# Rapport sur la Modélisation de la Base de Données pour les Parcours de Randonnées

### 3.2 Modèle Relationnel

**Requêtes SQL :**

1. **Les noms des étapes accessibles directement depuis Orléans en suivant les tronçons qui en partent :**
    - Nous avons besoin des noms des étapes accessibles directement depuis Orléans. Pour cela, nous devons trouver tous les tronçons qui partent d'Orléans.
    - Nous sélectionnons les étapes dont l'ID correspond à l'ID de départ des tronçons partant d'Orléans.
    - Utilisation de la clause DISTINCT pour éviter les doublons.
2. **La liste des étapes accessibles depuis Orléans avec une seule étape intermédiaire :**
    - Nous devons trouver toutes les étapes accessibles depuis Orléans avec une seule étape intermédiaire. Cela signifie qu'il y a deux tronçons nécessaires pour atteindre ces étapes.
    - Nous joignons les tables Etape et Troncon deux fois pour représenter les deux tronçons nécessaires.
    - La première jointure nous donne les étapes accessibles directement depuis Orléans.
    - La deuxième jointure nous donne les étapes accessibles depuis les étapes trouvées dans la première jointure.
    - Nous sélectionnons les étapes trouvées dans la deuxième jointure.
    - Utilisation de la clause DISTINCT pour éviter les doublons.
3. **La liste des étapes accessibles depuis Orléans avec un nombre quelconque d'étapes intermédiaires :**
    - Nous devons trouver toutes les étapes accessibles depuis Orléans avec un nombre quelconque d'étapes intermédiaires, c'est-à-dire qu'il n'y a pas de limite au nombre de tronçons nécessaires pour atteindre ces étapes.
    - Nous utilisons une requête récursive pour parcourir les tronçons et les étapes intermédiaires.
    - Dans la partie récursive de la requête, nous sélectionnons les tronçons et les étapes suivantes en utilisant les étapes précédemment trouvées.
    - Nous continuons à répéter cette opération jusqu'à ce qu'il n'y ait plus de nouvelles étapes à découvrir.
    - Enfin, nous sélectionnons toutes les étapes trouvées dans la table résultante.
    - Utilisation de la clause DISTINCT pour éviter les doublons.

### 3.3 Modèle Objet-Relationnel

**Requêtes SQL :**

1. Nombre de points d'intérêt par type pour chaque tronçon :
    - Cette requête vise à obtenir le nombre de points d'intérêt de chaque type présents sur chaque tronçon. Elle nous permet d'analyser la distribution des points d'intérêt le long des itinéraires.
    - Nous utilisons des agrégations pour compter le nombre de points d'intérêt de chaque type.
    - Les résultats sont regroupés par tronçon, permettant ainsi une analyse détaillée de chaque segment de l'itinéraire.
2. Impact de chaque indice de qualité sur les tronçons :
    - Cette requête est conçue pour évaluer l'impact de chaque indice de qualité sur les tronçons de l'itinéraire.

- Nous calculons une mesure d'impact en agrégeant les valeurs des indices de qualité pour chaque tronçon.
- Les résultats nous permettent d'identifier les tronçons les plus influencés par certains critères de qualité, ce qui peut être utile pour optimiser l'expérience des randonneurs.

### 3.4 Modèle Logique

**Implémentation avec Datalog/Requêtes :**

1. Création du modèle
2. Ce programme définit une relation `etapesConnexes` qui représente les étapes connexes entre elles. La première règle stipule que deux étapes sont connexes si elles sont reliées par un tronçon direct. La deuxième règle étend cette connexion aux étapes reliées par plusieurs tronçons successifs en utilisant la récursivité. Enfin, la requête `etapesConnexes('Orleans', 'Fleury-les-Aubrais')` permet de vérifier si les deux étapes spécifiées sont connectées.
3. Ce programme calcule la distance totale d'un parcours constitué de tronçons terrestres entre deux étapes. Il utilise la récursivité pour parcourir les tronçons successifs jusqu'à atteindre la destination. La première règle correspond au cas de base où les deux étapes sont directement reliées par un tronçon terrestre, tandis que la deuxième règle gère les cas où plusieurs tronçons terrestres sont nécessaires. La requête `parcoursEnTerre('Orleans', 'Fleury-les-Aubrais', DistanceTotale)` permet de calculer la distance totale du parcours entre Orléans et Fleury-les-Aubrais.
4. Ce programme identifie les parcours entre deux étapes qui ont un nombre impair de tronçons. Il utilise la relation `etapesConnexes` pour trouver les étapes connectées entre elles. Les trois règles gèrent différents cas de connexion entre les étapes et les tronçons. La requête `parcoursImpair('Orleans', 'Fleury-les-Aubrais')` permet de vérifier s'il existe un parcours impair entre Orléans et Fleury-les-Aubrais.
5. Ce programme identifie les circuits, c'est-à-dire les parcours qui reviennent à leur point de départ après avoir visité plusieurs étapes. La première règle vérifie si deux étapes sont connexes dans les deux sens, formant ainsi un circuit. La deuxième règle utilise la récursivité pour vérifier si un chemin de l'étape de départ à une autre étape peut être complété par un chemin de cette dernière à l'étape de départ, formant ainsi un circuit. La condition `X != Z` évite de sélectionner des chemins qui se terminent au point de départ.

### Conclusion

En résumé, nous avons réussi à modéliser et à implémenter une base de données pour gérer les informations sur les parcours de randonnée, en utilisant des approches relationnelles, objet-relationnelles et logiques. Ces différentes méthodes offrent des possibilités variées pour interroger et manipuler les données en fonction des besoins spécifiques des utilisateurs.

### Annexe

Relationnel :

```
DROP TABLE NIVEAU_CONFORT CASCADE CONSTRAINTS;
DROP TABLE TYPE_SOL CASCADE CONSTRAINTS;
DROP TABLE ETAPE CASCADE CONSTRAINTS;
DROP TABLE TRONCON CASCADE CONSTRAINTS;

CREATE TABLE NIVEAU_CONFORT (
   idNiveauConfort NUMBER PRIMARY KEY,
   nomNiveauConfort VARCHAR2(255) NOT NULL,
   descriptionNiveauConfort VARCHAR2(255) NOT NULL
```

```
);

CREATE TABLE TYPE_SOL (
   idTypeSol NUMBER PRIMARY KEY,
   nomTypeSol VARCHAR2(255) NOT NULL
);

CREATE TABLE ETAPE (
   codeEtape NUMBER PRIMARY KEY,
   nomEtape VARCHAR2(255) NOT NULL,
   latitude NUMBER NOT NULL,
   longitude NUMBER NOT NULL,
   idNiveauConfort NUMBER NOT NULL,
   FOREIGN KEY(idNiveauConfort) REFERENCES NIVEAU_CONFORT(idNiveauConfort)
);

CREATE TABLE TRONCON (
   idTroncon NUMBER PRIMARY KEY,
   nomTroncon VARCHAR2(255) NOT NULL,
   distance NUMBER NOT NULL,
   denivele NUMBER NOT NULL,
   idTypeSol NUMBER NOT NULL,
   idEtapeDepart NUMBER NOT NULL,
   idEtapeArrivee NUMBER NOT NULL,
   FOREIGN KEY(idTypeSol) REFERENCES TYPE_SOL(idTypeSol),
   FOREIGN KEY(idEtapeDepart) REFERENCES ETAPE(codeEtape),
   FOREIGN KEY(idEtapeArrivee) REFERENCES ETAPE(codeEtape)
);

INSERT INTO NIVEAU_CONFORT (idNiveauConfort, nomNiveauConfort, descriptionNiveauConfort) VALUES (1,
'Ville', 'Étape correspond à une ville');
INSERT INTO NIVEAU_CONFORT (idNiveauConfort, nomNiveauConfort, descriptionNiveauConfort) VALUES (2,
'Refuge', 'Étape correspond à un refuge');
INSERT INTO NIVEAU_CONFORT (idNiveauConfort, nomNiveauConfort, descriptionNiveauConfort) VALUES (3,
'Camping', 'Étape correspond à un emplacement de camping');
INSERT INTO NIVEAU_CONFORT (idNiveauConfort, nomNiveauConfort, descriptionNiveauConfort) VALUES (4,
'Montagne', 'Étape correspond à un endroit en montagne');
INSERT INTO NIVEAU_CONFORT (idNiveauConfort, nomNiveauConfort, descriptionNiveauConfort) VALUES (5,
'Plage', 'Étape correspond à une plage');
INSERT INTO NIVEAU_CONFORT (idNiveauConfort, nomNiveauConfort, descriptionNiveauConfort) VALUES (6,
'Forêt', 'Étape correspond à une forêt');
INSERT INTO NIVEAU_CONFORT (idNiveauConfort, nomNiveauConfort, descriptionNiveauConfort) VALUES (7,
'Campagne', 'Étape correspond à la campagne');
INSERT INTO NIVEAU_CONFORT (idNiveauConfort, nomNiveauConfort, descriptionNiveauConfort) VALUES (8,
'Désert', 'Étape correspond à un désert');
INSERT INTO NIVEAU_CONFORT (idNiveauConfort, nomNiveauConfort, descriptionNiveauConfort) VALUES (9,
'Zone urbaine', 'Étape correspond à une zone urbaine');
INSERT INTO NIVEAU_CONFORT (idNiveauConfort, nomNiveauConfort, descriptionNiveauConfort) VALUES (10,
'Zone rurale', 'Étape correspond à une zone rurale');

INSERT INTO TYPE_SOL (idTypeSol, nomTypeSol) VALUES (1, 'Terre');
INSERT INTO TYPE_SOL (idTypeSol, nomTypeSol) VALUES (2, 'Bitume');
INSERT INTO TYPE_SOL (idTypeSol, nomTypeSol) VALUES (3, 'Gravier');
INSERT INTO TYPE_SOL (idTypeSol, nomTypeSol) VALUES (4, 'Sable');
INSERT INTO TYPE_SOL (idTypeSol, nomTypeSol) VALUES (5, 'Pierre');
INSERT INTO TYPE_SOL (idTypeSol, nomTypeSol) VALUES (6, 'Asphalte');
INSERT INTO TYPE_SOL (idTypeSol, nomTypeSol) VALUES (7, 'Neige');
INSERT INTO TYPE_SOL (idTypeSol, nomTypeSol) VALUES (8, 'Glace');
INSERT INTO TYPE_SOL (idTypeSol, nomTypeSol) VALUES (9, 'Boue');
INSERT INTO TYPE_SOL (idTypeSol, nomTypeSol) VALUES (10, 'Eau');

INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (1, 'Paris', 48.8566,
2.3522, 1);
```

```
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (2, 'Lyon', 45.75, 4.85, 1);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (3, 'Marseille', 43.2965, 5.3698, 1);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (4, 'Toulouse', 43.6045, 1.4442, 1);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (5, 'Nice', 43.7102, 7.2620, 1);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (6, 'Nantes', 47.2184, 1.5536, 1);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (7, 'Strasbourg', 48.5734, 7.7521, 1);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (8, 'Montpellier', 43.6109, 3.8772, 1);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (9, 'Bordeaux', 44.8378, 0.5792, 1);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (10, 'Lille', 50.6292, 3.0573, 1);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (11, 'Orléans', 47.9029, 1.9099, 1);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (12, 'Refuge du Mont Blanc', 45.8325, 6.8655, 2);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (13, 'Camping Les Pins', 43.4211, 4.4526, 3);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (14, 'Chalet en Montagne', 46.1171, 6.0374, 4);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (15, 'Plage de Saint-Tropez', 43.2700, 6.6400, 5);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (16, 'Forêt de Fontainebleau', 48.4042, 2.5989, 6);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (17, 'Campagne Normande', 49.1104, 0.9947, 7);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (18, 'Désert des Bardenas', 42.1500, -1.4000, 8);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (19, 'Quartier de La Défense', 48.8923, 2.2384, 9);
INSERT INTO ETAPE (codeEtape, nomEtape, latitude, longitude, idNiveauConfort) VALUES (20, 'Champs de Blé', 47.7295, 3.5923, 10);

INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee) VALUES (1, 'Paris-Lyon', 465, 0, 2, 1, 2);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee) VALUES (2, 'Lyon-Marseille', 314, 0, 2, 2, 3);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee) VALUES (3, 'Marseille-Toulouse', 403, 0, 2, 3, 4);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee) VALUES (4, 'Toulouse-Nice', 471, 0, 2, 4, 5);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee) VALUES (5, 'Nice-Nantes', 1025, 0, 2, 5, 6);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee) VALUES (6, 'Nantes-Strasbourg', 785, 0, 2, 6, 7);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee) VALUES (7, 'Strasbourg-Montpellier', 800, 0, 2, 7, 8);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee) VALUES (8, 'Montpellier-Bordeaux', 470, 0, 2, 8, 9);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee) VALUES (9, 'Bordeaux-Lille', 700, 0, 2, 9, 10);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee) VALUES (10, 'Lille-Orléans', 400, 0, 2, 10, 11);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee) VALUES (11, 'Orléans-Refuge du Mont Blanc', 600, 0, 2, 11, 12);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee) VALUES (12, 'Refuge du Mont Blanc-Camping Les Pins', 200, 0, 2, 12, 13);
```

```
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee)
VALUES (13, 'Camping Les Pins-Chalet en Montagne', 300, 0, 2, 13, 14);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee)
VALUES (14, 'Chalet en Montagne-Plage de Saint-Tropez', 500, 0, 2, 14, 15);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee)
VALUES (15, 'Plage de Saint-Tropez-Forêt de Fontainebleau', 400, 0, 2, 15, 16);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee)
VALUES (16, 'Forêt de Fontainebleau-Campagne Normande', 300, 0, 2, 16, 17);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee)
VALUES (17, 'Campagne Normande-Désert des Bardenas', 600, 0, 2, 17, 18);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee)
VALUES (18, 'Désert des Bardenas-Quartier de La Défense', 800, 0, 2, 18, 19);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee)
VALUES (19, 'Quartier de La Défense-Champs de Blé', 500, 0, 2, 19, 20);
INSERT INTO TRONCON (idTroncon, nomTroncon, distance, denivele, idTypeSol, idEtapeDepart, idEtapeArrivee)
VALUES (20, 'Orléans-Champs de Blé', 1000, 0, 2, 11, 20);

-- Donner les noms des étapes que nous pouvons atteindre directement en suivant les tronçons qui partent d’Orléans.
-- Rep : Champs de Blé et Refuge du Mont Blanc
SELECT DISTINCT e.nomEtape
FROM ETAPE e
JOIN TRONCON t ON e.codeEtape = t.idEtapeArrivee
WHERE t.idEtapeDepart = (SELECT codeEtape FROM ETAPE WHERE nomEtape = 'Orléans');

-- Veuillez fournir la liste des étapes accessibles depuis Orléans avec une seule étape intermédiaire.
-- Rep : Camping Les Pins
SELECT DISTINCT e3.nomEtape
FROM ETAPE e1
JOIN TRONCON t1 ON e1.codeEtape = t1.idEtapeDepart
JOIN ETAPE e2 ON (e2.codeEtape = t1.idEtapeArrivee)
JOIN TRONCON t2 ON e2.codeEtape = t2.idEtapeDepart
JOIN ETAPE e3 ON (e3.codeEtape = t2.idEtapeArrivee)
WHERE e1.nomEtape = 'Orléans'
AND e3.nomEtape != 'Orléans'
AND (e1.codeEtape != e3.codeEtape);

-- Veuillez fournir la liste des étapes accessibles depuis Orléans, avec un nombre quelconque d’étapes intermédiaires
-- Rep : Champs de Blé, Refuge du Mont Blanc, Camping Les Pins, Chalet en Montagne, Plage de Saint-Tropez, Forêt de Fontainebleau, Campagne Normande, Désert des Bardenas, Quartier de La Défense

SELECT DISTINCT nomEtape
FROM (
   SELECT e.nomEtape, CONNECT_BY_ROOT e_orleans.nomEtape AS depart, LEVEL AS niveau
   FROM TRONCON t
   JOIN ETAPE e ON t.idEtapeArrivee = e.codeEtape
   JOIN ETAPE e_orleans ON t.idEtapeDepart = e_orleans.codeEtape
   START WITH e_orleans.nomEtape = 'Orléans'
   CONNECT BY NOCYCLE PRIOR t.idEtapeArrivee = t.idEtapeDepart
)
WHERE nomEtape != 'Orléans';

Objet-Relationnel :

DROP TABLE Troncon;
DROP TYPE ListePointInteret;
DROP TYPE ListeIndicesQualite;
DROP TYPE IndiceQualite;
DROP TYPE PointInteret;

CREATE TYPE PointInteret AS OBJECT (
   nom VARCHAR2(100),
   type VARCHAR2(50)
);
```

```
/
CREATE TYPE IndiceQualite AS OBJECT (
   nom VARCHAR2(100),
   poidsActuel NUMBER(5, 2),
   valeur NUMBER(5, 2)
);
/
CREATE TYPE ListeIndicesQualite AS TABLE OF IndiceQualite;
/
CREATE TYPE ListePointInteret AS TABLE OF PointInteret;
/
CREATE TABLE Troncon (
   idTroncon VARCHAR2(10),
   nomUsuel VARCHAR2(100),
   pointDep VARCHAR2(100),
   pointArr VARCHAR2(100),
   distance NUMBER(10, 2),
   pointsInteret ListePointInteret,
   indicesQualite ListeIndicesQualite
) NESTED TABLE indicesQualite STORE AS indicesQualite_tab
NESTED TABLE pointsInteret STORE AS pointsInteret_tab;

-- Insertion d'un tronçon avec des points d'intérêt et des indices de qualité
INSERT INTO Troncon (idTroncon, nomUsuel, pointDep, pointArr, distance, pointsInteret, indicesQualite)
VALUES ('t347', 'rue des Roses', 'Orleans', 'Fleury-les-Aubrais', 3.5,
       ListePointInteret(PointInteret('Parc Floral', 'parc')),
       ListeIndicesQualite(IndiceQualite('difficulte', 2.0, 1.5), IndiceQualite('securite', 4.2, 3.0)));

-- Insertion d'un autre tronçon avec des points d'intérêt et des indices de qualité
INSERT INTO Troncon (idTroncon, nomUsuel, pointDep, pointArr, distance, pointsInteret, indicesQualite)
VALUES ('t348', 'rue de la République', 'Fleury-les-Aubrais', 'Orleans', 2.8,
       ListePointInteret(PointInteret('Théâtre Municipal', 'théâtre')),
       ListeIndicesQualite(IndiceQualite('difficulte', 1.8, 1.2), IndiceQualite('securite', 4.7, 2.5)));

-- Insertion d'un tronçon sans points d'intérêt ni indices de qualité
INSERT INTO Troncon (idTroncon, nomUsuel, pointDep, pointArr, distance, pointsInteret, indicesQualite)
VALUES ('t349', 'rue du Pont', 'Orleans', 'Saran', 6.2, NULL, NULL);

-- Insertion d'un tronçon avec plusieurs points d'intérêt et des indices de qualité
INSERT INTO Troncon (idTroncon, nomUsuel, pointDep, pointArr, distance, pointsInteret, indicesQualite)
VALUES ('t350', 'avenue de la Liberté', 'Orleans', 'Saint-Jean-de-Braye', 4.5,
       ListePointInteret(PointInteret('Musée des Beaux-Arts', 'musée'), PointInteret('Bibliothèque Municipale',
'bibliothèque')),
       ListeIndicesQualite(IndiceQualite('difficulte', 2.5, 1.8), IndiceQualite('securite', 4.5, 2.9)));

-- Insertion d'un autre tronçon avec plusieurs points d'intérêt et des indices de qualité
INSERT INTO Troncon (idTroncon, nomUsuel, pointDep, pointArr, distance, pointsInteret, indicesQualite)
VALUES ('t351', 'rue de la Paix', 'Orleans', 'La Chapelle-Saint-Mesmin', 3.2,
       ListePointInteret(PointInteret('Centre Commercial', 'centre commercial'), PointInteret('Cinéma', 'cinéma'),
PointInteret('Piscine Municipale', 'piscine'), PointInteret('Piscine Communautaire', 'piscine')),
       ListeIndicesQualite(IndiceQualite('difficulte', 1.7, 1.1), IndiceQualite('securite', 4.3, 2.7)));

-- Requete 1
SELECT t.idTroncon, pi.type, COUNT(pi.nom) AS nombre_points_interet
FROM Troncon t, TABLE(t.pointsInteret) pi
GROUP BY t.idTroncon, pi.type;

-- Requete 2
SELECT
   t.idTroncon,
   iq.nom AS nom_indice,
   (iq.poidsActuel * iq.valeur) AS impact
FROM
```

```
    Troncon t,
    TABLE(t.indicesQualite) iq;
```

Logique :

```
-- Identifiez quels sont les prédicats extensionnels de votre base de données déductive
etape(depart, arrivee).
troncon(depart, arrivee, distance, enTerre).
pointInteret(nom, type).
indiceQualite(nom, poidsActuel, valeur).

etape('Orleans', 'Fleury-les-Aubrais').
etape('Fleury-les-Aubrais', 'Saran').
etape('Orleans', 'La Chapelle-Saint-Mesmin').

troncon('Orleans', 'Fleury-les-Aubrais', 52).
troncon('Fleury-les-Aubrais', 'Saran', 38).
troncon('Orleans', 'La Chapelle-Saint-Mesmin', 75).

pointInteret('Parc Floral', 'parc').
pointInteret('Musée des Beaux-Arts', 'musée').
pointInteret('Cinéma', 'cinéma').

indiceQualite('difficulte', 20, 35).
indiceQualite('securite', 42, 48).
indiceQualite('difficulte', 18, 27).
indiceQualite('securite', 47, 51).

-- Q2
etapesConnexes(depart, arrivee).
etapesConnexes(X, Y) :- troncon(X, Y, _).
etapesConnexes(X, Y) :- troncon(X, Z, _), etapesConnexes(Z, Y).

etapesConnexes('Orleans', 'Fleury-les-Aubrais') ?

-- Q3

parcoursEnTerre(X, Y, DistanceTotale) :-
    troncon(X, Y, enTerre, Distance),
    enTerre = 'terre',
    DistanceTotale = Distance.
parcoursEnTerre(X, Z, DistanceTotale) :-
    troncon(X, Y, enTerre, Distance),
    enTerre = 'terre',
    parcoursEnTerre(Y, Z, DistanceRestante),
    DistanceTotale = Distance + DistanceRestante.

parcoursEnTerre('Orleans', 'Fleury-les-Aubrais', DistanceTotale) ?

-- Q4
parcoursImpair(X, Y) :- etapesConnexes(X, Y), not troncon(X, Y).
parcoursImpair(X, Y) :- troncon(X, Z), etapesConnexes(Z, Y), not troncon(X, Y), not troncon(Y, Z).
parcoursImpair(X, Y) :- troncon(X, Z), etapesConnexes(Z, Y), troncon(X, Y), not troncon(Y, Z).

parcoursImpair('Orleans', 'Fleury-les-Aubrais') ?

-- Q5
circuit(X, Y) :- etapesConnexes(X, Y), etapesConnexes(Y, X).
circuit(X, Z) :- etapesConnexes(X, Y), circuit(Y, Z), X != Z.
```