soigneurs provide **a single** treatment.
- **Multiple** teams receive **a single** treatment.

Finally:

- **Multiple** soigneurs administer **multiple** products.

becomes:

- **Multiple** soigneurs administer **a single** dose.
- **Multiple** products are used within **the same** dose.

This results in the following model:

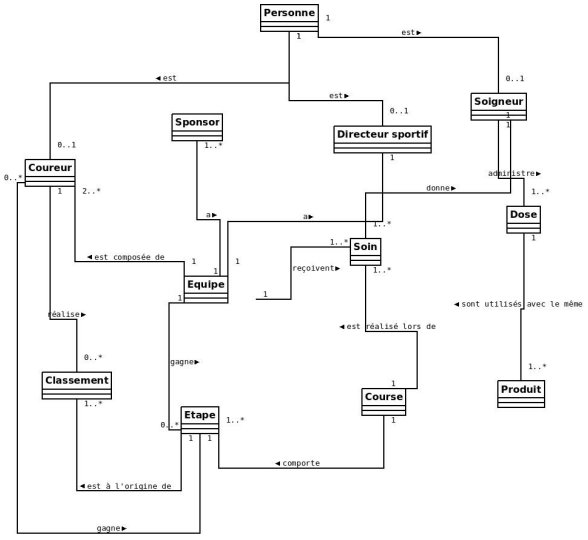


Figure 3: Model: Association Cardinalities

5 Entity-Attributes

After identifying the key entities from the specifications, we must extract the information that characterizes them.

5.1 Attribute Identification

From our specifications, we can retain the following characteristics:

- A person is identified by a **name**, a **surname**, and their **date of birth**.
- A rider is identified by their **height**.
- A team is identified by its **name** and has a **budget**.

- A sponsor has a **name**, an **address**, and a **business sector**.
- A race has a **name** and a **total distance**.
- A stage has an **order number**, a **date**, a **type**, a **departure city**, and an **arrival city**.
- A ranking corresponds to a **position** obtained.
- A soigneur is identified by their **nationality**.
- A dose is a **quantity** of a product.
- A product has a **number**, a **name**, an **indication**, a **contraindication**, and a **dosage**.

5.2 UML Modeling (Version 1.4)

We can represent the entity attributes in a third version of the UML model.

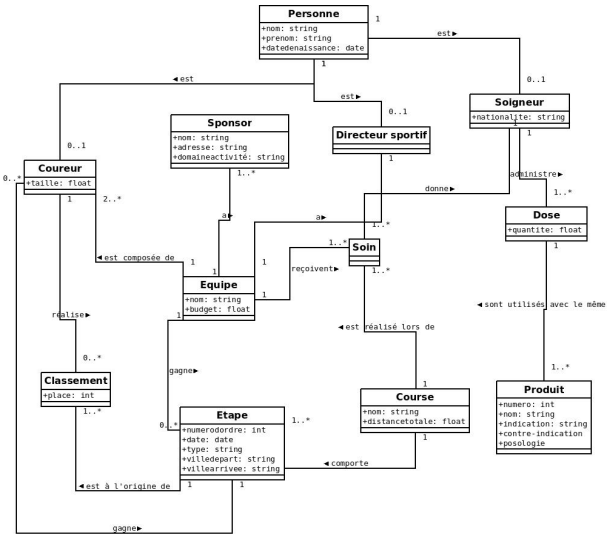


Figure 4: Model: Attributes

This version can be considered as the first version to be proposed to the client based on the initial specifications.

6 From UML to SQL

The transition from a UML-based representation to a relational database structure requires defining attributes within entities that will serve as primary keys for tables and foreign keys referencing the primary keys of associated entities (tables).

6.1 UML Modeling (Version 1.4)

Based on the cardinalities of the associations, we can represent primary and foreign keys in the previous UML model (e.g., personne.id, soigneur.personne_id, etc.).

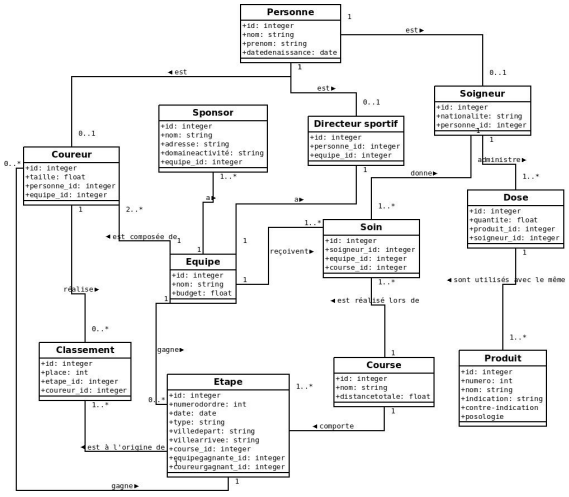


Figure 5: Model: Key Attributes

6.2 Database Structuring

From this UML schema, we can implement the database structure in SQL.

- personne (<u>id</u>, nom, prenom, datedenaissance)
- coureur (<u>id</u>, taille, #personne_id, #equipe_id)
- sponsor (<u>id</u>, nom, adresse, domaineactivité, #equipe_id)
- directeur sportif (<u>id</u>, #personne_id, #equipe_id)
- soigneur (<u>id</u>, nationalite, #personne_id)
- equipe (<u>id</u>, nom, budget, #ds_id)
- soin (<u>#soigneur_id, #equipe_id, #course_id</u>)
- dose (<u>#produit_id, #soigneur_id</u>, quantite)
- classement (<u>id</u>, place, #etape_id, #coureur_id)
- etape (<u>id</u>, numerodordre, date, type, villedepart, villearrivee, #course_id, #equipegagnante_id, #coureurgagnante_id)
- course (<u>id</u>, quantite, #produit_id, #soigneur_id)
- produit (<u>id</u>, quantite, #produit_id, #soigneur_id)

In this representation, primary keys are underlined, and foreign keys are prefixed with a hashtag (#). In SQLite, it is neither necessary nor recommended to define an automatically incremented (AUTOINCREMENT) primary key, as a rowid is directly available.

With this structure, we can create the database.

```
--creation de coureur
CREATE TABLE coureur (
        taille FLOAT,
        personne_id INTEGER,
        equipe_id INTEGER,
        FOREIGN KEY ( personne_id ) REFERENCES personnes,
        FOREIGN KEY ( equipe_id ) REFERENCES equipe
);

-- postes primary key : rowid
SELECT rowid from coureur ;
rowid
-- ---
1
2
...
```

The full SQL script for creating all database tables is available in the appendix.

6.3 Inserting Data into the Database

With this structure in place, we can populate the database by inserting data and performing updates using SQL commands such as:

```
INSERT INTO ... VALUES (...);
UPDATE ... SET ... WHERE ...;
```

```
-- insertion de personne
INSERT INTO personne ( prenom , nom , datedenaissance ) VALUES ( 'Bernard' , 'Hinault' , '14-11-1954') ;
INSERT INTO personne ( prenom , nom , datedenaissance ) VALUES ( 'Thomas' , 'Voeckler' , '22-06-1979') ;
INSERT INTO personne ( prenom , nom , datedenaissance ) VALUES ( 'Bryan' , 'Coquard' , '25-04-1992');
INSERT INTO personne ( prenom , nom , datedenaissance ) VALUES ( 'Morgan' , 'Lamoisson' , '07-09-1988' );
INSERT INTO personne ( prenom , nom , datedenaissance ) VALUES ( 'Jean' , 'Dupont' , '04-03-1960');

-- insertion de equipe
INSERT INTO equipe ( nom, budget ) VALUES ( 'Vend e U', 80000.15);
INSERT INTO equipe ( nom, budget ) VALUES ( 'Gitane-Campagnolo', 40010.25);

-- insertion de coureur
INSERT INTO coureur(taille, personne_id, equipe_id) VALUES (
    1.70,
    (SELECT rowid FROM personne WHERE prenom="Thomas" AND nom="Voeckler"),
    (SELECT rowid FROM equipe WHERE nom="Vend e U"));

INSERT INTO coureur(taille, personne_id, equipe_id) VALUES (
    1.75,
    (SELECT rowid FROM personne WHERE prenom="Bryan" AND nom="Coquard"),
    (SELECT rowid FROM equipe WHERE nom="Vend e U"));

INSERT INTO coureur(taille, personne_id, equipe_id) VALUES (
    1.80,
    (SELECT rowid FROM personne WHERE prenom="Bernard" AND nom="Hinault"),
    (SELECT rowid FROM equipe WHERE nom="Gitane-Campagnolo"));

-- insertion de directeur sportif
INSERT INTO directeur_sportif ( personne_id , equipe_id ) VALUES (
        (SELECT rowid FROM personne WHERE prenom = 'Morgan' AND nom = 'Lamoisson'),
        (SELECT rowid FROM equipe WHERE nom = 'Vend e U'));

-- insertion de soigneur
INSERT INTO soigneur ( nationalite, personne_id) VALUES (
        'Francais',
        (SELECT rowid FROM personne WHERE prenom = 'Jean' AND nom = 'Dupont'));

-- insertion de sponsor
INSERT INTO sponsor ( nom, adresse, domaineactivite, equipe_id ) VALUES ( 'Syst me U',
    '20 RUE D ARCUEIL PARC TERTIAIRE SILIC BATIMENT MO 94150 RUNGIS',
    'Grande distribution',
    (SELECT rowid FROM equipe WHERE nom = 'Vend e U'));

INSERT INTO sponsor ( nom, adresse, domaineactivite, equipe_id ) VALUES ( 'D partement Vend e',
    '2 Av. Gordon Bennett, 75016 Paris',
    'D partement',
    (SELECT rowid FROM equipe WHERE nom = 'Vend e U'));

-- insertion de course
INSERT INTO course ( nom, distancetotale ) VALUES ( 'Tour de France',  3000);
INSERT INTO course ( nom, distancetotale ) VALUES ( 'Tour d Espagne',  1000);

-- insertion de soin
INSERT INTO soin ( soigneur_id, equipe_id, course_id ) VALUES (
        (SELECT s.rowid FROM personne p CROSS JOIN soigneur s WHERE prenom = 'Jean' AND nom = 'Dupont'
             AND s.personne_id = p.rowid),
        (SELECT rowid FROM equipe WHERE nom = 'Vend e U'),
        (SELECT rowid FROM course WHERE nom = 'Tour de France'));

-- insertion de produit
INSERT INTO produit ( numero, nom, indication, contre_indication, posologie ) VALUES ( 1547,
        'Salbutamol',
        'douleur musculaire',
        'ne pas administrer en dessous de 20 ans',
        '1 comprim  par jour');

-- insertion de dose
INSERT INTO dose ( quantite, produit_id, soigneur_id ) VALUES ( '2.85',
        (SELECT rowid FROM produit WHERE nom = 'Salbutamol'),
```

```
69          (SELECT s.rowid FROM personne p CROSS JOIN soigneur s WHERE prenom = 'Jean' AND nom = 'Dupont'
                AND s.personne_id = p.rowid));
70
71  -- insertion de etape
72  INSERT INTO etape ( numeroordre, date, type, villedepart, villearivee, course_id, equipegagnante_id,
        coureurgagnant_id ) VALUES (
73      '1re␣ tape ',
74      '15-01-2015',
75      'Contre␣la␣montre␣individuel',
76      'Lorient',
77      'Brest',
78      (SELECT rowid FROM course WHERE nom = 'Tour␣de␣France'),
79      (SELECT rowid FROM equipe WHERE nom = 'Vend e␣U'),
80      (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Thomas' AND nom = 'Voeckler'
            AND c.personne_id = p.rowid));
81
82  INSERT INTO etape ( numeroordre, date, type, villedepart, villearivee, course_id, equipegagnante_id,
        coureurgagnant_id ) VALUES (
83      '2nd␣ tape ',
84      '16-02-2015',
85      'Course␣en␣ligne',
86      'Brest',
87      'Lannion',
88      (SELECT rowid FROM course WHERE nom = 'Tour␣de␣France'),
89      NULL,
90      NULL);
91
92  INSERT INTO etape ( numeroordre, date, type, villedepart, villearivee, course_id, equipegagnante_id,
        coureurgagnant_id ) VALUES (
93      '3 me ␣ tape ',
94      '17-02-2015',
95      'Course␣en␣ligne',
96      'Lannion',
97      'Rennes',
98      (SELECT rowid FROM course WHERE nom = 'Tour␣de␣France'),
99      NULL,
100     NULL);
101
102 INSERT INTO etape ( numeroordre, date, type, villedepart, villearivee, course_id, equipegagnante_id,
        coureurgagnant_id ) VALUES (
103     '4 me ␣ tape ',
104     '18-02-2015',
105     'Course␣en␣ligne',
106     'Rennes',
107     'Nantes',
108     (SELECT rowid FROM course WHERE nom = 'Tour␣de␣France'),
109     NULL,
110     NULL);
111
112 -- insertion de classement
113 INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
114     '1er',
115     (SELECT rowid FROM etape WHERE numeroordre = '1re␣ tape '),
116     (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Thomas' AND nom = 'Voeckler'
            AND c.personne_id = p.rowid));
117
118 INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
119     '3 me ',
120     (SELECT rowid FROM etape WHERE numeroordre = '3 me ␣ tape '),
121     (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Thomas' AND nom = 'Voeckler'
            AND c.personne_id = p.rowid));
122
123 INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
124     '2nd',
125     (SELECT rowid FROM etape WHERE numeroordre = '2nd␣ tape '),
126     (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Thomas' AND nom = 'Voeckler'
            AND c.personne_id = p.rowid));
127
128 INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
129     '4 me ',
130     (SELECT rowid FROM etape WHERE numeroordre = '4 me ␣ tape '),
131     (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Thomas' AND nom = 'Voeckler'
            AND c.personne_id = p.rowid));
132
133 INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
134     '2nd',
135     (SELECT rowid FROM etape WHERE numeroordre = '1re␣ tape '),
136     (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Bernard' AND nom = 'Hinault'
            AND c.personne_id = p.rowid));
137
138 INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
```

```
    '1er',
    (SELECT rowid FROM etape WHERE numeroordre = '3 me ␣ tape '),
    (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Bernard' AND nom = 'Hinault'
        AND c.personne_id = p.rowid));

INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
    '1er',
    (SELECT rowid FROM etape WHERE numeroordre = '2nd␣ tape '),
    (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Bernard' AND nom = 'Hinault'
        AND c.personne_id = p.rowid));

INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
    '3 me ',
    (SELECT rowid FROM etape WHERE numeroordre = '1re␣ tape '),
    (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Bryan' AND nom = 'Coquard'
        AND c.personne_id = p.rowid));
```

6.4 Joins: Retrieving Information

Once the database is properly structured, we must be able to retrieve all necessary information using table joins. The SQL query below demonstrates a join across all database tables. Column renaming is applied where multiple tables contain attributes with the same name.

```
-- les coureurs de l' quipe  'Vend e U'
SELECT p.nom
FROM equipe e, coureur c, personne p
WHERE e.rowid = c.equipe_id AND p.rowid = c.personne_id AND e.nom = 'Vend e␣U';

-- jointure entre toutes les tables ( renommage de colonnes de meme ... nom )
SELECT DISTINCT per.nom AS "NOM",
                                per.prenom AS "PRENOM",
                                per.datedenaissance,
                                eq.nom AS "EQUIPE",
                                eq.budget,
                                cr.taille ,
                                sg.nationalite,
                                spon.nom AS "SPONSOR",
                                spon.adresse,
                                spon.domaineactivite ,
                                cs.nom AS "COURSE",
                                cs.distancetotale ,
                                pro.numero,
                                pro.nom  AS "PRODUIT",
                                pro.indication,
                                pro.contre_indication,
                                pro.posologie,
                                do.quantite AS "DOSE",
                                et.numeroordre AS "ETAPE",
                                et.date,
                                et.type,
                                et.villedepart,
                                et.villearivee,
                                cla.place AS "CLASSEMENT"
FROM personne per, equipe eq ,coureur cr,
         directeur_sportif ds, soigneur sg, sponsor spon,
         course cs, soin s, produit pro, dose do, etape et,
         classement cla
WHERE per.rowid = cr.personne_id
 AND eq.rowid = cr.equipe_id
-- AND per.rowid = ds.personne_id
 AND eq.rowid = ds.equipe_id
-- AND per.rowid = sg.personne_id
 AND eq.rowid = spon.equipe_id
 AND sg.rowid = s.soigneur_id
 AND eq.rowid = s.equipe_id
 AND cs.rowid = s.course_id
 AND pro.rowid = do.produit_id
 AND sg.rowid = do.soigneur_id
 AND cs.rowid = et.course_id
 AND eq.rowid = et.equipegagnante_id
 AND cr.rowid = et.coureurgagnant_id
 AND et.rowid = cla.etape_id
 AND cr.rowid = cla.coureur_id;
```

7 Use Case

Based on this data model, we can represent the use cases that will allow us to test the queries:

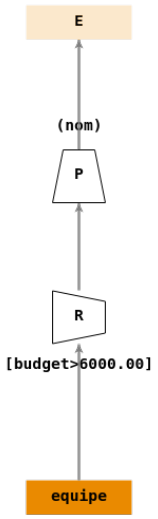
1. on a single table with projections ($\prod$) and restriction criteria (σ)
2. on multiple tables using joins ($\bowtie$) and restriction criteria (σ)
3. using set queries ($\cup, \cap,$)
4. with a relational division ($\div$)
5. applying aggregate functions (count(), sum(), max(), min(), avg() ...)
6. performing groupings (GROUP BY)
7. then groupings with restriction criteria (GROUP BY ... HAVING)

7.1 Information Retrieval

Using this company management model, we could implement the following use cases:

1. $\prod$, σ: the name of all teams with a budget greater than €6000.00
2. $\prod$, $\bowtie$, σ: the height of all cyclists in the "Vendée U" team
3. $\cup$, $\cap$, : the name of the sponsors of the "Vendée U" team that do not have retail as their business domain
4. $\div$: the stage where all the cyclists of the "Vendée U" team would be ranked
5. count(), sum(), max(), min(), avg() ... : the average total distance of the races
6. GROUP BY: the number of rankings per cyclist of the "Vendée U" team
7. GROUP BY, HAVING: the number of rankings per cyclist of the "Vendée U" team when this number is greater than 3

1. The name of all teams with a budget greater than €6000.00
   - Relational Calculus:
     - E = {(e.name) | e ∈ team ∧ e.budget > 6000.00}
   - Relational Algebra:
     - E = $\prod_{(name)}(\sigma_{[budget>6000.00]}(team))$
   - Query Tree:



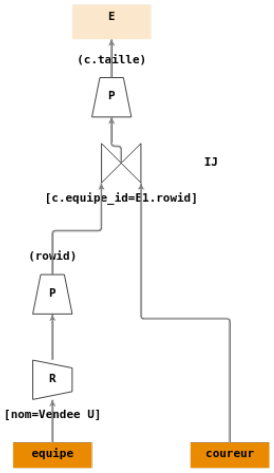
   - SQL Query:

```
SELECT nom
FROM equipe
WHERE budget > 6000.00;
```

2. The height of all cyclists in the "Vendée U" team
   - Relational Calculus:
     - E1: The "Vendée U" team

- E1 = {e.rowid | team(e) ∧ e.name = 'Vendée U' }
- E = {c.height | cyclist(c) ∧ E1(e1) ∧ c.team_id = e1.rowid}

- Relational Algebra:
  - E1: The "Vendée U" team
  - E1 = $\prod_{(rowid)}(\sigma_{[name=VendeU]}(team))$
  - E = $\prod_{(c.height)}(\bowtie_{[c.team\_id=E1.rowid]} (cyclist\ c, E1))$
- Query Tree:



- SQL Query:

```
SELECT p.prenom || ' ' || p.nom AS "coureur", taille
FROM coureur c, equipe e, personne p
WHERE c.equipe_id = e.rowid AND e.nom = 'Vend e U' and c.personne_id = p.rowid;
```

8 Appendices

8.1 Table Creation

```
DROP TABLE IF EXISTS personne;
DROP TABLE IF EXISTS equipe;
DROP TABLE IF EXISTS coureur;
DROP TABLE IF EXISTS directeur_sportif;
DROP TABLE IF EXISTS soigneur;
DROP TABLE IF EXISTS sponsor;
DROP TABLE IF EXISTS course;
DROP TABLE IF EXISTS soin;
DROP TABLE IF EXISTS produit;
DROP TABLE IF EXISTS dose;
DROP TABLE IF EXISTS etape;
DROP TABLE IF EXISTS classement;

--creation de personne
CREATE TABLE personne (
        nom varchar (20),
        prenom varchar (20),
        datedenaissance TEXT
);

--creation de equipe
CREATE TABLE equipe (
        nom varchar(20),
        budget FLOAT
);

--creation de coureur
CREATE TABLE coureur (
        taille FLOAT,
        personne_id INTEGER,
        equipe_id INTEGER,
        FOREIGN KEY ( personne_id ) REFERENCES personnes,
        FOREIGN KEY ( equipe_id ) REFERENCES equipe
```

```
);

--creation de directeur sportif
CREATE TABLE directeur_sportif(
        personne_id INTEGER,
        equipe_id INTEGER,
        FOREIGN KEY ( personne_id ) REFERENCES personnes,
        FOREIGN KEY ( equipe_id ) REFERENCES equipe
);

--creation de soigneur
CREATE TABLE soigneur(
        nationalite varchar(20),
        personne_id INTEGER,
        FOREIGN KEY ( personne_id ) REFERENCES personnes
);

--creation de sponsor
CREATE TABLE sponsor(
        nom varchar(20),
        adresse varchar(30),
        domaineactivite varchar(20),
        equipe_id INTEGER,
        FOREIGN KEY ( equipe_id ) REFERENCES equipe
);

--creation de course
CREATE TABLE course(
        nom varchar(20),
        distancetotale FLOAT
);

--creation de soin
CREATE TABLE soin(
        soigneur_id INTEGER,
        equipe_id INTEGER,
        course_id INTEGER,
        FOREIGN KEY ( equipe_id ) REFERENCES equipe,
        FOREIGN KEY ( course_id ) REFERENCES course
);

--creation de produit
CREATE TABLE produit(
        numero INTEGER,
        nom varchar(20),
        indication varchar(30),
        contre_indication varchar(30),
        posologie varchar(30)
);

--creation de dose
CREATE TABLE dose(
        quantite FLOAT,
        produit_id INTEGER,
        soigneur_id INTEGER,
        FOREIGN KEY ( produit_id ) REFERENCES produit,
        FOREIGN KEY ( soigneur_id ) REFERENCES soigneur
);

--creation de etape
CREATE TABLE etape(
        numeroordre varchar(20),
        date TEXT,
        type varchar(30),
        villedepart varchar(20),
        villearivee varchar(20),
        course_id INTEGER,
        equipegagnante_id INTEGER,
        coureurgagnant_id INTEGER,
        FOREIGN KEY ( equipegagnante_id ) REFERENCES equipe,
        FOREIGN KEY ( coureurgagnant_id ) REFERENCES coureur
);

--creation de classement
CREATE TABLE classement(
        place varchar(10),
        etape_id INTEGER,
        coureur_id INTEGER,
        FOREIGN KEY ( etape_id ) REFERENCES etape,
        FOREIGN KEY ( coureur_id ) REFERENCES coureur
);
```

8.2 Data Insertion

```
-- insertion de personne
INSERT INTO personne ( prenom , nom , datedenaissance ) VALUES ( 'Bernard' , 'Hinault' , '14-11-1954') ;
INSERT INTO personne ( prenom , nom , datedenaissance ) VALUES ( 'Thomas' , 'Voeckler' , '22-06-1979') ;
INSERT INTO personne ( prenom , nom , datedenaissance ) VALUES ( 'Bryan' , 'Coquard' , '25-04-1992');
INSERT INTO personne ( prenom , nom , datedenaissance ) VALUES ( 'Morgan' , 'Lamoisson' , '07-09-1988' );
INSERT INTO personne ( prenom , nom , datedenaissance ) VALUES ( 'Jean' , 'Dupont' , '04-03-1960');

-- insertion de equipe
INSERT INTO equipe ( nom, budget ) VALUES ( 'Vend e U', 80000.15);
INSERT INTO equipe ( nom, budget ) VALUES ( 'Gitane-Campagnolo', 40010.25);

-- insertion de coureur
INSERT INTO coureur(taille, personne_id, equipe_id) VALUES (
    1.70,
    (SELECT rowid FROM personne WHERE prenom="Thomas" AND nom="Voeckler"),
    (SELECT rowid FROM equipe WHERE nom="Vend e U"));

INSERT INTO coureur(taille, personne_id, equipe_id) VALUES (
    1.75,
    (SELECT rowid FROM personne WHERE prenom="Bryan" AND nom="Coquard"),
    (SELECT rowid FROM equipe WHERE nom="Vend e U"));

INSERT INTO coureur(taille, personne_id, equipe_id) VALUES (
    1.80,
    (SELECT rowid FROM personne WHERE prenom="Bernard" AND nom="Hinault"),
    (SELECT rowid FROM equipe WHERE nom="Gitane-Campagnolo"));

-- insertion de directeur sportif
INSERT INTO directeur_sportif ( personne_id , equipe_id ) VALUES (
        (SELECT rowid FROM personne WHERE prenom = 'Morgan' AND nom = 'Lamoisson'),
        (SELECT rowid FROM equipe WHERE nom = 'Vend e U'));

-- insertion de soigneur
INSERT INTO soigneur ( nationalite, personne_id) VALUES (
        'Francais',
        (SELECT rowid FROM personne WHERE prenom = 'Jean' AND nom = 'Dupont'));

-- insertion de sponsor
INSERT INTO sponsor ( nom, adresse, domaineactivite, equipe_id ) VALUES ( 'Syst me U',
    '20 RUE D ARCUEIL PARC TERTIAIRE SILIC BATIMENT MO 94150 RUNGIS',
    'Grande distribution',
    (SELECT rowid FROM equipe WHERE nom = 'Vend e U'));

INSERT INTO sponsor ( nom, adresse, domaineactivite, equipe_id ) VALUES ( 'D partement Vend e',
    '2 Av. Gordon Bennett, 75016 Paris',
    'D partement',
    (SELECT rowid FROM equipe WHERE nom = 'Vend e U'));

-- insertion de course
INSERT INTO course ( nom, distancetotale ) VALUES ( 'Tour de France',  3000);
INSERT INTO course ( nom, distancetotale ) VALUES ( 'Tour d Espagne',  1000);

-- insertion de soin
INSERT INTO soin ( soigneur_id, equipe_id, course_id ) VALUES (
        (SELECT s.rowid FROM personne p CROSS JOIN soigneur s WHERE prenom = 'Jean' AND nom = 'Dupont'
             AND s.personne_id = p.rowid),
        (SELECT rowid FROM equipe WHERE nom = 'Vend e U'),
        (SELECT rowid FROM course WHERE nom = 'Tour de France'));

-- insertion de produit
INSERT INTO produit ( numero, nom, indication, contre_indication, posologie ) VALUES ( 1547,
        'Salbutamol',
        'douleur musculaire',
        'ne pas administrer en dessous de 20 ans',
        '1 comprim  par jour');

-- insertion de dose
INSERT INTO dose ( quantite, produit_id, soigneur_id ) VALUES ( '2.85',
        (SELECT rowid FROM produit WHERE nom = 'Salbutamol'),
        (SELECT s.rowid FROM personne p CROSS JOIN soigneur s WHERE prenom = 'Jean' AND nom = 'Dupont'
             AND s.personne_id = p.rowid));

-- insertion de etape
INSERT INTO etape ( numeroordre, date, type, villedepart, villearivee, course_id, equipegagnante_id,
    coureurgagnant_id ) VALUES (
    '1re  tape ',
```

```
74      '15-01-2015',
75      'Contre␣la␣montre␣individuel',
76      'Lorient',
77      'Brest',
78      (SELECT rowid FROM course WHERE nom = 'Tour␣de␣France'),
79      (SELECT rowid FROM equipe WHERE nom = 'Vend e␣U'),
80      (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Thomas' AND nom = 'Voeckler'
            AND c.personne_id = p.rowid));
81
82  INSERT INTO etape ( numeroordre, date, type, villedepart, villearivee, course_id, equipegagnante_id,
        coureurgagnant_id ) VALUES (
83      '2nd␣ tape ',
84      '16-02-2015',
85      'Course␣en␣ligne',
86      'Brest',
87      'Lannion',
88      (SELECT rowid FROM course WHERE nom = 'Tour␣de␣France'),
89      NULL,
90      NULL);
91
92  INSERT INTO etape ( numeroordre, date, type, villedepart, villearivee, course_id, equipegagnante_id,
        coureurgagnant_id ) VALUES (
93      '3 me ␣ tape ',
94      '17-02-2015',
95      'Course␣en␣ligne',
96      'Lannion',
97      'Rennes',
98      (SELECT rowid FROM course WHERE nom = 'Tour␣de␣France'),
99      NULL,
100     NULL);
101
102 INSERT INTO etape ( numeroordre, date, type, villedepart, villearivee, course_id, equipegagnante_id,
        coureurgagnant_id ) VALUES (
103     '4 me ␣ tape ',
104     '18-02-2015',
105     'Course␣en␣ligne',
106     'Rennes',
107     'Nantes',
108     (SELECT rowid FROM course WHERE nom = 'Tour␣de␣France'),
109     NULL,
110     NULL);
111
112 -- insertion de classement
113 INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
114     '1er',
115     (SELECT rowid FROM etape WHERE numeroordre = '1re␣ tape '),
116     (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Thomas' AND nom = 'Voeckler'
            AND c.personne_id = p.rowid));
117
118 INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
119     '3 me ',
120     (SELECT rowid FROM etape WHERE numeroordre = '3 me ␣ tape '),
121     (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Thomas' AND nom = 'Voeckler'
            AND c.personne_id = p.rowid));
122
123 INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
124     '2nd',
125     (SELECT rowid FROM etape WHERE numeroordre = '2nd␣ tape '),
126     (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Thomas' AND nom = 'Voeckler'
            AND c.personne_id = p.rowid));
127
128 INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
129     '4 me ',
130     (SELECT rowid FROM etape WHERE numeroordre = '4 me ␣ tape '),
131     (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Thomas' AND nom = 'Voeckler'
            AND c.personne_id = p.rowid));
132
133 INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
134     '2nd',
135     (SELECT rowid FROM etape WHERE numeroordre = '1re␣ tape '),
136     (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Bernard' AND nom = 'Hinault'
            AND c.personne_id = p.rowid));
137
138 INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
139     '1er',
140     (SELECT rowid FROM etape WHERE numeroordre = '3 me ␣ tape '),
141     (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Bernard' AND nom = 'Hinault'
            AND c.personne_id = p.rowid));
142
143 INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
144     '1er',
```

```
    (SELECT rowid FROM etape WHERE numeroordre = '2nd tape '),
    (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Bernard' AND nom = 'Hinault'
        AND c.personne_id = p.rowid));

INSERT INTO classement ( place, etape_id, coureur_id ) VALUES (
    '3 me ',
    (SELECT rowid FROM etape WHERE numeroordre = '1re tape '),
    (SELECT c.rowid FROM coureur c CROSS JOIN personne p WHERE prenom = 'Bryan' AND nom = 'Coquard'
        AND c.personne_id = p.rowid));
```

8.3 Data Update

```
UPDATE coureur
SET taille = 1.82
WHERE personne_id = (SELECT rowid FROM personne WHERE nom = 'Thomas' AND prenom = 'Voeckler');

UPDATE soigneur
SET nationalite = 'Allemand/Francais'
WHERE personne_id = (SELECT rowid FROM personne WHERE nom = 'Jean' AND prenom = 'Dupont');

UPDATE sponsor
SET adresse = '2bis rue Louis Armand, 75015 PARIS'
WHERE nom = 'Syst me U';
```

8.4 Table Joins

```
-- les coureurs de l' quipe  'Vend e U'
SELECT p.nom
FROM equipe e, coureur c, personne p
WHERE e.rowid = c.equipe_id AND p.rowid = c.personne_id AND e.nom = 'Vend e U';

-- jointure entre toutes les tables ( renommage de colonnes de meme ... nom )
SELECT DISTINCT per.nom AS "NOM",
                                per.prenom AS "PRENOM",
                                per.datedenaissance,
                                eq.nom AS "EQUIPE",
                                eq.budget,
                                cr.taille ,
                                sg.nationalite,
                                spon.nom AS "SPONSOR",
                                spon.adresse,
                                spon.domaineactivite ,
                                cs.nom AS "COURSE",
                                cs.distancetotale ,
                                pro.numero,
                                pro.nom  AS "PRODUIT",
                                pro.indication,
                                pro.contre_indication,
                                pro.posologie,
                                do.quantite AS "DOSE",
                                et.numeroordre AS "ETAPE",
                                et.date,
                                et.type,
                                et.villedepart,
                                et.villearivee,
                                cla.place AS "CLASSEMENT"
FROM personne per, equipe eq ,coureur cr,
         directeur_sportif ds, soigneur sg, sponsor spon,
         course cs, soin s, produit pro, dose do, etape et,
         classement cla
WHERE per.rowid = cr.personne_id
 AND eq.rowid = cr.equipe_id
-- AND per.rowid = ds.personne_id
 AND eq.rowid = ds.equipe_id
-- AND per.rowid = sg.personne_id
 AND eq.rowid = spon.equipe_id
 AND sg.rowid = s.soigneur_id
 AND eq.rowid = s.equipe_id
 AND cs.rowid = s.course_id
 AND pro.rowid = do.produit_id
 AND sg.rowid = do.soigneur_id
 AND cs.rowid = et.course_id
 AND eq.rowid = et.equipegagnante_id
 AND cr.rowid = et.coureurgagnant_id
 AND et.rowid = cla.etape_id
 AND cr.rowid = cla.coureur_id;
```