SQL : Structured Query Language

- Langage de requête sur les bases de données relationnelles
- 4 opérations principales
  - SELECT : rechercher
    - dans 1 ou plusieurs tables
    - des enregistrement possédant
    - une ou plusieurs propriétés
  - INSERT / DELETE / UPDATE : ajouter / supprimer / modifier des enregistrement
- Opérations supplémentaires pour
  - Créer / détruire des tables
  - Gérer les droits d'accès etc ...

Créer la base de données

• **CREATE DATABASE** pour créer une base de données.

– CREATE DATABASE nom_base
OWNER = nom_proprio
…
;

– http://www.postgresqltutorial.com/postgresql-create-database/

• **ALTER DATABASE** pour modifier ses propriétés.

– ALTER DATABASE nom_base
RENAME TO OWNER = nouveau_nom_base
OWNER TO nouveau_proprio
…
;

– http://www.postgresqltutorial.com/postgresql-alter-database/

Insérer des données dans une base

• **CREATE TABLE** pour créer une table vide.

– CREATE TABLE nom_table (
colonne1 type,
colonne2 type,
…
);

– http://www.postgresqltutorial.com/postgresql-create-table/

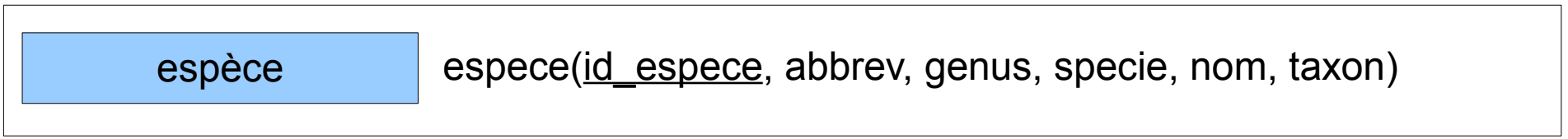
• Les (principaux) types dans PostgreSQL :

- **text**, **char (n)** pour les chaînes
- **integer**, **bigint** pour les entiers
- **real**, **double precision** pour les nombres « flottants »
- **numeric(p,s)** pour les nombres exactes (p = nombre total de chiffres, s = nombres de chiffres après la virgule)
- …

- http://docs.postgresql.fr/current/datatype.html

Insérer des données dans une base

```
– CREATE TABLE espece (
id_espace integer PRIMARY KEY,
Abbrev text NOT NULL,
Genus text,
Specie text,
Nom text,
Taxon text
);
```



Insérer des données dans une base

- **INSERT INTO** pour ajouter des lignes dans une table. Exemples:

– INSERT INTO nom_table (colonne1, colonne2, …) VALUES (valeur1, valeur2, …);

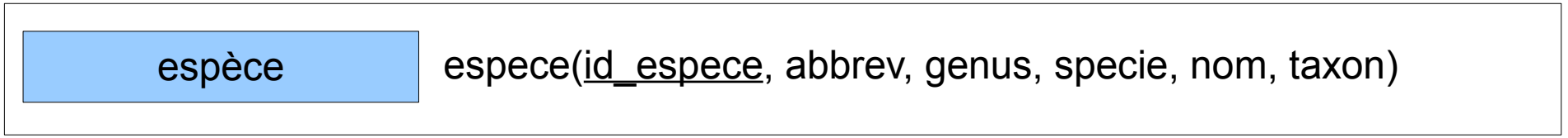
– INSERT INTO nom_table VALUES (valeur1, valeur2, …, valeurN);

!!! respecter le nombre et l'ordre des colonnes !!!

– http://www.postgresqltutorial.com/postgresql-insert/

- **INSERT INTO** espece (id_espece, abbrev, genus, specie, taxon)
  VALUES (1, 'bidop' , 'arabidopsis', 'silene', '1234') ;

– INSERT INTO espece (1, 'bidop', 'arabidopsis', 'silene', 'arabette', '1234');



Opération de sélection

- Sélectionner des enregistrement dans une ou plusieurs tables
- Forme

  SELECT <liste d'attributs>

  FROM <liste de tables>

  [WHERE <liste de conditions/jointures>]

  [GROUP BY <liste d'attributs>]

  [ORDER BY <liste d'attributs>]

- Table = nom_de_table
- Attribut = nom_de_table.nom_d'attribut
- Liste = éléments séparés par des ,

Sélectionner les colonnes

SELECT NOM FROM ESPECE

SELECT col FROM table
Extraire 1 colonne pour toutes
Les lignes de la table

SELECT GENUS, SPECIE FROM SPECE

SELECT colA, colB FROM table
Extraire 2 colonnes

SELECT NOM, ABBREV, TAXON FROM ESPECE

SELECT colD, colA, colB from table
Ordre d'extraction # ordre de la table

SELECT * FROM ESPECE

SELECT * FROM table
Extraire toutes les colonnes

Attention : SQL nom des tables et des colonnes = minuscules ou majuscules

espèce

espece(id_espece, abbrev, genus, specie, nom, taxon)

Trier les sorties : ORDER BY

```
SELECT NOM, ABBREV, TAXON FROM ESPECE
ORDER BY NOM
```
Tri sur les valeurs de la colonne nom

```
SELECT NOM, ABBREV, TAXON FROM ESPECE
ORDER BY NOM ASC
```
Tri sur les valeurs de la colonne nom
Par ordre croissant (option par défaut)
(décroissant = DESC)

```
SELECT NOM, ABBREV, TAXON FROM ESPECE
ORDER BY GENUS
```
Tri sur les valeurs une colonne non affichée

```
SELECT NOM, ABBREV, TAXON FROM ESPECE
ORDER BY GENUS, SPECIE
```
Tri sur les valeurs de plusieurs colonnes

espèce

espece(id_espece, abbrev, genus, specie, nom, taxon)

Sélectionner les lignes (WHERE)

```
SELECT NOM, ABBREV, TAXON FROM ESPECE
WHERE NOM = 'Arabette'
```

```
SELECT colA, colB, colC from table
WHERE colA = valeurA ;
```

Extraire uniquement les lignes dont 1 colonne a une valeur précise

```
SELECT NOM, ABBREV, TAXON FROM ESPECE
WHERE GENUS = 'Arabidopsis'
```

```
SELECT colA, colB, colC from table
WHERE colD = valeurD ;
```

Colonne de sélection différente des colonnes affichées

Opérateurs de comparaison : <, <=, >, >=, =, <>

Attention : SQL constantes entre ' ou " selon les logiciels, PostgreSQL ' autorisé

espèce

espece(id_espece, abbrev, genus, specie, nom, taxon)

Sélectionner à partir d'un modèle de valeur

```
SELECT NOM, ABBREV, TAXON FROM ESPECE
WHERE GENUS = 'Arabidopsis'
```

```
SELECT colA, colB, colC from table
WHERE colD = valeurD ;
```

Extraire uniquement les lignes dont 1 colonne a une valeur précise

```
SELECT NOM, ABBREV, TAXON FROM ESPECE
WHERE GENUS LIKE 'Ara%'
```

```
SELECT colA, colB, colC from table
WHERE colD LIKE 'MODELED';
```

Extraire uniquement les lignes dont 1 colonne a une valeur ressemblant à un modèle

modèle = chaîne dans laquelle % représente n'importe quelle sous-chaîne de caractères

espèce

espece(id_espece, abbrev, genus, specie, nom, taxon)

Sélectionner à partir d'une plage ou d'une liste de valeurs

```
SELECT NOM, ABBREV, TAXON FROM ESPECE
WHERE TAXON BETWEEN 1000  AND 100000
```

```
SELECT colA, colB, colC from table
 WHERE colD BETWEEN val1 AND val2';
```

Extraire uniquement les lignes dont la valeur d'une colonne est dans une plage

```
SELECT NOM, ABBREV, TAXON FROM ESPECE
WHERE GENUS IN ('Beta', 'Silene')
```

```
SELECT colA, colB, colC from table
  WHERE colD IN ['val1', 'val2', ...];
```

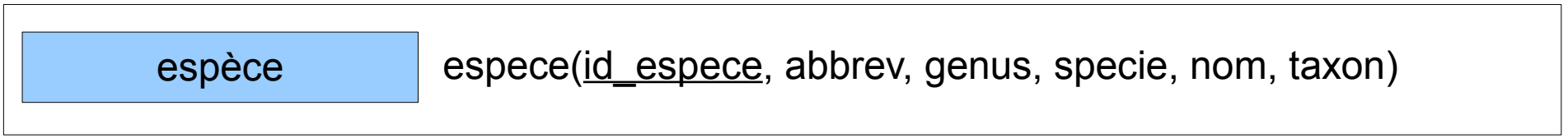
Extraire uniquement les lignes dont la valeur d'une colonne est dans une liste

```
SELECT NOM, ABBREV, TAXON FROM ESPECE
WHERE GENUS NOT IN ('Beta', 'Silene')
```

```
SELECT colA, colB, colC from table
 WHERE colD NOT IN ['val1', 'val2', ...];
```

Extraire uniquement les lignes dont la valeur d'une colonne n'est pas dans la liste

ATTENTION SQL : ['val1', 'val2'] # PostgreSQL : ('val1', 'val2')



Combinaison d'opérateurs (AND, OR)

```
SELECT NOM, ABBREV, TAXON FROM ESPECE
WHERE GENUS = 'Arabidopsis'
AND SPECIE = 'thaliana'
```

```
SELECT colA, colB, colC from table
WHERE colD = valeurD
AND colE = valeurE ;
```

Les 2 critères doivent être vérifiés

```
SELECT NOM, ABBREV, TAXON FROM ESPECE
WHERE GENUS = 'Arabidopsis'
OR GENUS = 'Silene'
```

```
SELECT colA, colB, colC from table
WHERE colD = valeurD
OR colD = valeurD2 ;
```

L'un des 2 critères doit être vérifié
Exemple avec 1 colonne

```
SELECT NOM, ABBREV, TAXON FROM ESPECE
WHERE GENUS = 'Arabidopsis'
OR TAXON = '1234'
```

```
SELECT colA, colB, colC from table
WHERE colD = valeurD
OR colE = valeurE ;
```

L'un des 2 critères doit être vérifié
Exemple avec 2 colonnes

espèce

espece(id_espece, abbrev, genus, specie, nom, taxon)

Sélectionner les valeurs distinctes

SELECT GENUS FROM ESPECE

SELECT colA FROM table

Extraire la colonne genus pour toutes les lignes
! il peut y avoir plusieurs fois le même genre

SELECT DISTINCT(GENUS) FROM ESPECE

SELECT DISTINCT(colA) FROM table

Affiche une seule fois chaque valeur
Donc une seul fois chaque genre
Même si la table comporte plusieurs lignes pour un même genre
(avec un espèce différente)

espèce

espece(id_espece, abbrev, genus, specie, nom, taxon)

Fonctions d'agrégation : COUNT, SUM, AVG, MIN, MAX

SELECT COUNT(NOM) FROM ESPECE
Nombre *de lignes dans la table*

SELECT SUM(N_SSP) FROM ESPECE
Total des sous-espèces

SELECT AVG(N_SSP) FROM ESPECE
Moyenne du nombre de sous-espèces (sur l'ensemble des espèces dans la table)

SELECT MAX(N_SSP) FROM ESPECE
Nombre maximum de sous-espèces (parmi toutes les espèces dans la table)

SELECT COUNT(DISTINCT(GENUS)) FROM ESPECE
Combinaison de fonctions

espèce

espece(id_espece, abbrev, genus, specie, nom, taxon, number_of_sub_species)

Fonctions d'agrégation + critères de sélection

SELECT COUNT(NOM) FROM ESPECE
WHERE GENUS = 'Silene'
Nombre *d'espèces du genre 'Silene' (ie nombre de lignes dont la colonne GENUS = Silene)*

SELECT SUM(N_SSP) FROM ESPECE
WHERE GENUS = 'Silene'
Total des sous-espèces pour le genre Silene (ie total des colonnes N_SSP pour les lignes dont la colonne GENUS = Silene)

SELECT AVG(N_SSP) FROM ESPECE
WHERE GENUS = 'Silene'
Moyenne du nombre de sous-espèces uniquement sur les espèces du genre Silene

SELECT MAX(N_SSP) FROM ESPECE
WHERE GENUS = 'Silene' OR GENUS = 'Beta'
Nombre maximum de sous-espèces par espèce dans les genres Silene ou Beta

espèce

espece(id_espece, abbrev, genus, specie, nom, taxon, number_of_sub_species)

Fonctions d'agrégation + GROUP BY

SELECT GENUS, SUM(N_SSP) FROM ESPECE
GROUP BY GENUS

SELECT GENUS, MAX(N_SSP) FROM ESPECE
GROUP BY GENUS

SELECT GENUS, MIN(N_SSP), MAX(N_SSP), AVG(N_SSP)
FROM ESPECE
GROUP BY GENUS

espèce

espece(id_espece, abbrev, genus, specie, nom, taxon, number_of_sub_species)

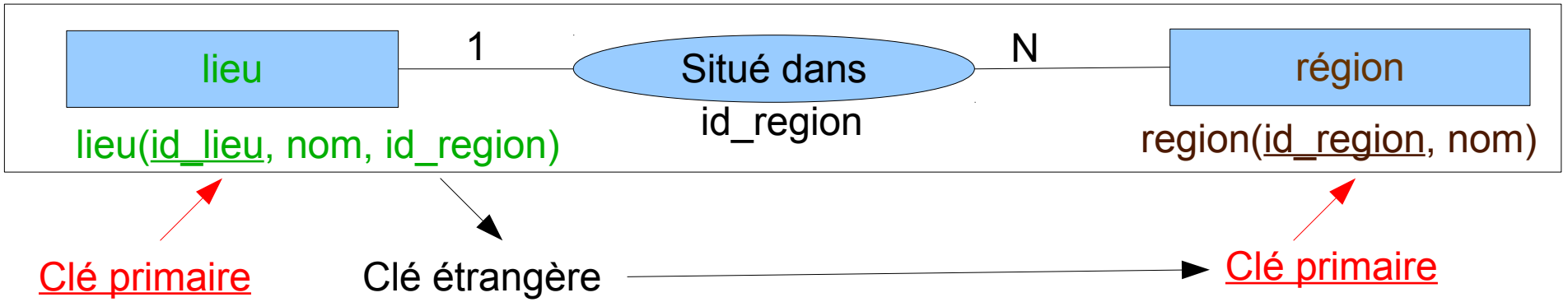
Sélection : jointure sur 2 tables : relation 1/N

SELECT LIEU.NOM, REGION.NOM FROM LIEU, REGION
WHERE LIEU.ID_REGION = REGION.ID_REGION

Jointure + affichage de colonnes intéressantes
ie les noms (les id_ ne sont pas affichés)
Chaque nom de colonne est préfixé par le nom de la table

SELECT L.NOM, R.NOM FROM LIEU L, REGION R
WHERE L.ID_REGION = R.ID_REGION

Possibilité de renommer les tables pour la requête



Sélection : jointure sur 3 tables : relation N/M

SELECT E.NOM, L.NOM FROM ESPECE E, LIEU L, OBSERVATION O
WHERE E.ID_ESPECE = O.ID_ESPECE AND O.ID_LIEU = L.ID_LIEU

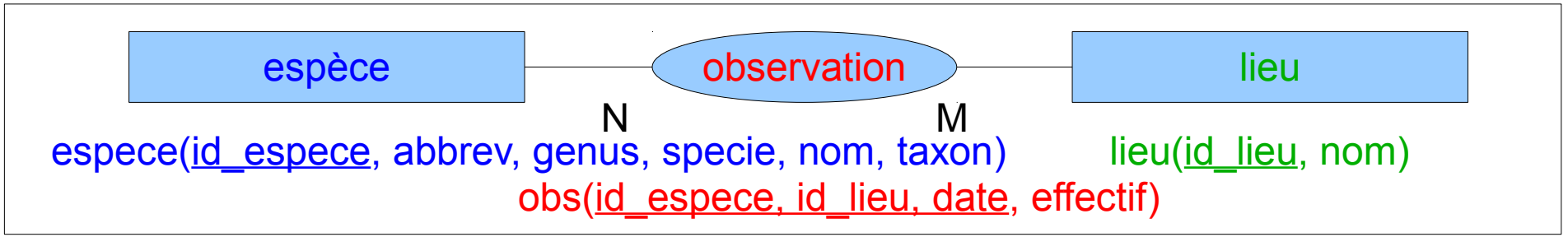
Jointure + affichage de colonnes de 2 des 3 tables
Ordre des critères indifférent

SELECT E.NOM, L.NOM, O.DATE, O.EFFECTIF FROM ESPECE E, LIEU L,
OBSERVATION O WHERE E.ID_ESPECE = O.ID_ESPECE AND O.ID_LIEU = L.ID_LIEU

Jointure + affichage de colonnes des 3 tables

SELECT E.NOM, L.NOM, O.DATE, O.EFFECTIF
FROM ESPECE E LIEU L, OBSERVATION O
WHERE E.ID_ESPECE = O.ID_ESPECE
AND O.ID_LIEU = L.ID_LIEU

- *Retour à la ligne et espacement indifférent pour le SGBD*
- *Utiliser pour rendre les requêtes + lisibles*



Sélection : jointure sur 4 tables

```
SELECT E.NOM,  L.NOM, R.NOM, O.DATE, O.EFFECTIF
FROM ESPECE E, OBSERVATION O,  LIEU L, REGION R
WHERE E.ID_ESPECE = O.ID_ESPECE
AND O.ID_LIEU = L.ID_LIEU
AND L.ID_REGION = R.ID_REGION
```

espece(id_espece, abbrev, genus, specie, nom, taxon)

obs(id_espece, id_lieu, date, effectif)

lieu(id_lieu, id_region, nom)

region(id_region, nom)

SQL & les dates

Fonctions sur les dates

Fonction	Description
CURDATE()	date courante
CURTIME()	heure courante
DATEDIFF(chaîne, dateheure1, dateheure2)	nombre d'unités de temps écoulée entre les 2. 'chaîne' = unité de temps ms=millisecondes, ss=secondes, mi=minute, hh=heure, dd=jour, mm=mois, yy=année
DAYNAME(date)	nom du jour (en Anglais)
DAYOFMONTH(date)	jour du mois (1-31)
DAYOFWEEK(date)	jour de la semaine (1 = Dimanche)
DAYOFYEAR(date)	jour de l'année (1-366)
HOUR(time)	heure (0-23)
MINUTE(time)	minutes (0-59)
MONTH(date)	mois (1-12)
MONTHNAME(date)	nom du mois (en Anglais)
QUARTER(date)	trimestre (1-4)
SECOND(time)	secondes (0-59)
WEEK(date)	semaine
YEAR(date)	année
CURRENT_DATE	date courante
CURRENT_TIME	heure courante
CURRENT_TIMESTAMP ou now()	un horodatage (format date et heure)

Exemples

```
SELECT MONTH(date) FROM obs
```

```
SELECT ...
GROUP BY (month(date))
```

```
SELECT OBS.DATE, CURRENT_DATE, DATEDIFF('yy',
OBS.DATE, CURRENT_DATE)
FROM OBS
WHERE DATEDIFF('yy', OBS.DATE, CURRENT_DATE) > 1
```

En résumé

SQL cheat sheet

REBELLABS by ZEROTURNAROUND

Basic Queries

```
-- filter your columns
   SELECT col1, col2, col3, ... FROM table1
-- filter the rows
   WHERE col4 = 1 AND col5 = 2
-- aggregate the data
   GROUP by ...
-- limit aggregated data
   HAVING count(*) > 1
-- order of the results
   ORDER BY col2
```

Useful keywords for **SELECTS**:

DISTINCT - return unique results
BETWEEN a **AND** b - limit the range, the values can be numbers, text, or dates
LIKE - pattern search within the column text
IN (a, b, c) - check if the value is contained among given.

Data Modification

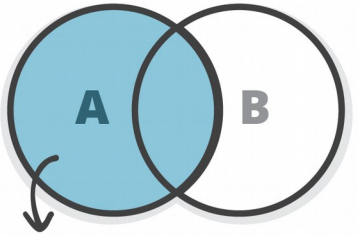
```
-- update specific data with the WHERE clause
   UPDATE table1 SET col1 = 1 WHERE col2 = 2
-- insert values manually
   INSERT INTO table1 (ID, FIRST_NAME, LAST_NAME)
      VALUES (1, 'Rebel', 'Labs');
-- or by using the results of a query
   INSERT INTO table1 (ID, FIRST_NAME, LAST_NAME)
      SELECT id, last_name, first_name FROM table2
```

Views

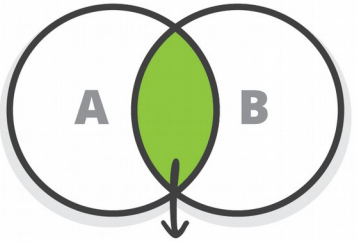
A **VIEW** is a virtual table, which is a result of a query.
They can be used to create virtual tables of complex queries.

```
CREATE VIEW view1 AS
SELECT col1, col2
FROM table1
WHERE ...
```

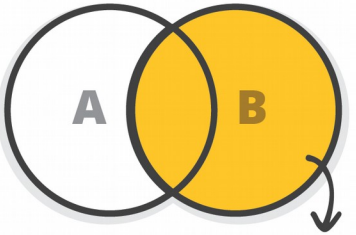
The Joy of JOINs



LEFT OUTER JOIN - *all rows from table A, even if they do not exist in table B*



INNER JOIN - *fetch the results that exist in both tables*



RIGHT OUTER JOIN - *all rows from table B, even if they do not exist in table A*

Updates on JOINed Queries

You can use **JOIN**s in your **UPDATE**s

```
UPDATE t1 SET a = 1
FROM table1 t1 JOIN table2 t2 ON t1.id = t2.t1_id
WHERE t1.col1 = 0 AND t2.col2 IS NULL;
```

NB! Use database specific syntax, it might be faster!

Semi JOINs

You can use subqueries instead of **JOIN**s:

```
SELECT col1, col2 FROM table1 WHERE id IN
   (SELECT t1_id FROM table2 WHERE date >
      CURRENT_TIMESTAMP)
```

Indexes

If you query by a column, index it!

```
CREATE INDEX index1 ON table1 (col1)
```

Don't forget:
Avoid overlapping indexes
Avoid indexing on too many columns
Indexes can speed up **DELETE** and **UPDATE** operations

Useful Utility Functions

```
-- convert strings to dates:
   TO_DATE (Oracle, PostgreSQL), STR_TO_DATE (MySQL)
-- return the first non-NULL argument:
   COALESCE (col1, col2, "default value")
-- return current time:
   CURRENT_TIMESTAMP
-- compute set operations on two result sets
   SELECT col1, col2 FROM table1
   UNION / EXCEPT / INTERSECT
   SELECT col3, col4 FROM table2;
```

Union - returns data from both queries
Except - rows from the first query that are not present in the second query
Intersect - rows that are returned from both queries

Reporting

Use aggregation functions

COUNT - return the number of rows
SUM - cumulate the values
AVG - return the average for the group
MIN / MAX - smallest / largest value

APPLICATIONS

1 – Étude des espaces naturels

Espaces naturels

Sur une carte régionale, on a représenté des zones correspondant aux différents Parcs Naturels Régionaux (PNR), Parcs Naturels Transfrontaliers (PNT) et Espaces Naturels Régionaux (ENR) des Hauts-De-France. Chacune de ces zones est repérée par un numéro à 2 chiffres. On souhaite associer des informations complémentaires à ces zones :

- nom de la zone (PNR des Caps et Marais d'Opale, ENR Lille Métropole…)

- type (PNR, PNT, EN)

- liste de villes se trouvant sur le territoire de la zone

- liste des organismes prenant part à la gestion du parc

Voici les zones se trouvant sur votre carte (les listes de villes et d'organismes sont volontairement non exhaustives) :

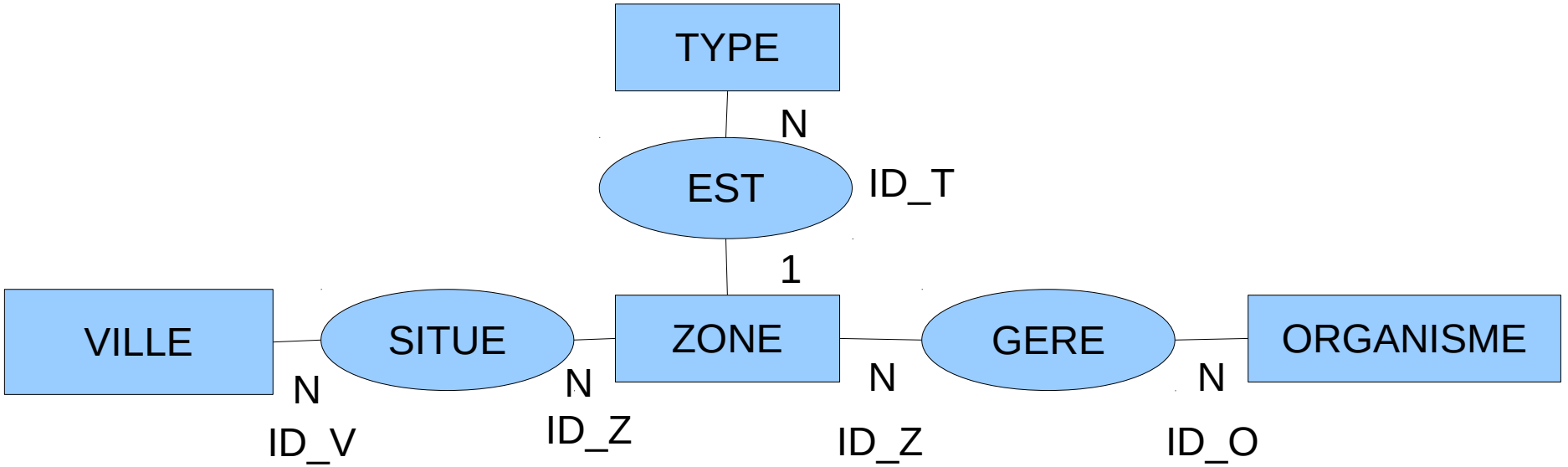
N°	Nom	Type	Villes	Organismes
01	PNR des Caps et Marais d'Opale	PNR	Marquise, Samer, Guines, Licques	France, Région HdF, Département du Pas-de-Calais
02	PNR Scarpe-Escaut	PNR	Saint-Amand-les-Eaux, Raismes, Marchiennes	France, Région HdF, Département du Nord
03	PNR Avesnois	PNR	Maroilles, Avesnes/Helpe, Le Quesnoy	France, Région HdF, Département du Nord
04	PNT du Hainaut	PNT	Saint-Amand-les-Eaux, Raismes, Marchiennes	Europe, France, Région HdF, Département du Nord, Belgique, Région Wallone
05	ENRLille Métropole	ENR	Croix, Leers, Marcq-en-Barœul, Roubaix, Tourcoing, Wasquehal, Wattrelos	Etat, Région HdF, Département du Nord, Métropole Européenne de Lille

Construisez le modèle entité-relation

Vous organiserez au mieux vos données pour éviter les redondances d'informations et faciliter les recoupements d'informations...

Indiquez pour chacune des entités et des relations la liste des attributs

Espaces naturels



- VILLE(<u>ID_V</u>, NOM)
- ORG(<u>ID_O</u>, NOM)
- TYPE(<u>ID_T</u>, NOM)
- ZONE(<u>ID_Z</u>, NOM, ID_T)
- SITUE(<u>ID_Z, ID_V</u>)
- GERE(<u>ID_O, ID_Z</u>)

Espaces naturels

Écrivez en SQL les commandes de création des tables et insertions de quelques une des données

Écrivez en SQL les requêtes donnant les résultats ci-dessous

- Liste des types de zones
- Liste des villes se trouvant dans un PNR
- Liste des organismes s'occupant de la gestion de PNT
- Liste des noms de zones dont le Département du Nord assure la gestion (seul ou non)

APPLICATIONS

2 – Étude de lignes ferroviaires

Lignes ferroviaires

On veut constituer une base de données avec les lignes ferroviaires de la région. Chaque ligne est repérée par un numéro.

On souhaite associer des informations complémentaires à ces lignes :

- nom de la ligne (exemples : "Lille-Dunkerque", "Lille-Calais-Londres"...)

- type de la ligne (exemples : "TER", "TGV"...)

- liste des gares desservies par la ligne, et pour chaque gare, la ville dans laquelle elle se trouve

Voici la liste des lignes (les listes de gares sont volontairement non exhaustives) :

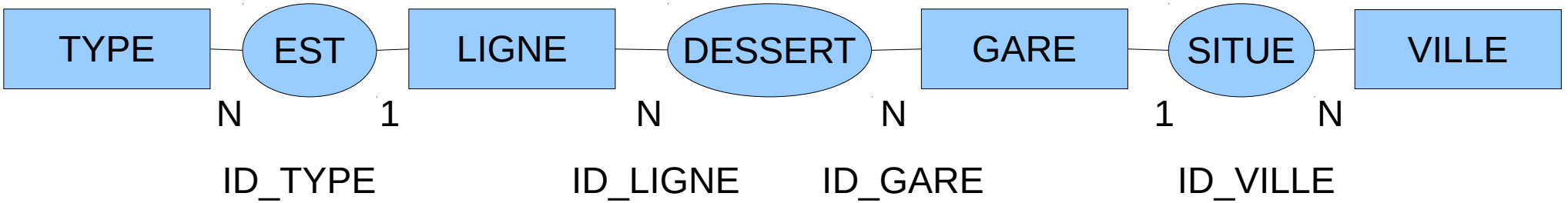
N°	Nom	Type	Gares (Ville)
08	Lille-Dunkerque	TER	Lille Flandres (Lille), Armentières (Armentières), Hazebrouck (Hazebrouck), Dunkerque (Dunkerque)
101	Lille-Paris	TGV	Lille Flandres (Lille), Paris Nord (Paris)
02	Lille-Douai-Arras-Amiens-Rouen	TER	Lille Flandres (Lille), Douai (Douai), Arras (Arras), Amiens (Amiens), Rouen Rive Droite (Rouen)
201	Lille-Calais-Londres	Eurostar	Lille Europe (Lille), Calais Frethun (Frethun), Londres Waterloo (Londres)
12	Lille-Hazebouck-Calais-Boulogne	TER	Lille Flandres (Lille), Armentières (Armentières), Hazebrouck (Hazebrouck), Calais Ville (Calais), Calais Frethun (Frethun), Boulogne Tintelleries (Boulogne-sur-Mer), Boulogne Ville (Boulogne-sur-Mer)

Construisez le modèle entité-relation

Vous organiserez au mieux vos données pour éviter les redondances d'informations et faciliter les recoupements d'informations...

Indiquez pour chacune des entités et des relations la liste des attributs

Lignes ferroviaires



- TYPE(ID_TYPE, NOM)
- VILLE(ID_VILLE, NOM)
- LIGNE(ID_LIGNE, NOM, NUMERO, ID_TYPE)
- GARE(ID_GARE, NOM, ID_VILLE)
- DESSERT(ID_LIGNE, ID_GARE)

Lignes ferroviaires

Écrivez en SQL les commandes de création des tables et insertions de quelques une des données

Écrivez en SQL les requêtes donnant les résultats ci-dessous

- Liste des types de lignes
- Liste des gares sur une ligne TGV
- Liste des villes par lesquelles passe une ligne TER
- Liste des lignes qui passent par Calais
- Nombre de lignes qui passent par Calais
- Nombre de gares desservies par la ligne Lille-Dunkerque
- Nombre de gare par type de ligne

APPLICATIONS

3 – Immeubles

Immeubles

On veut constituer une base de données d'immeubles en construction. Pour chacun de ces bâtiments, on connait :

- le numéro de permis de construire
- le nom du bâtiment
- la ville dans laquelle se réalise le projet
- le porteur du projet de construction
- les architectes
- les entreprises qui participent au chantier

Voici la liste des bâtiments :

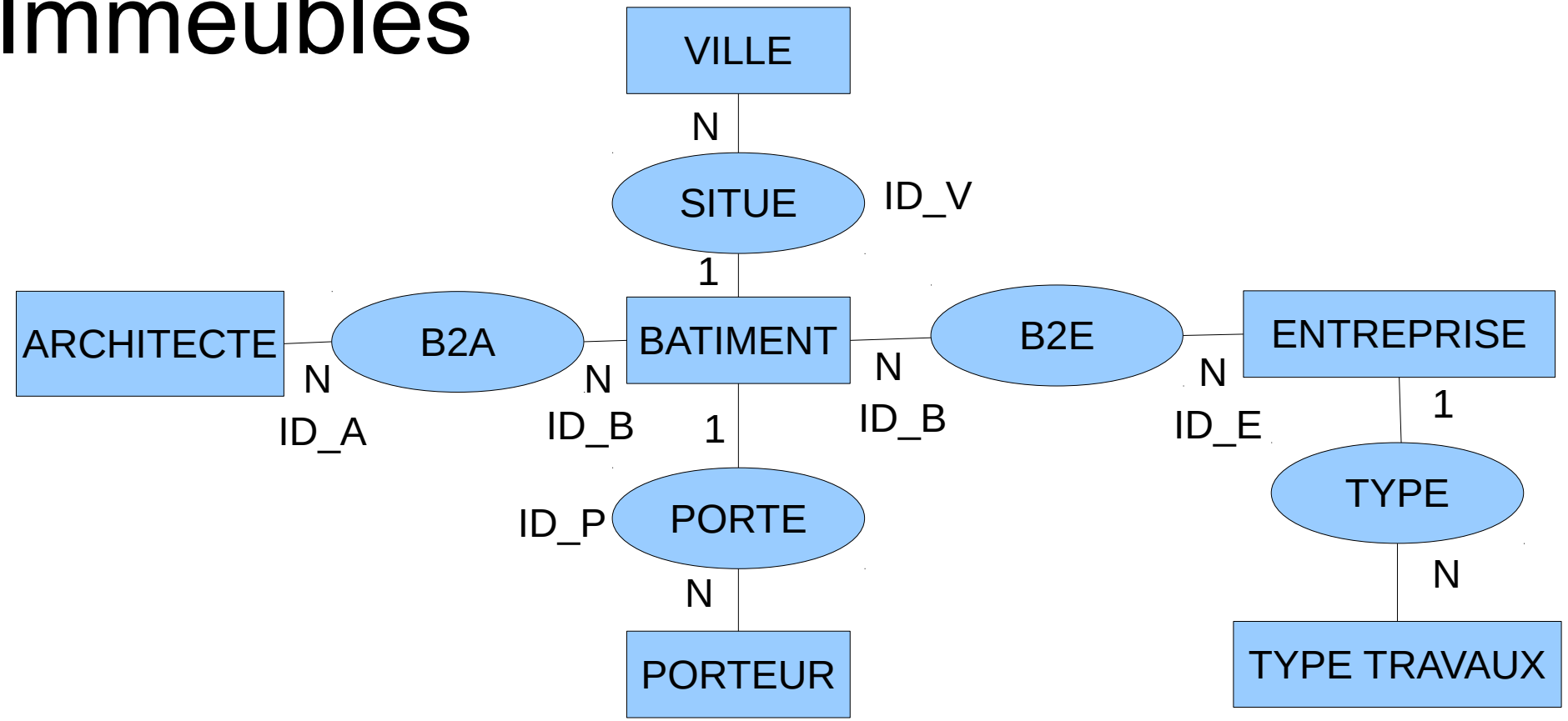
N°de permis	Nom du bâtiment	Ville	Porteur du projet	Architectes	Entreprises
45XV5	Résidence Camus	Lille	M. Dupont	Cabinet Lorem	SGEN (électricité), SGO (gros-œuvre), SPN (plomberie)
23GC4	Résidence Vinci	Douai	Mme Durand	Cabinet Sic	DOUELEC (électricité), SGO (gros-œuvre), PLOMBONOR (plomberie)
01SD6	Résidence Europe	Lille	M. Paul	Cabinet Ipsum Cabinet Lorem	DOUELEC (électricité), GROUVRE (gros-œuvre), SPMPLM (plomberie)
87PX9	Résidence Soleil	Arras	M. Dupont	Cabinet Sic	SGEN (électricité), SGO (gros-œuvre), PLOMBONOR (plomberie)

Construisez le modèle entité-relation

Vous organiserez au mieux vos données pour éviter les redondances d'informations et faciliter les recoupements d'informations...

Indiquez pour chacune des entités et des relations la liste des attributs

Immeubles



- VILLE(<u>ID_V</u>, NOM)
- ARCH(<u>ID_A</u>, NOM)
- PORTEUR(<u>ID_P</u>, NOM)
- TT(<u>ID_TT</u>, NOM)
- ENTR(<u>ID_E</u>, NOM, ID_TT)
- BAT(<u>ID_B</u>, NOM, ID_P, ID_V, PERMIS)
- B2A(<u>ID_B, ID_A</u>)
- B2E(<u>ID_B, ID_E</u>)

Immeubles

Écrivez en SQL les commandes de création des tables et insertions de quelques une des données

Écrivez en SQL les requêtes donnant les résultats ci-dessous

- Liste des porteurs de projets
- Liste des bâtiments en construction à Lille
- Liste des architectes travaillant sur un projet sur lequel intervient la société SGEN
- Liste des entreprises d'électricité intervenant sur des projets à Lille
- Nombre de bâtiments en construction à Lille
- Nombre de bâtiments en construction par ville

APPLICATIONS

4 – Étude des lérots

Lérots

On veut constituer une base de données des observations relevées pour le lérot, dans la région des Hauts-De-France sur plusieurs années.

Les observateurs fournissent des informations sur l'age, l'effectif et les conditions dans lesquelles l'observation a eu lieu (type de contact). Attention, une personne peut faire plusieurs observations sur le même lieu le même jour.

Voici un jeu de données minimales pour le projet.

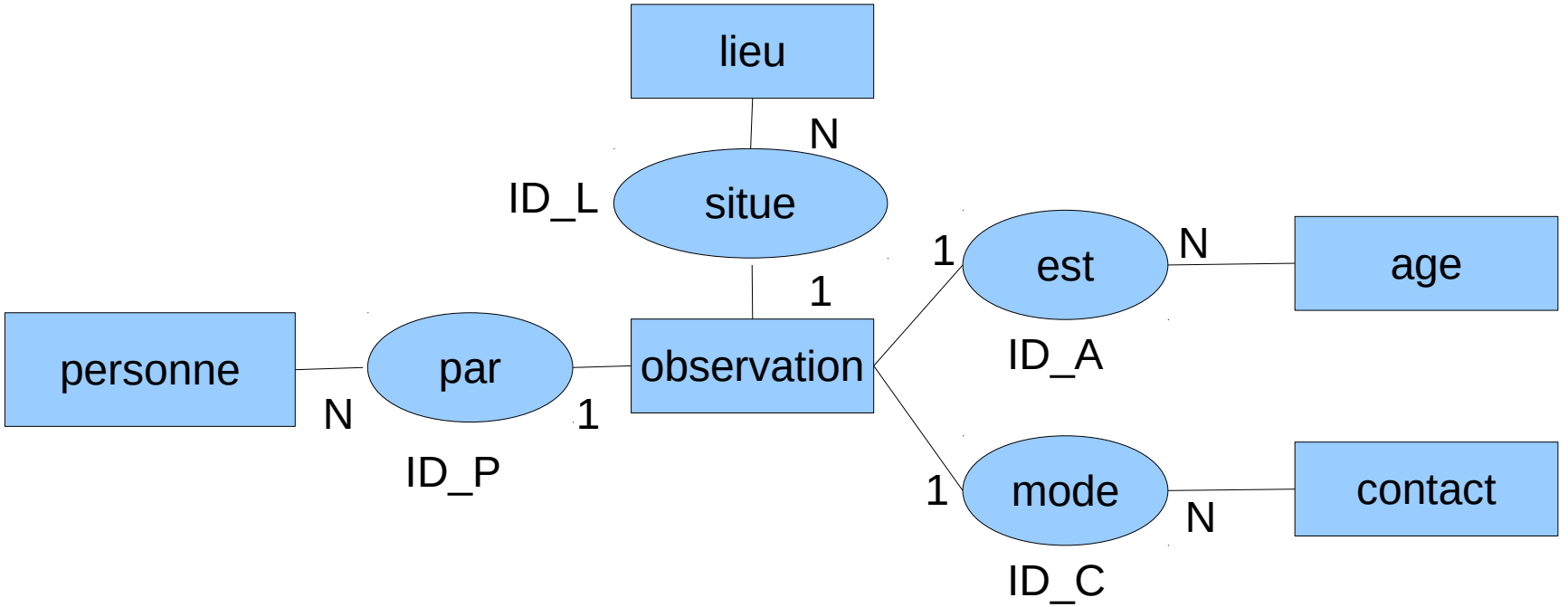
Observateur	Date	Commune	Dept	Âge	Effectif	Type de conta
P Olivier	01/03/2009	LILLE	59		1	mort
G José	01/09/2000	SAINT-AYBERT	59		2	cris
G José	01/10/2009	SAINT-AYBERT	59		1	traces de dent
G José	02/09/2000	SAINT-AYBERT	59		2	cris
H David	08/09/2002	DOUAI	59	Juvénile	1	obs. visuelle
G José	10/11/2009	SAINT-AYBERT	59	Adulte	1	obs. visuelle
S Hubert	15/08/2006	TAISNIERES-EN-THIERACHE	59	Adulte	1	obs. visuelle
S Hubert	15/08/2006	TAISNIERES-EN-THIERACHE	59	Juvénile	4	obs. visuelle
H David	21/08/2002	DOUAI	59	Juvénile	1	obs. visuelle
G Antoine	27/04/2003	WILLERVAL	62	Inconnu	1	mort

Construisez le modèle entité-relation

Vous organiserez au mieux vos données pour éviter les redondances d'informations et faciliter les recoupements d'informations...

Indiquez pour chacune des entités et des relations la liste des attributs

Lérots



- PERSONNE(ID_PERSONNE, NOM)
- LIEU(ID_LIEU, COMMUNE, DEP)
- AGE(ID_AGE, NOM)
- CONTACT(ID_CONTACT, NOM)
- OBSERVATION(ID_O, ID_P, ID_L, ID_C, ID_A, DATE, EFFECTIF)

Lérots

Écrivez en SQL les commandes de création des tables et insertions de quelques une des données

Écrivez en SQL les requêtes donnant les résultats ci-dessous

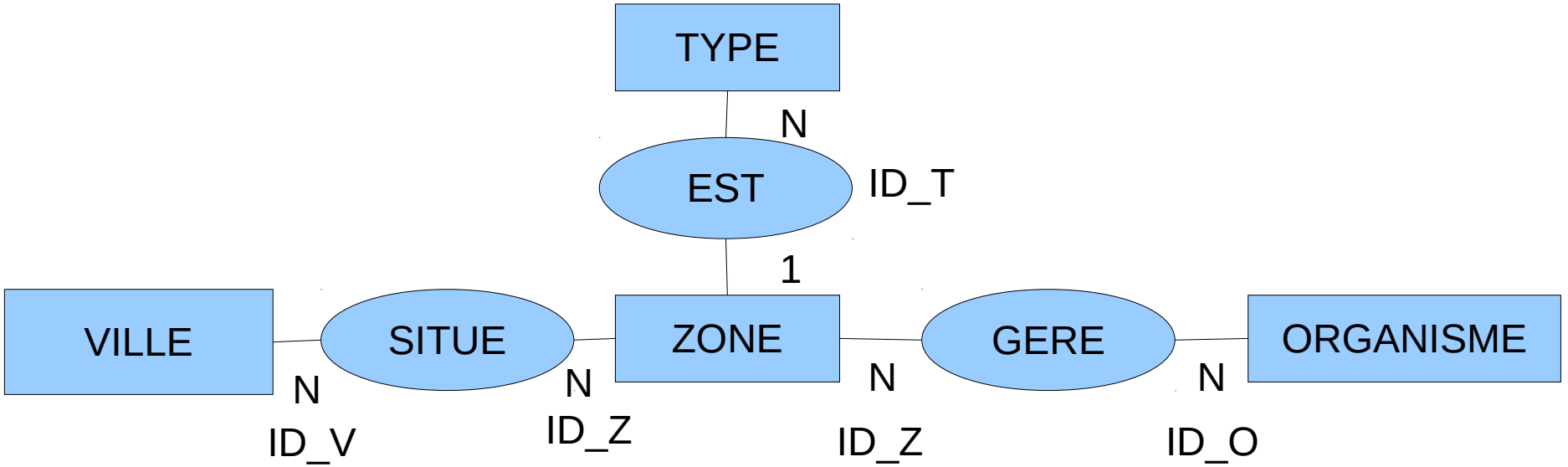
- Liste des observateurs
- Dates des observations réalisées à Lille
- Effectif pour chaque observation visuelle
- Liste des personnes ayant observé des adultes en 2006
- Tableau identique à celui donné en énoncé
- Nombre d'observateurs
- Nombre d'observations sans données pour l'age
- Nombre de lérots observés, par département
- Nombre de lérots observés, par département et par age

ANNEXES

ANNEXES

1 – Étude des espaces naturels

Espaces Naturels



- VILLE(<u>ID_V</u>, NOM)
- ORG(<u>ID_O</u>, NOM)
- TYPE(<u>ID_T</u>, NOM)
- ZONE(<u>ID_Z</u>, NOM, ID_T)
- SITUE(<u>ID_Z, ID_V</u>)
- GERE(<u>ID_O, ID_Z</u>)

Espaces naturels

1. Liste des types de zones

SELECT NOM FROM TYPE

Espaces naturels

2. Liste des villes se trouvant dans un PNR

```
SELECT V.NOM
FROM VILLE V, SITUE S, ZONE Z, TYPE T
WHERE T.NOM = 'PNR'
AND T.ID_T = Z.ID_T
AND Z.ID_Z = S.ID_Z
AND S.ID_V = V.ID_V
```

Espaces naturels

3. Liste des organismes s'occupant de la gestion d'un PNT

```
SELECT O.NOM
FROM TYPE T, ZONE Z, GERE G, ORGANISME O
WHERE T.NOM = 'PNT'
AND T.ID_T = Z.ID_T
AND Z.ID_Z = G.ID_Z
AND G.ID_O = O.ID_O
```

Espaces naturels

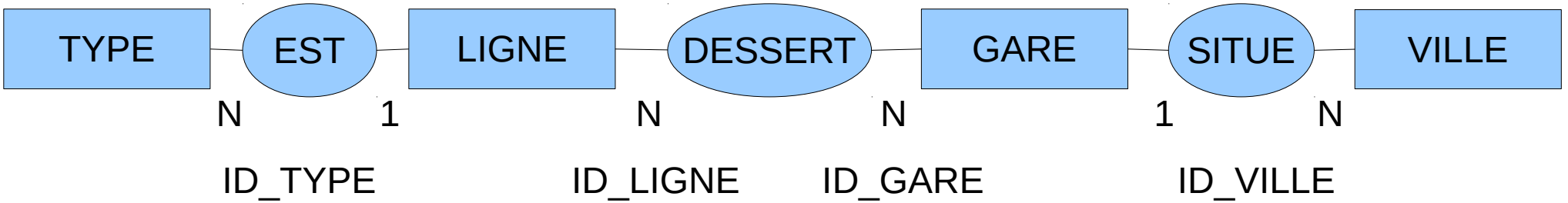
4. Liste des zones gérées par le département du nord

```
SELECT Z.NOM
FROM  ORGANISME O, GERE G, ZONE Z
WHERE O.NOM = 'DEP59'
AND O.ID_O = G.ID_O
AND G.ID_Z = Z.ID_Z
```

ANNEXES

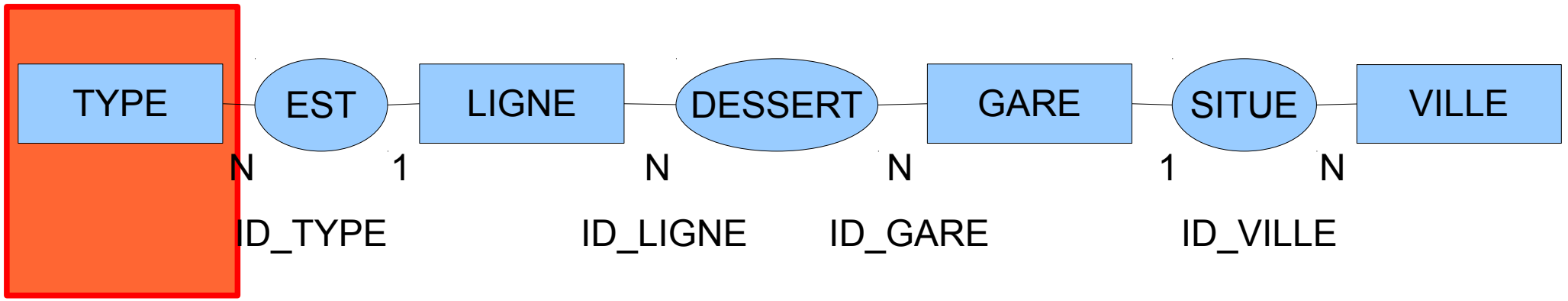
2 – Étude de lignes ferroviaires

Lignes Ferroviaires



- TYPE(ID_TYPE, NOM)
- VILLE(ID_VILLE, NOM)
- LIGNE(ID_LIGNE, NOM, NUMERO, ID_TYPE)
- GARE(ID_GARE, NOM, ID_VILLE)
- DESSERT(ID_LIGNE, ID_GARE)

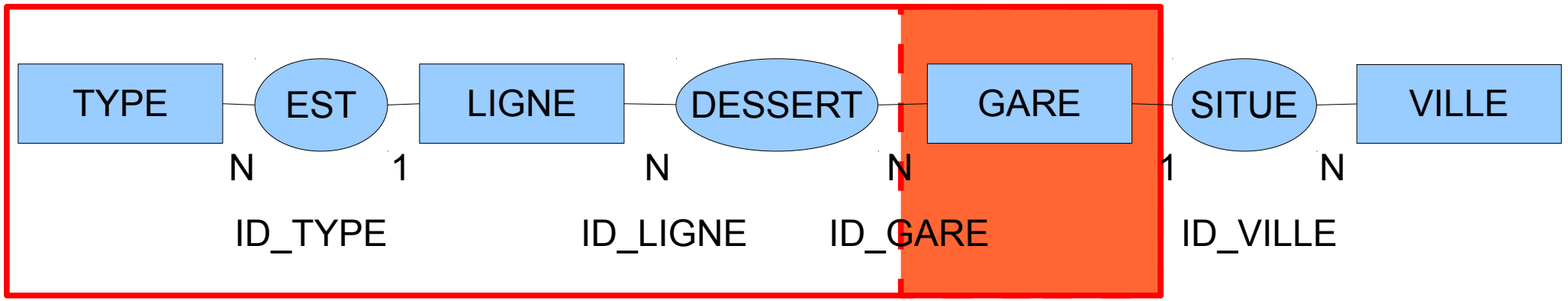
Ligne Ferroviaire – R1
Liste des types de lignes



SELECT TYPE.NOM FROM TYPE

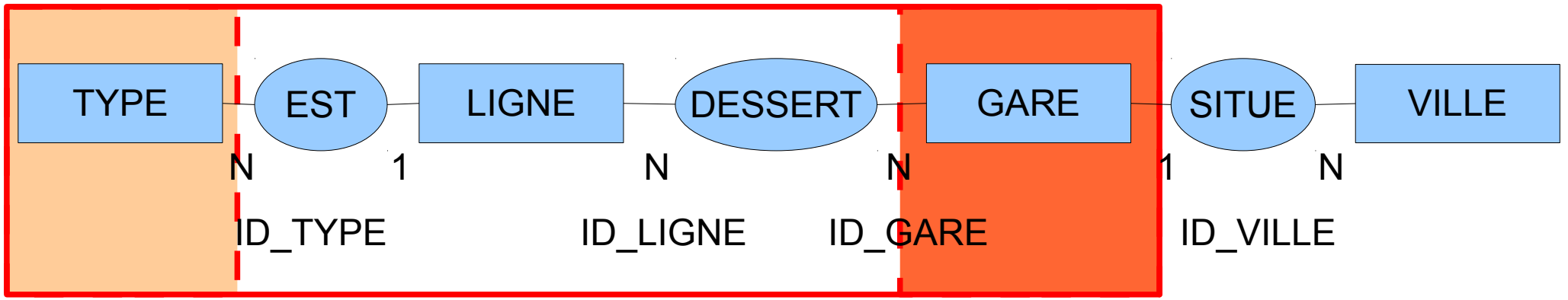
ORDER BY TYPE.NOM ASC

Ligne Ferroviaire – R2 1/6
Liste des gares sur une ligne TGV



SELECT GARE.NOM FROM GARE, DESSERT, LIGNE, TYPE

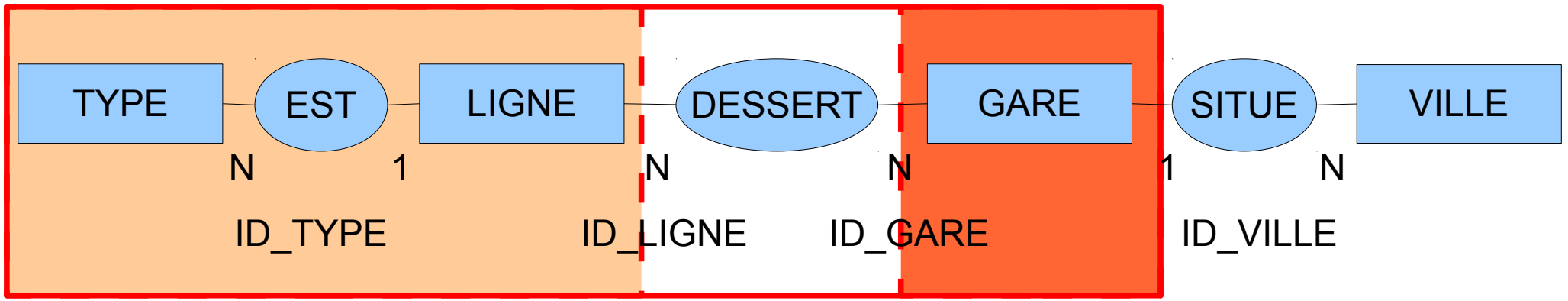
Ligne Ferroviaire – R2 2/6
Liste des gares sur une ligne TGV



SELECT GARE.NOM FROM GARE, DESSERT, LIGNE, TYPE

WHERE TYPE.NOM = 'TGV'

Ligne Ferroviaire – R2 3/6
Liste des gares sur une ligne TGV

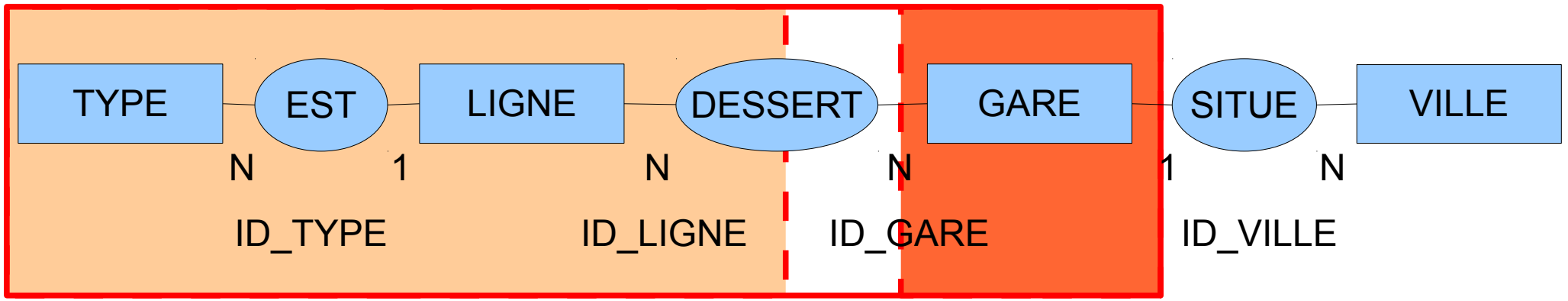


SELECT GARE.NOM FROM GARE, DESSERT, LIGNE, TYPE

WHERE TYPE.NOM = 'TGV'

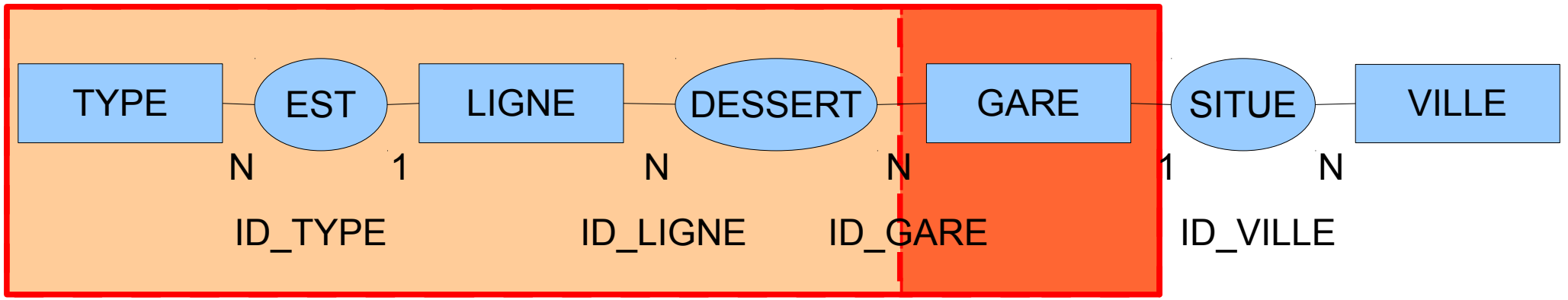
AND TYPE.ID_TYPE = LIGNE . ID_TYPE

Ligne Ferroviaire – R2 4/6
Liste des gares sur une ligne TGV



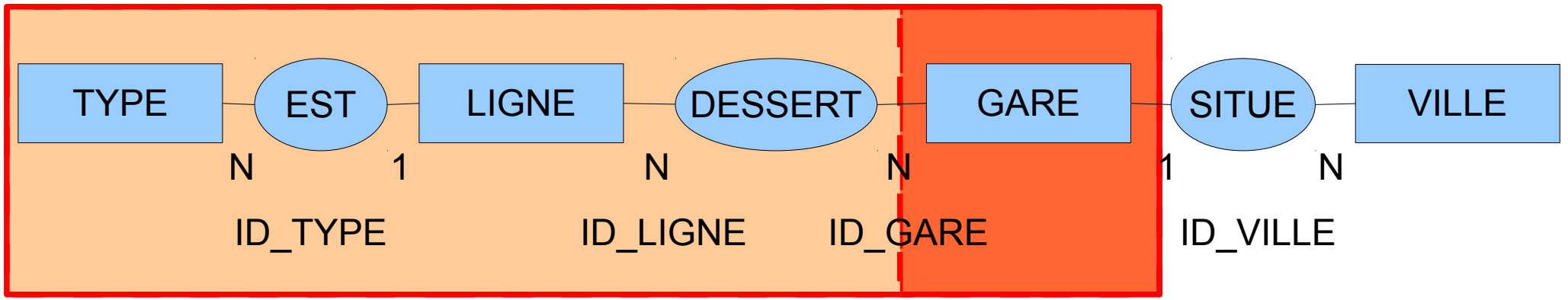

```
SELECT GARE.NOM FROM GARE, DESSERT, LIGNE, TYPE
WHERE TYPE.NOM = 'TGV'
AND TYPE.ID_TYPE = LIGNE . ID_TYPE
AND LIGNE.ID_LIGNE = DESSERT.ID_LIGNE
```

Ligne Ferroviaire – R2 5/6
Liste des gares sur une ligne TGV



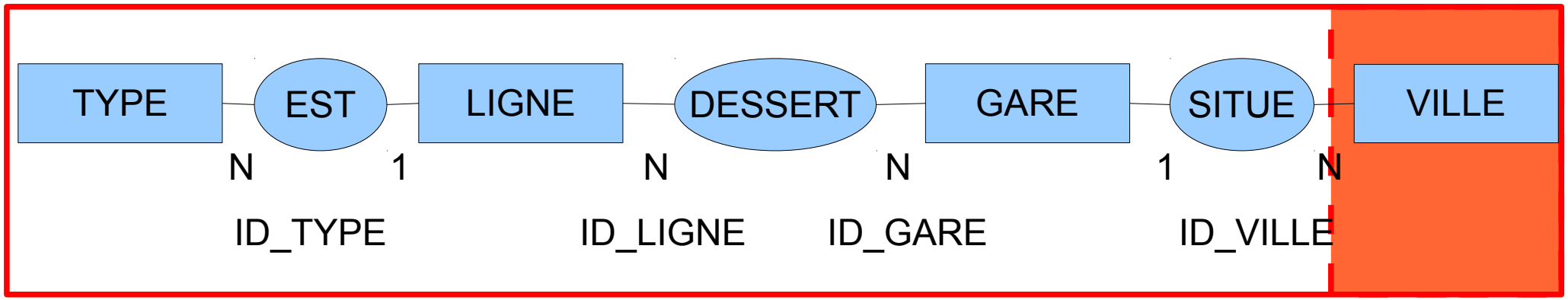

```
SELECT GARE.NOM FROM GARE, DESSERT, LIGNE, TYPE
WHERE TYPE.NOM = 'TGV'
AND TYPE.ID_TYPE = LIGNE . ID_TYPE
AND LIGNE.ID_LIGNE = DESSERT.ID_LIGNE
AND DESSERT.ID_GARE = GARE.ID_GARE
```

Ligne Ferroviaire – R2 6/6
Liste des gares sur une ligne TGV



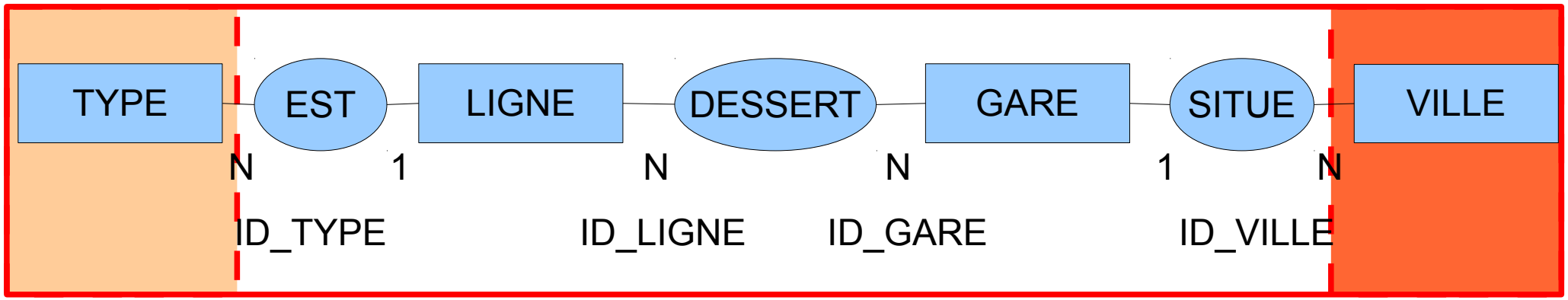

```
SELECT GARE.NOM FROM GARE, DESSERT, LIGNE, TYPE
WHERE TYPE.NOM = 'TGV'
AND TYPE.ID_TYPE = LIGNE . ID_TYPE
AND LIGNE.ID_LIGNE = DESSERT.ID_LIGNE
AND DESSERT.ID_GARE = GARE.ID_GARE
ORDER BY GARE.NOM ASC
```

Ligne Ferroviaire – R3 1/7
Liste des villes par lesquelles passent une ligne TER



SELECT VILLE.NOM FROM TYPE, LIGNE, DESSERT, GARE, VILLE

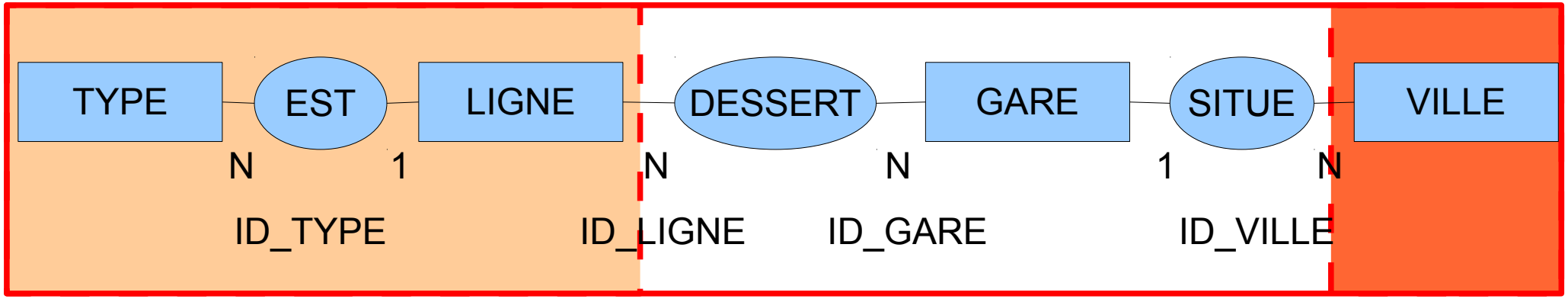
Ligne Ferroviaire – R3 2/7
Liste des villes par lesquelles passent une ligne TER



SELECT VILLE.NOM FROM TYPE, LIGNE, DESSERT, GARE, VILLE

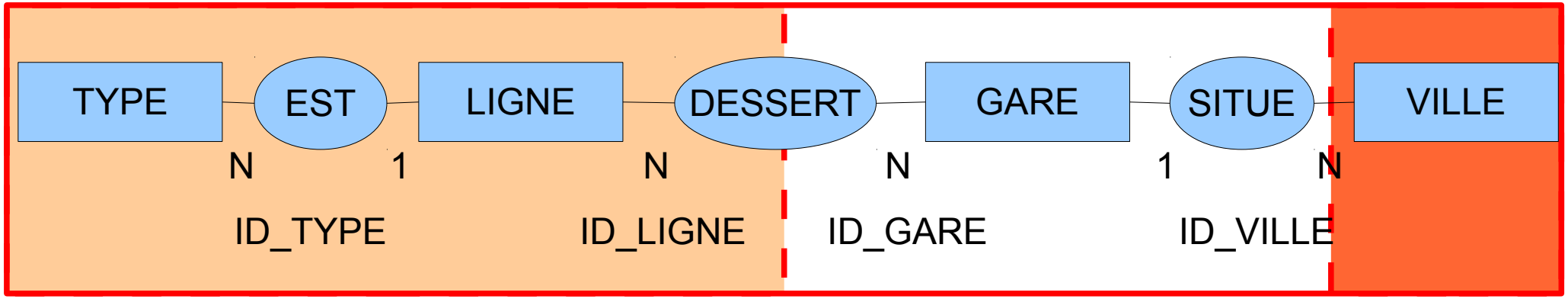
WHERE TYPE.NOM = 'TER'

Ligne Ferroviaire – R3 3/7
Liste des villes par lesquelles passent une ligne TER



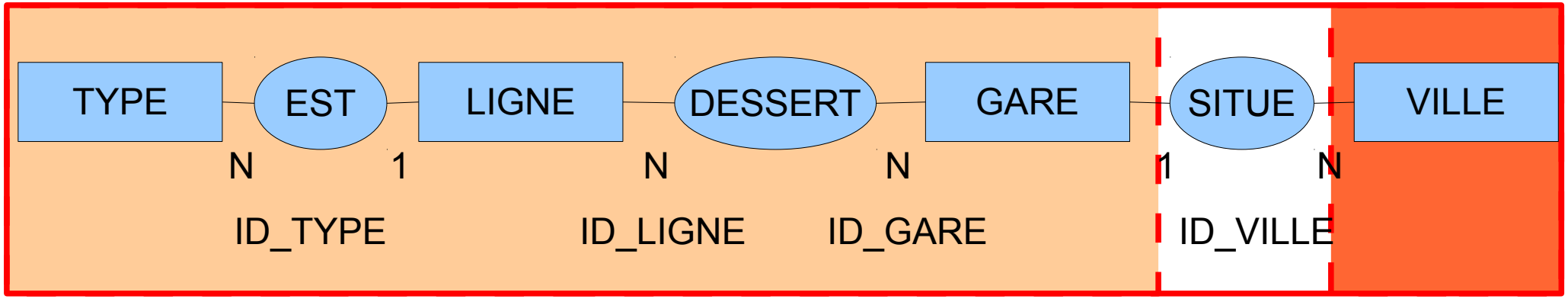

```
SELECT VILLE.NOM FROM TYPE, LIGNE, DESSERT, GARE, VILLE
WHERE TYPE.NOM = 'TER'
AND TYPE.ID_TYPE = LIGNE.ID_TYPE
```

Ligne Ferroviaire – R3 4/7
Liste des villes par lesquelles passent une ligne TER



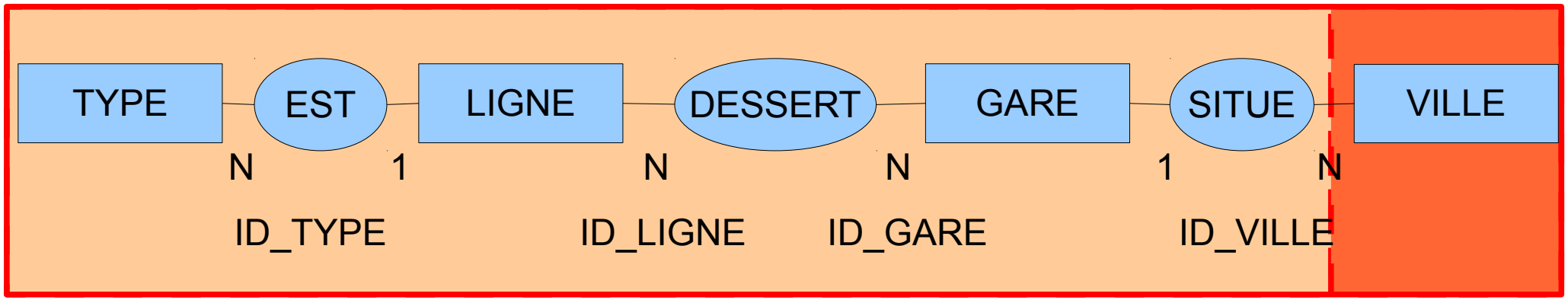

```
SELECT VILLE.NOM FROM TYPE, LIGNE, DESSERT, GARE, VILLE
WHERE TYPE.NOM = 'TER'
AND TYPE.ID_TYPE = LIGNE.ID_TYPE
AND LIGNE.ID_LIGNE = DESSERT.ID_LIGNE
```

Ligne Ferroviaire – R3 5/7
Liste des villes par lesquelles passent une ligne TER



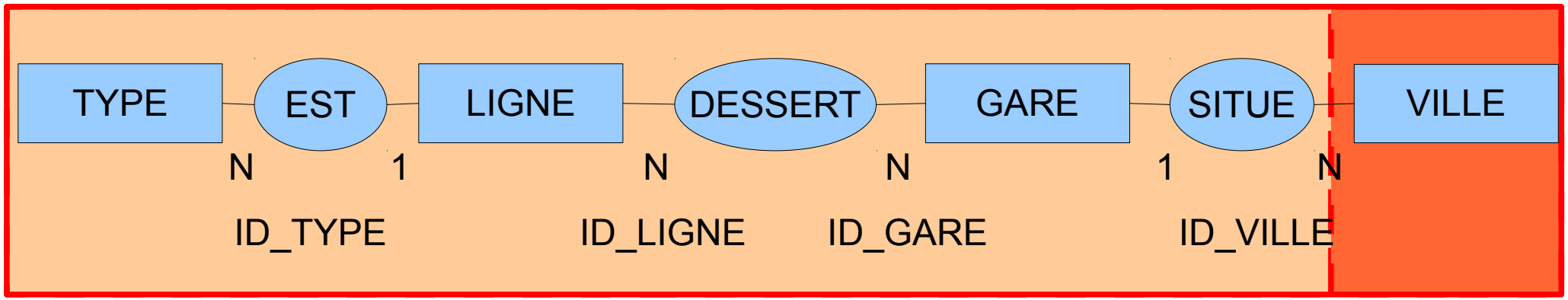

```
SELECT VILLE.NOM FROM TYPE, LIGNE, DESSERT, GARE, VILLE
WHERE TYPE.NOM = 'TER'
AND TYPE.ID_TYPE = LIGNE.ID_TYPE
AND LIGNE.ID_LIGNE = DESSERT.ID_LIGNE
AND DESSERT.ID_GARE = GARE.ID_GARE
```

Ligne Ferroviaire – R3 6/7
Liste des villes par lesquelles passent une ligne TER



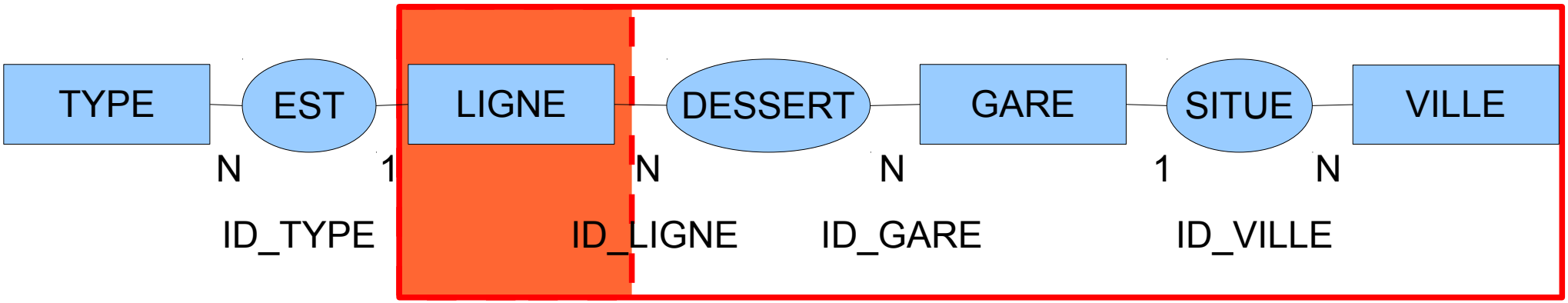

```
SELECT VILLE.NOM FROM TYPE, LIGNE, DESSERT, GARE, VILLE
WHERE TYPE.NOM = 'TER'
AND TYPE.ID_TYPE = LIGNE.ID_TYPE
AND LIGNE.ID_LIGNE = DESSERT.ID_LIGNE
AND DESSERT.ID_GARE = GARE.ID_GARE
AND GARE.ID_VILLE = VILLE.ID_VILLE
```

Ligne Ferroviaire – R3 7/7
Liste des villes par lesquelles passent une ligne TER



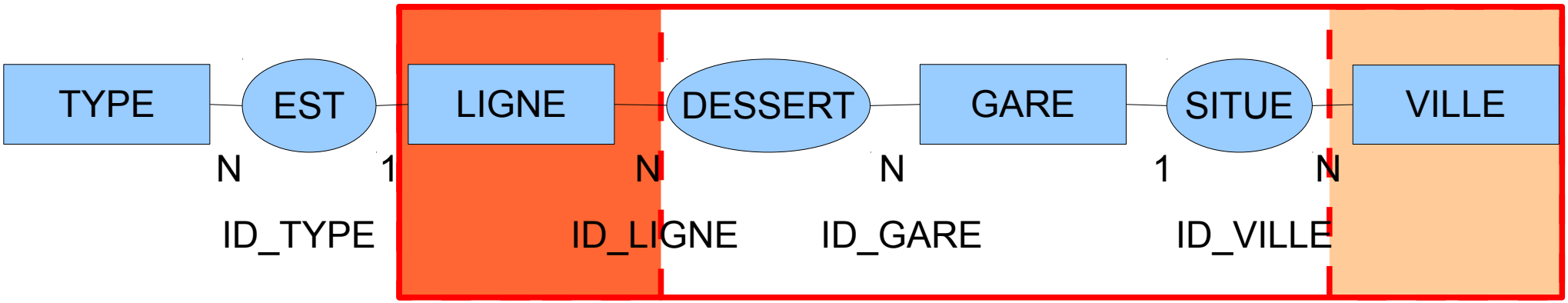

```
SELECT VILLE.NOM FROM TYPE, LIGNE, DESSERT, GARE, VILLE
WHERE TYPE.NOM = 'TER'
AND TYPE.ID_TYPE = LIGNE.ID_TYPE
AND LIGNE.ID_LIGNE = DESSERT.ID_LIGNE
AND DESSERT.ID_GARE = GARE.ID_GARE
AND GARE.ID_VILLE = VILLE.ID_VILLE
GROUP BY VILLE.NOM ORDER BY VILLE.NOM ASC
```

Ligne Ferroviaire – R4 1/5
Liste des lignes qui passent par Calais



SELECT LIGNE.NOM FROM LIGNE, DESSERT, GARE, VILLE

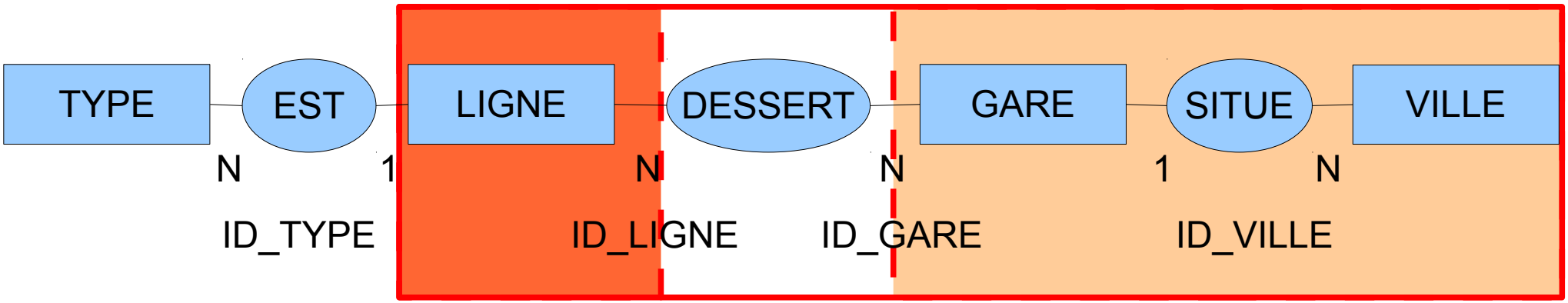
Ligne Ferroviaire – R4 2/5
Liste des lignes qui passent par Calais



SELECT LIGNE.NOM FROM LIGNE, DESSERT, GARE, VILLE

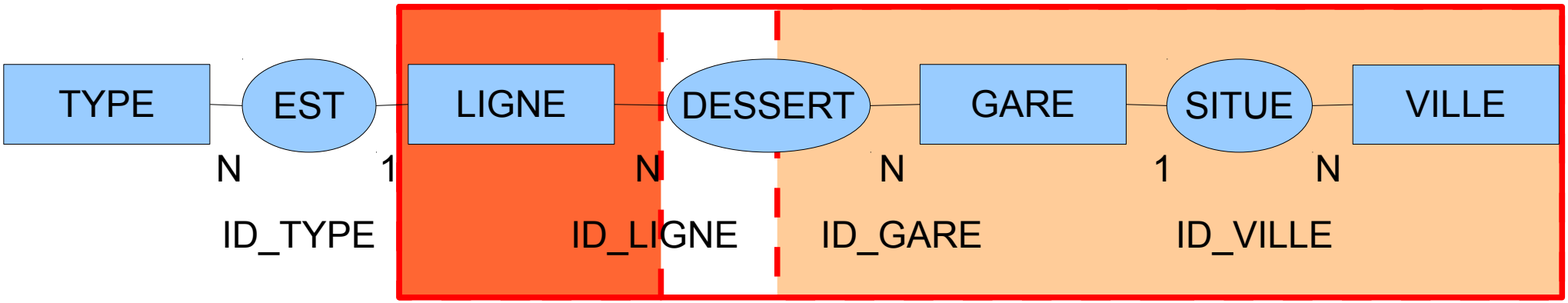
WHERE VILLE.NOM = 'Calais'

Ligne Ferroviaire – R4 3/5
Liste des lignes qui passent par Calais



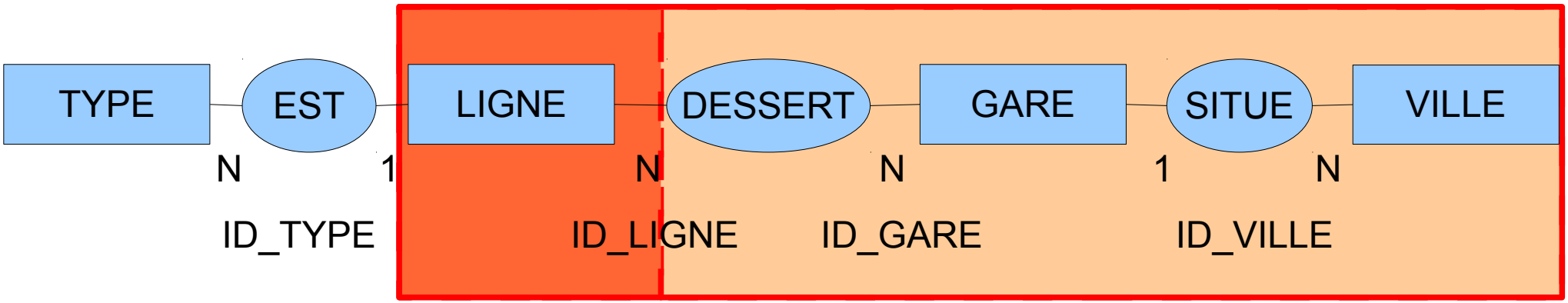

```
SELECT LIGNE.NOM FROM LIGNE, DESSERT, GARE, VILLE
WHERE VILLE.NOM = 'Calais'
AND VILLE.ID_VILLE = GARE.ID_VILLE
```

Ligne Ferroviaire – R4 4/5
Liste des lignes qui passent par Calais




```
SELECT LIGNE.NOM FROM LIGNE, DESSERT, GARE, VILLE
WHERE VILLE.NOM = 'Calais'
AND VILLE.ID_VILLE = GARE.ID_VILLE
AND GARE.ID_GARE = DESSERT.ID_GARE
```

Ligne Ferroviaire – R4 5/5
Liste des lignes qui passent par Calais




```
SELECT LIGNE.NOM FROM LIGNE, DESSERT, GARE, VILLE
WHERE VILLE.NOM = 'Calais'
AND VILLE.ID_VILLE = GARE.ID_VILLE
AND GARE.ID_GARE = DESSERT.ID_GARE
AND DESSERT.ID_LIGNE = LIGNE.ID_LIGNE
```

Ligne Ferroviaire – R5
Nombre de lignes qui passent par Calais

```
SELECT COUNT(LIGNE.NOM)
FROM LIGNE, DESSERT, GARE, VILLE
WHERE VILLE.NOM = 'Calais'
AND VILLE.ID_VILLE = GARE.ID_VILLE
AND GARE.ID_GARE = DESSERT.ID_GARE
AND DESSERT.ID_LIGNE = LIGNE.ID_LIGNE
```

Ligne Ferroviaire – R6
Nombre de gares desservies par la ligne Lille-Dunkerque

```
SELECT COUNT( GARE.NOM )
FROM GARE, DESSERT, LIGNE
WHERE LIGNE.NOM = 'Lille-Dunkerque'
AND GARE.ID_GARE = DESSERT.ID_GARE
AND DESSERT.ID_LIGNE = LIGNE.ID_LIGNE
```

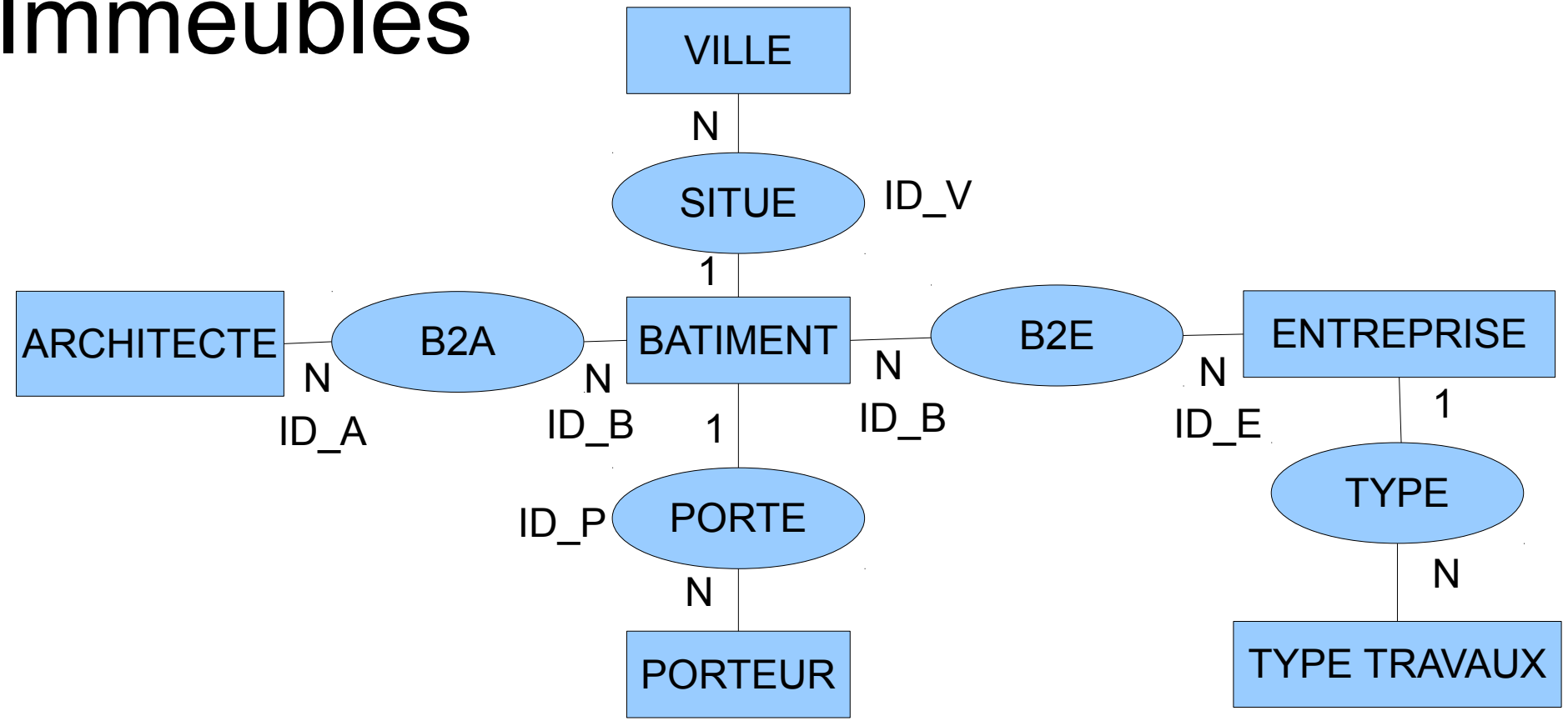
Ligne Ferroviaire – R7
Nombre de gare par type de ligne

```
SELECT TYPE.NOM, COUNT(GARE.NOM)
FROM TYPE, LIGNE, DESSERT, GARE
WHERE TYPE.ID_TYPE = LIGNE.ID_TYPE
AND LIGNE.ID_LIGNE = DESSERT.ID_LIGNE
AND DESSERT.ID_GARE = GARE.ID_GARE
GROUP BY TYPE.NOM
```

ANNEXES

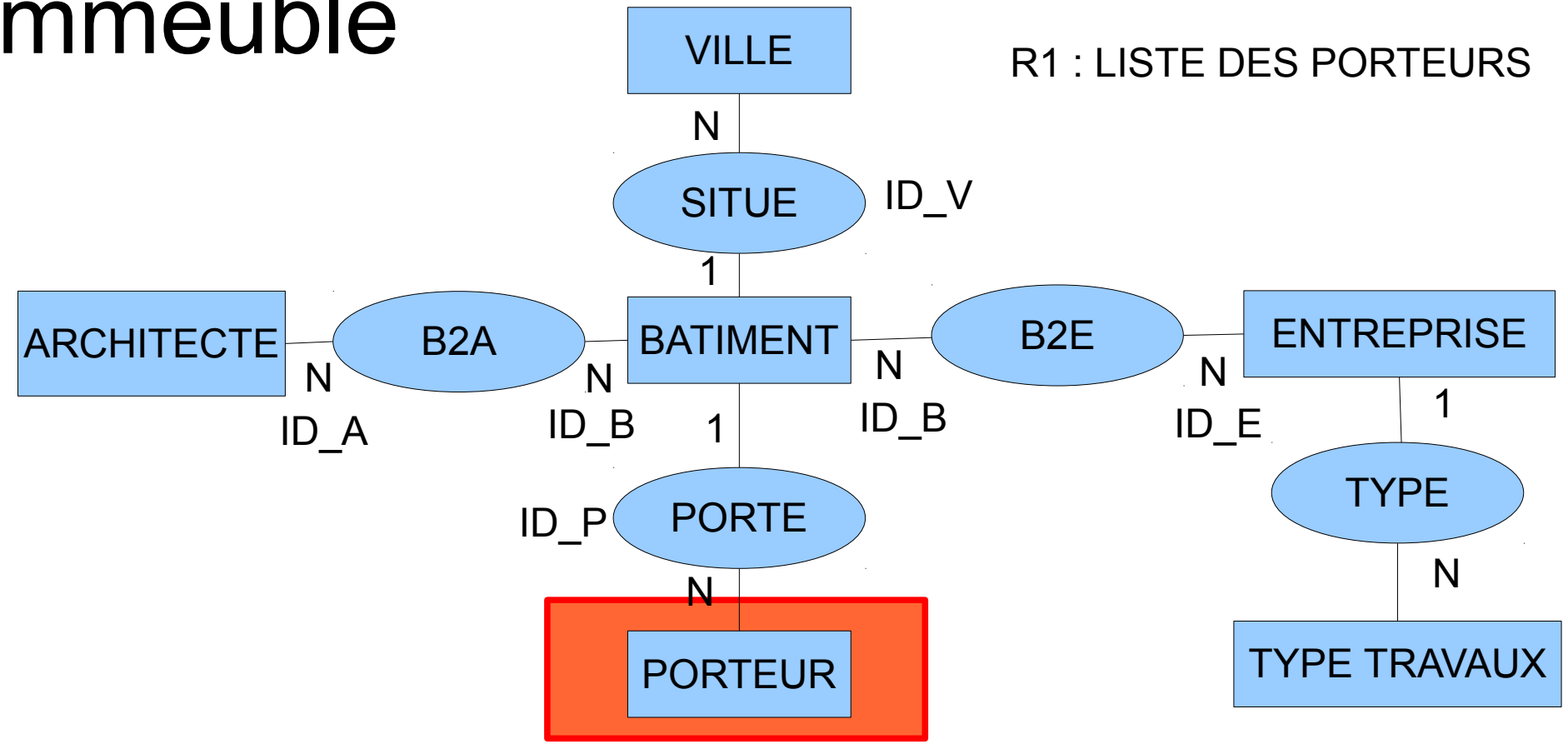
3 – Immeubles

Immeubles



- VILLE(<u>ID_V</u>, NOM)
- ARCH(<u>ID_A</u>, NOM)
- PORTEUR(<u>ID_P</u>, NOM)
- TT(<u>ID_TT</u>, NOM)
- ENTR(<u>ID_E</u>, NOM, ID_TT)
- BAT(<u>ID_B</u>, NOM, ID_P, ID_V, PERMIS)
- B2A(<u>ID_B, ID_A</u>)
- B2E(<u>ID_B, ID_E</u>)

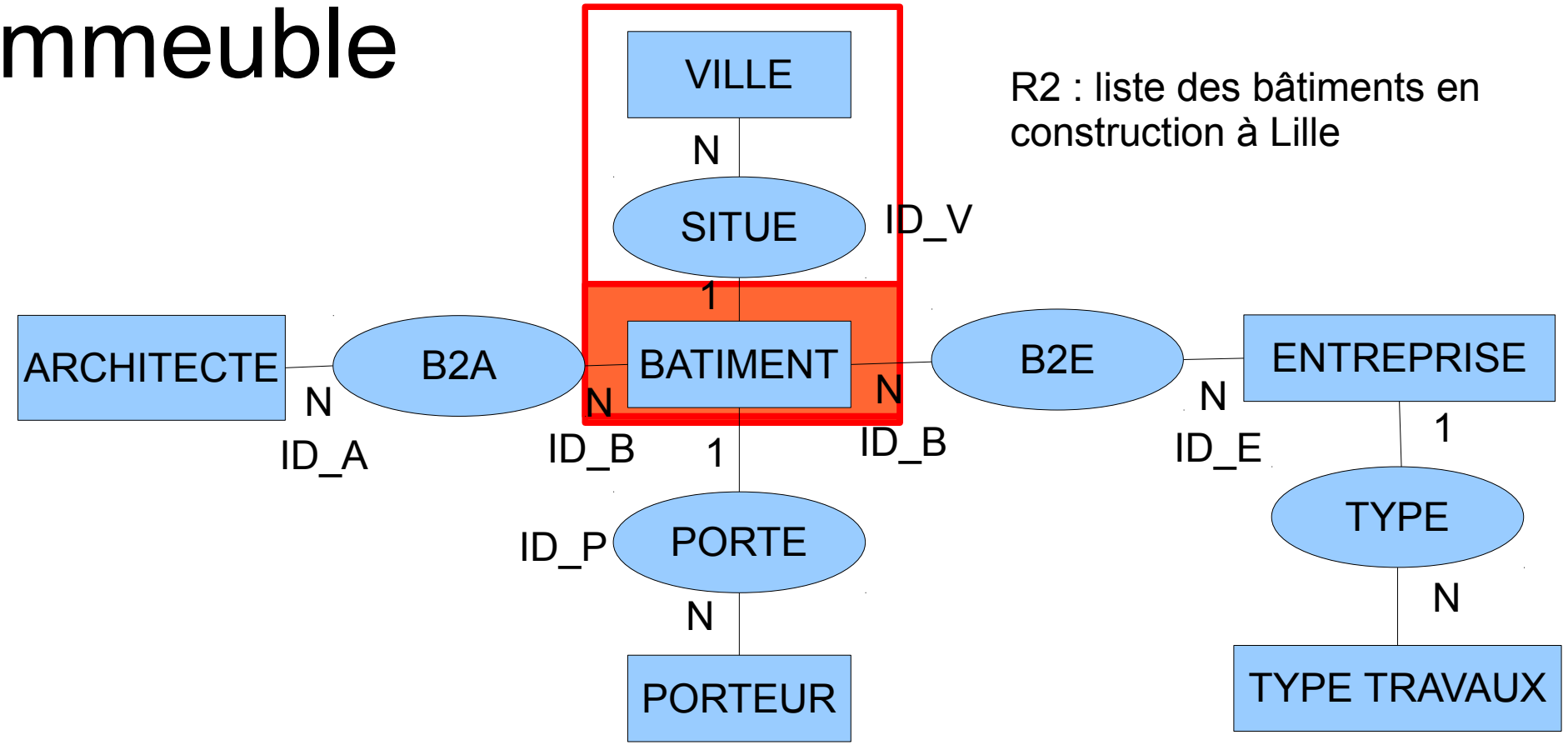
Immeuble



R1 : LISTE DES PORTEURS

SELECT PORTEUR.NOM FROM PORTEUR

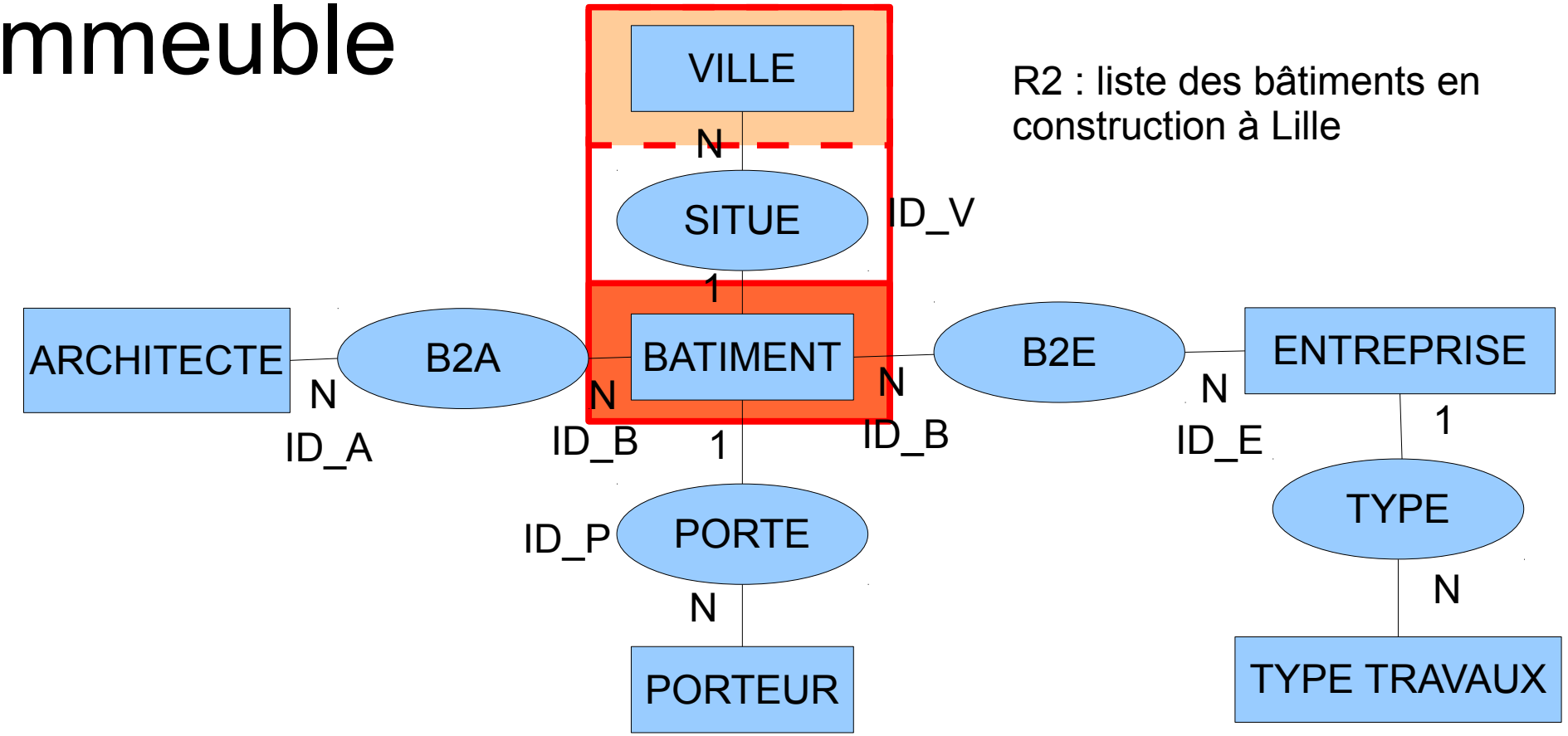
Immeuble



R2 : liste des bâtiments en construction à Lille

SELECT BAT.NOM FROM BAT, VILLE

Immeuble

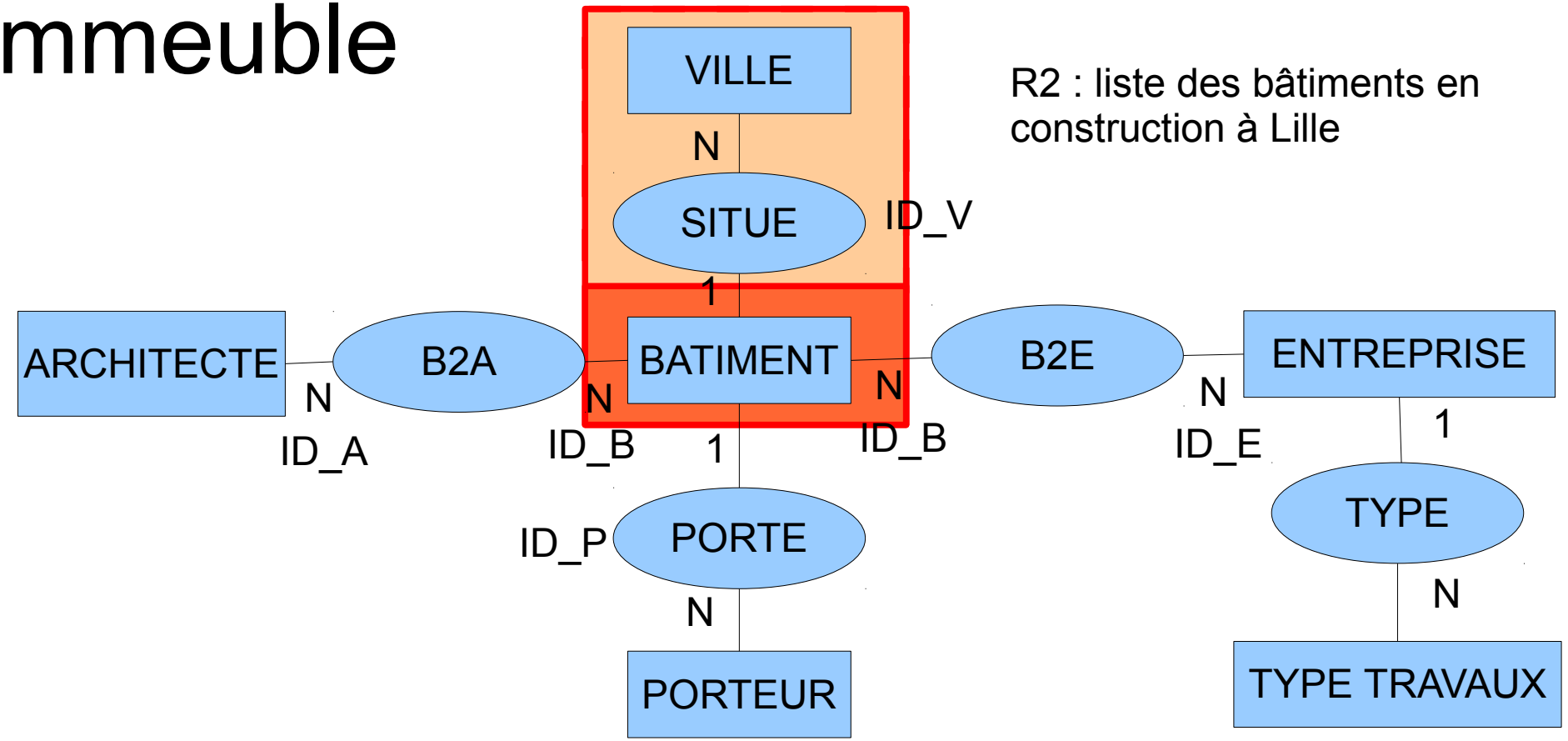


R2 : liste des bâtiments en construction à Lille

SELECT BAT.NOM FROM BAT, VILLE

WHERE VILLE.NOM = 'Lille'

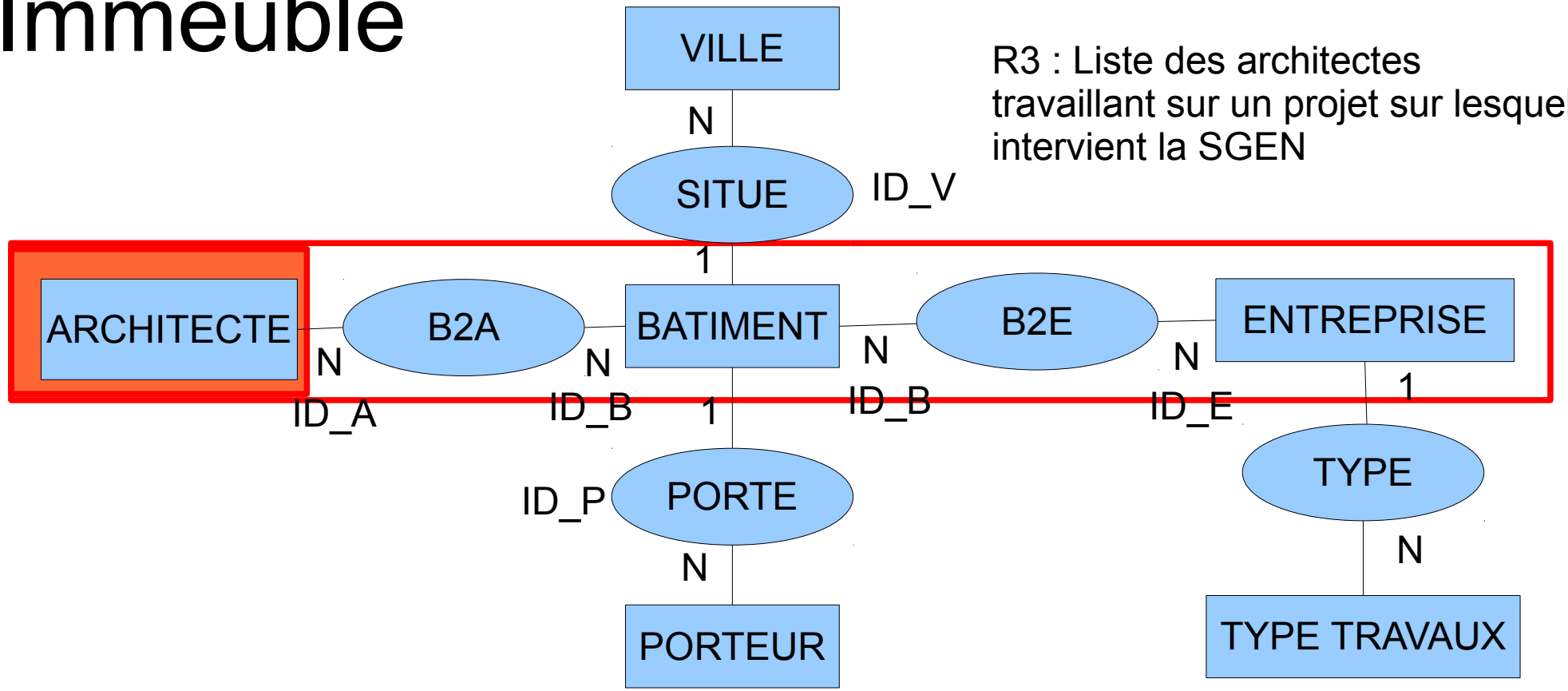
Immeuble



R2 : liste des bâtiments en construction à Lille

```
SELECT BAT.NOM FROM BAT, VILLE
WHERE VILLE.NOM = 'Lille'
AND VILLE.ID_V = BAT.ID_V
```

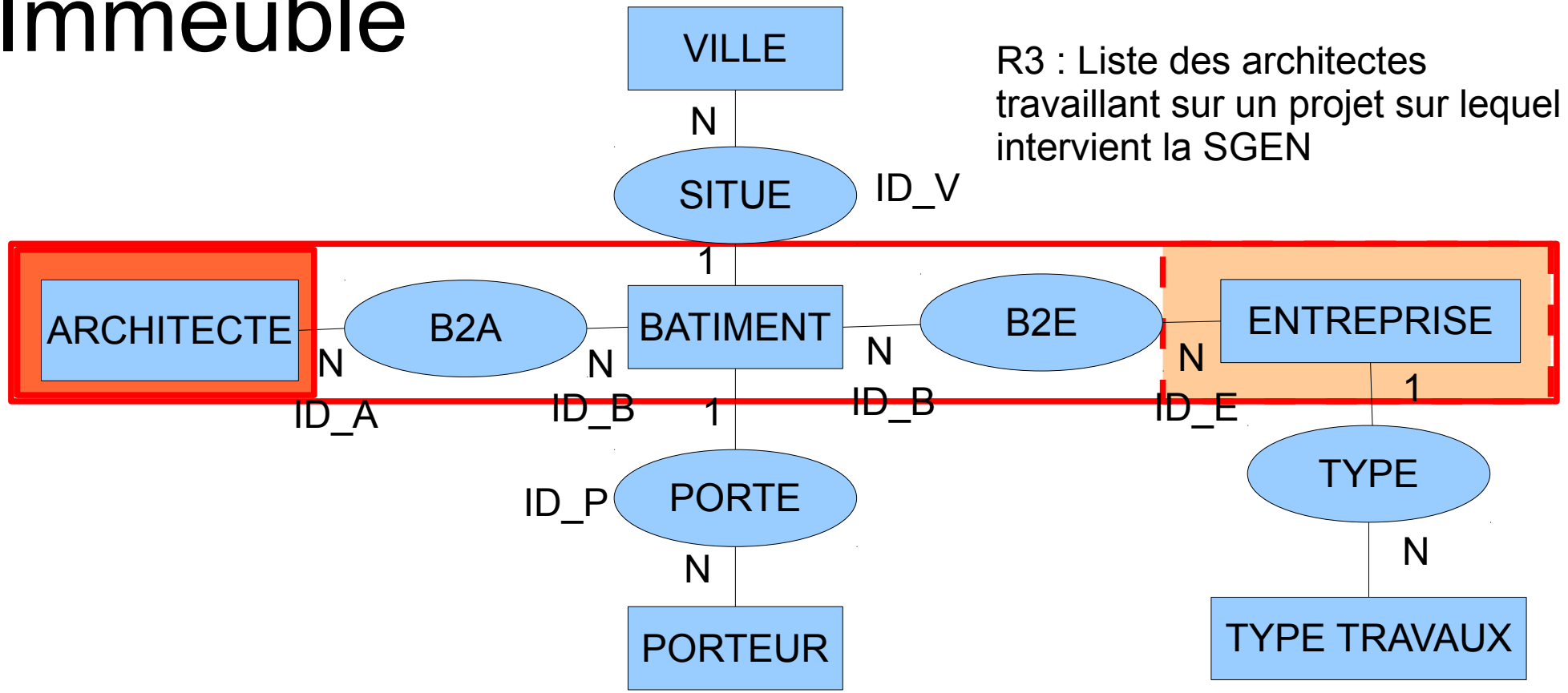
Immeuble



R3 : Liste des architectes travaillant sur un projet sur lesquel intervient la SGEN

SELECT ARCH.NOM FROM ARCH, B2A, BAT, B2E, ENT

Immeuble

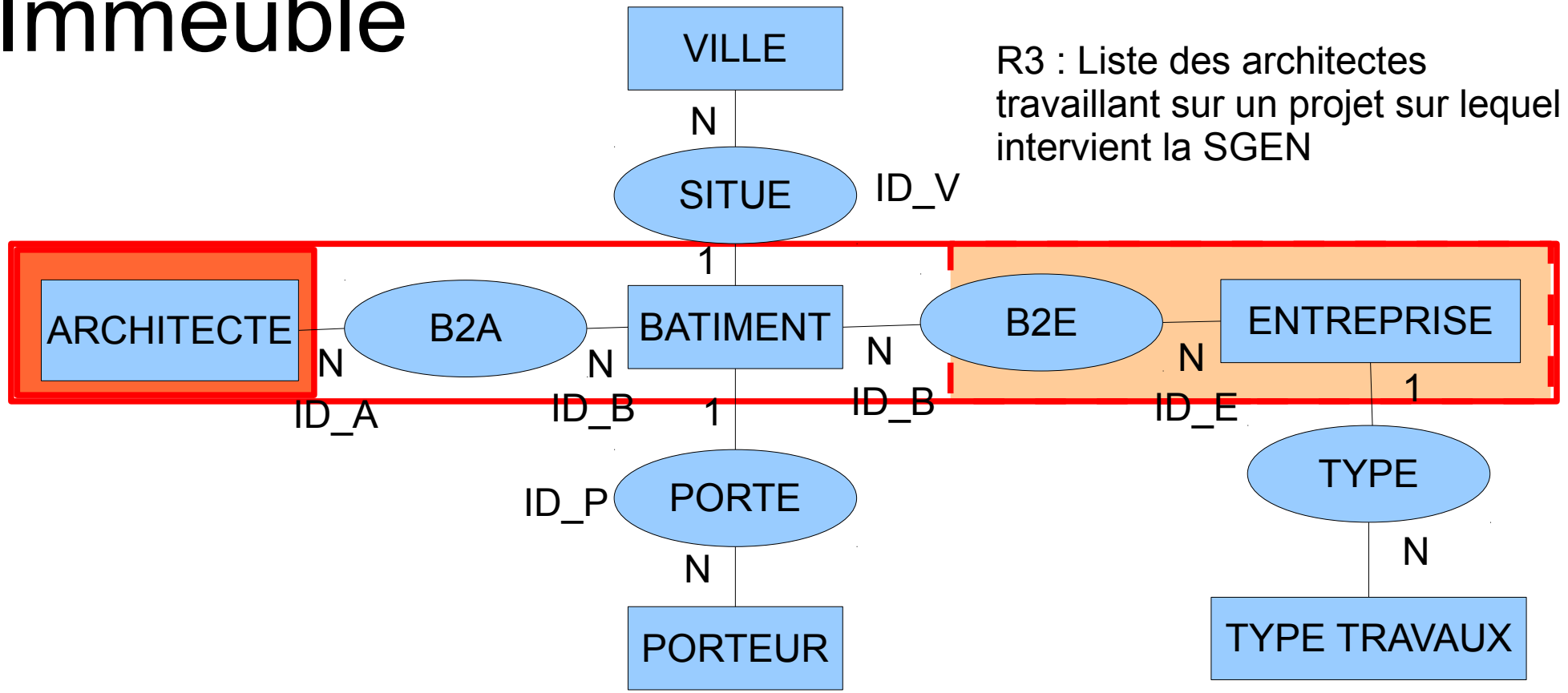


R3 : Liste des architectes travaillant sur un projet sur lequel intervient la SGEN

SELECT ARCH.NOM FROM ARCH, B2A, BAT, B2E, ENT

WHERE ENT.NOM = 'SGEN'

Immeuble



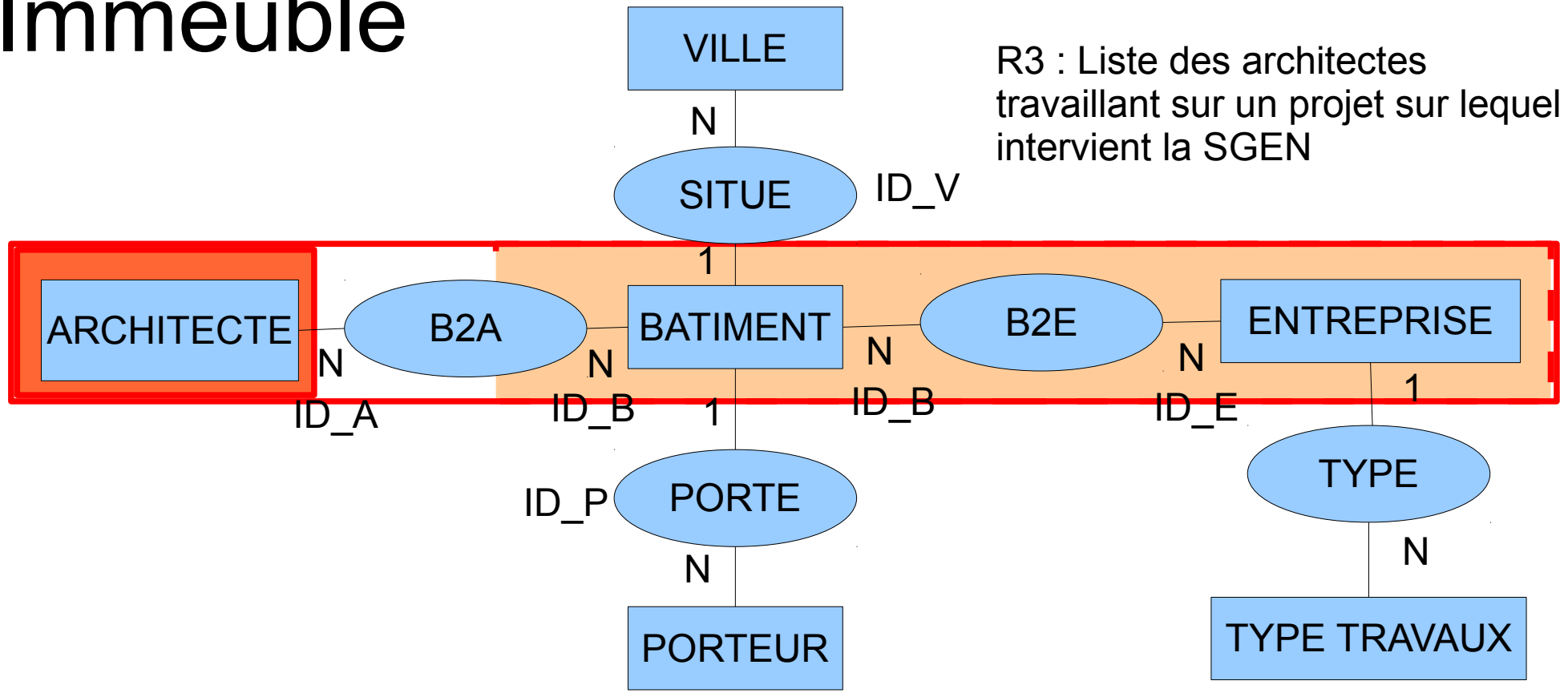
R3 : Liste des architectes travaillant sur un projet sur lequel intervient la SGEN

SELECT ARCH.NOM FROM ARCH, B2A, BAT, B2E, ENT

WHERE ENT.NOM = 'SGEN'

AND ENT.ID_E = B2E.ID_E

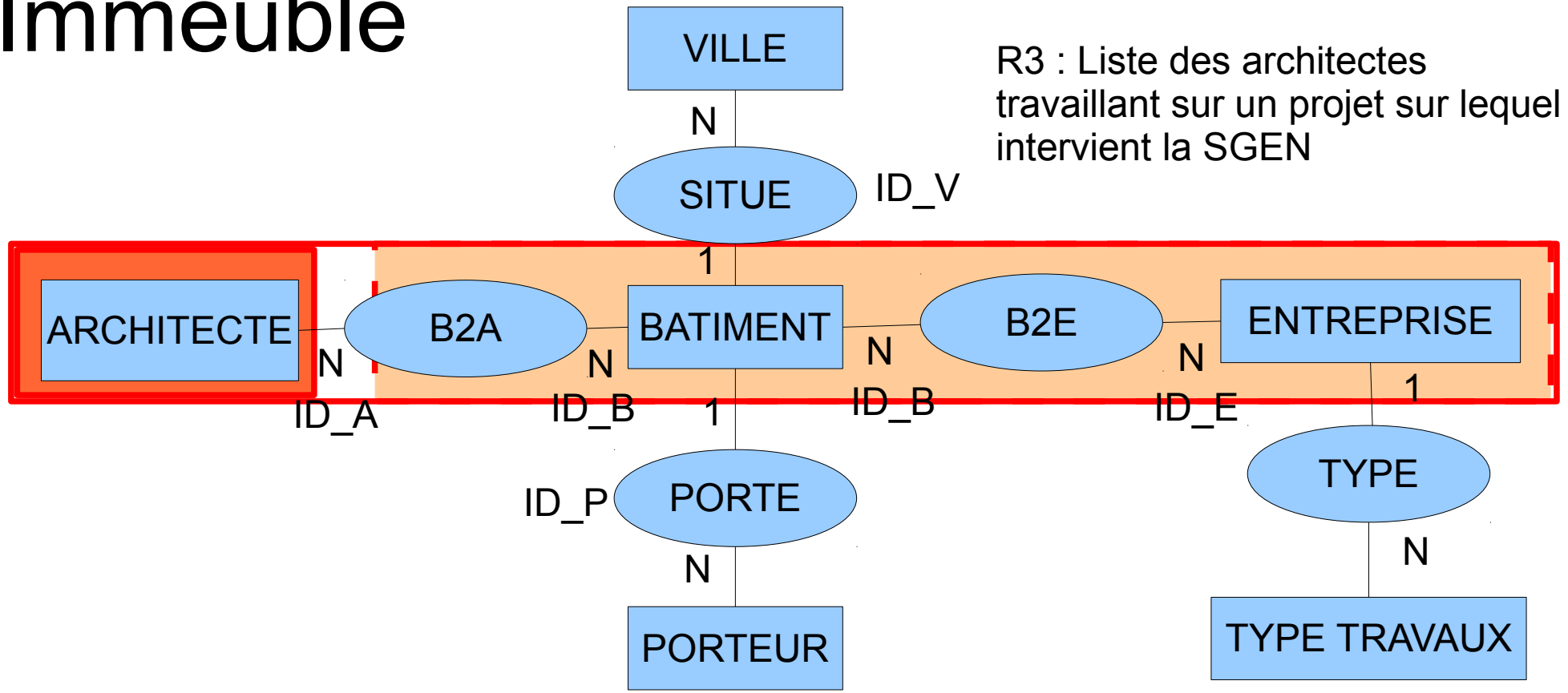
Immeuble



R3 : Liste des architectes travaillant sur un projet sur lequel intervient la SGEN

```
SELECT ARCH.NOM FROM ARCH, BAT, B2E, ENT
WHERE ENT.NOM = 'SGEN'
AND ENT.ID_E = B2E.ID_E
AND B2E_ID_B = BAT.ID_BAT
```

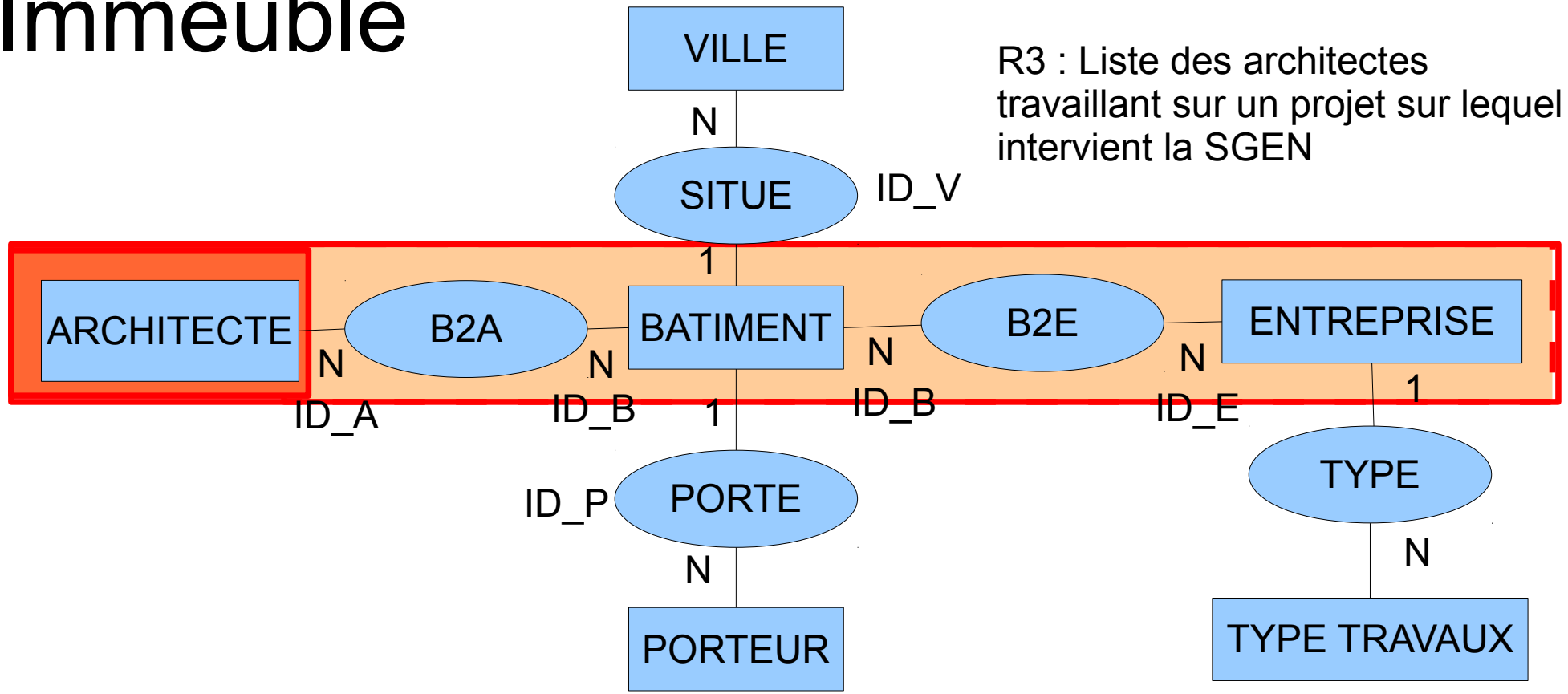
Immeuble



R3 : Liste des architectes travaillant sur un projet sur lequel intervient la SGEN

```
SELECT ARCH.NOM FROM ARCH, BAT, B2E, ENT
WHERE ENT.NOM = 'SGEN'
AND ENT.ID_E = B2E.ID_E
AND B2E_ID_B = BAT.ID_B
AND BAT.ID_B = B2A.ID_B
```

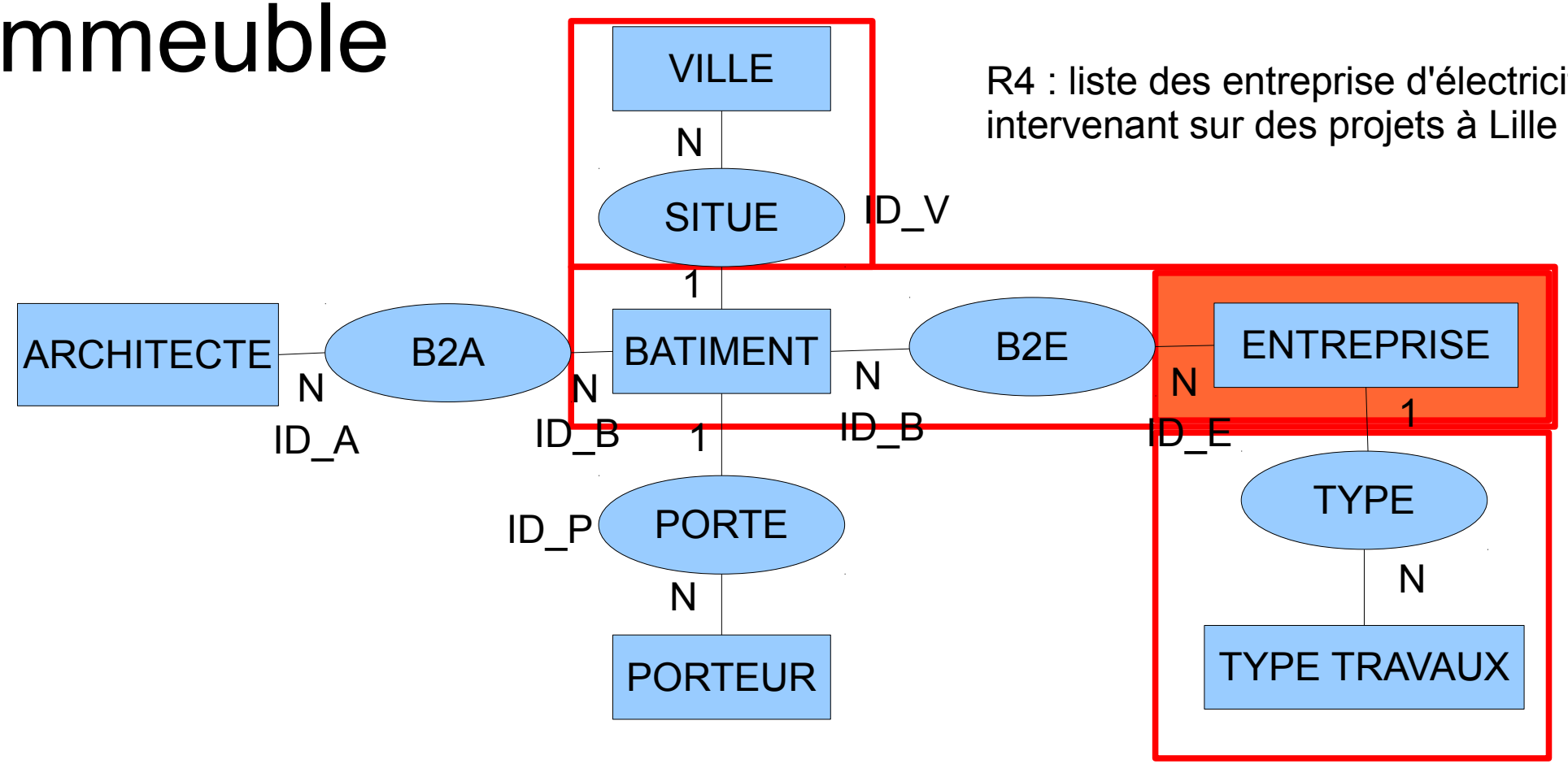
Immeuble



R3 : Liste des architectes travaillant sur un projet sur lequel intervient la SGEN

```
SELECT ARCH.NOM FROM ARCH, BAT, B2E, ENT
WHERE ENT.NOM = 'SGEN'
AND ENT.ID_E = B2E.ID_E
AND B2E_ID_B = BAT.ID_B
AND BAT.ID_B = B2A.ID_B
AND B2A_ID_A = ARCH.ID_A
```

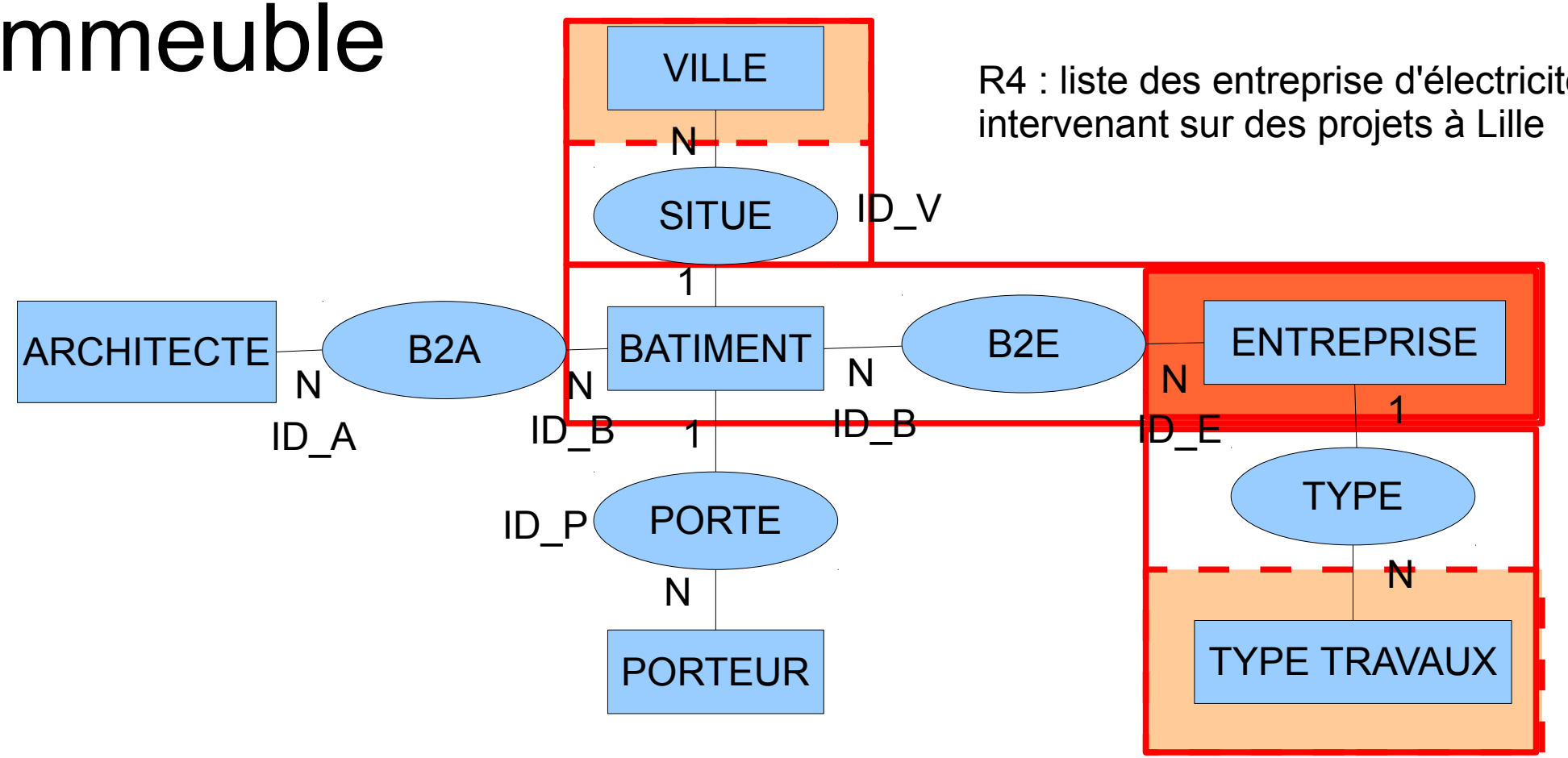
Immeuble



R4 : liste des entreprise d'électricité intervenant sur des projets à Lille

SELECT ENT.NOM FROM ENT, TT, B2E, BAT, VILLE

Immeuble

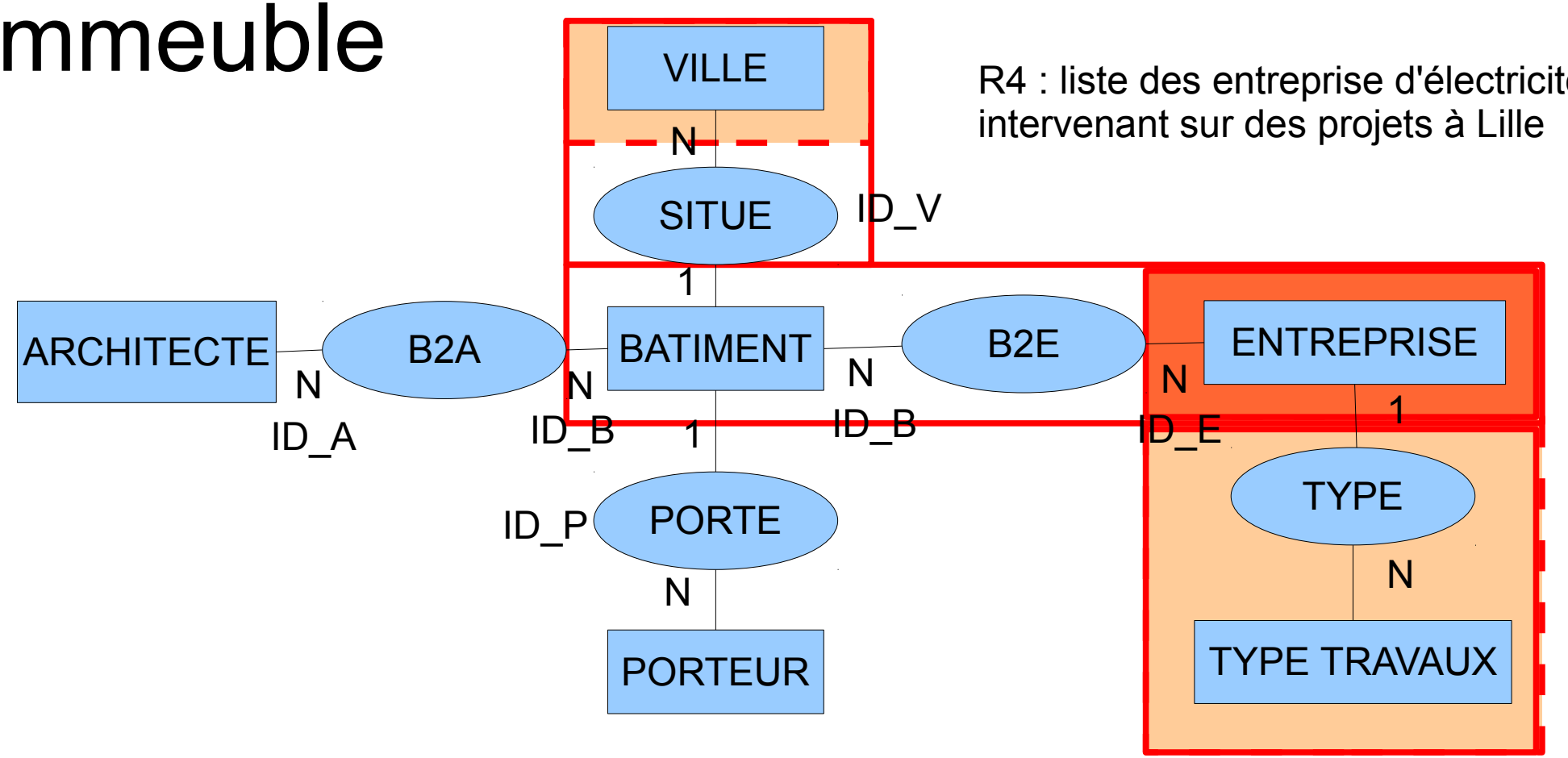


R4 : liste des entreprise d'électricité intervenant sur des projets à Lille

SELECT ENT.NOM FROM ENT, TT, B2E, BAT, VILLE

WHERE TT.NOM = 'Electricité' AND VILLE.NOM = 'Lille'

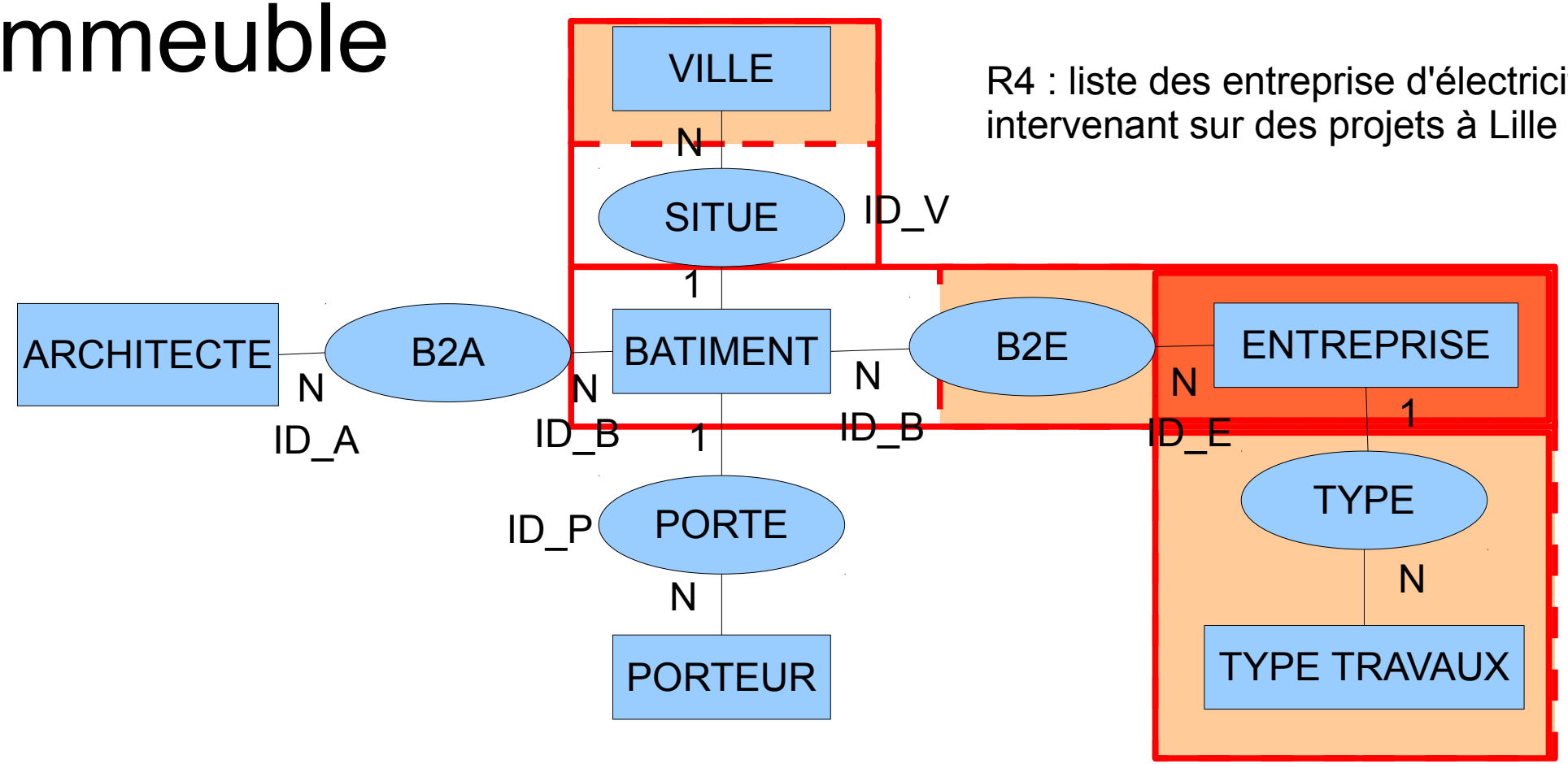
Immeuble



R4 : liste des entreprise d'électricité intervenant sur des projets à Lille

```
SELECT ENT.NOM FROM ENT, TT, B2E, BAT, VILLE
WHERE TT.NOM = 'Electricité' AND VILLE.NOM = 'Lille'
AND TT.ID_TT = ENT.ID_TT
```

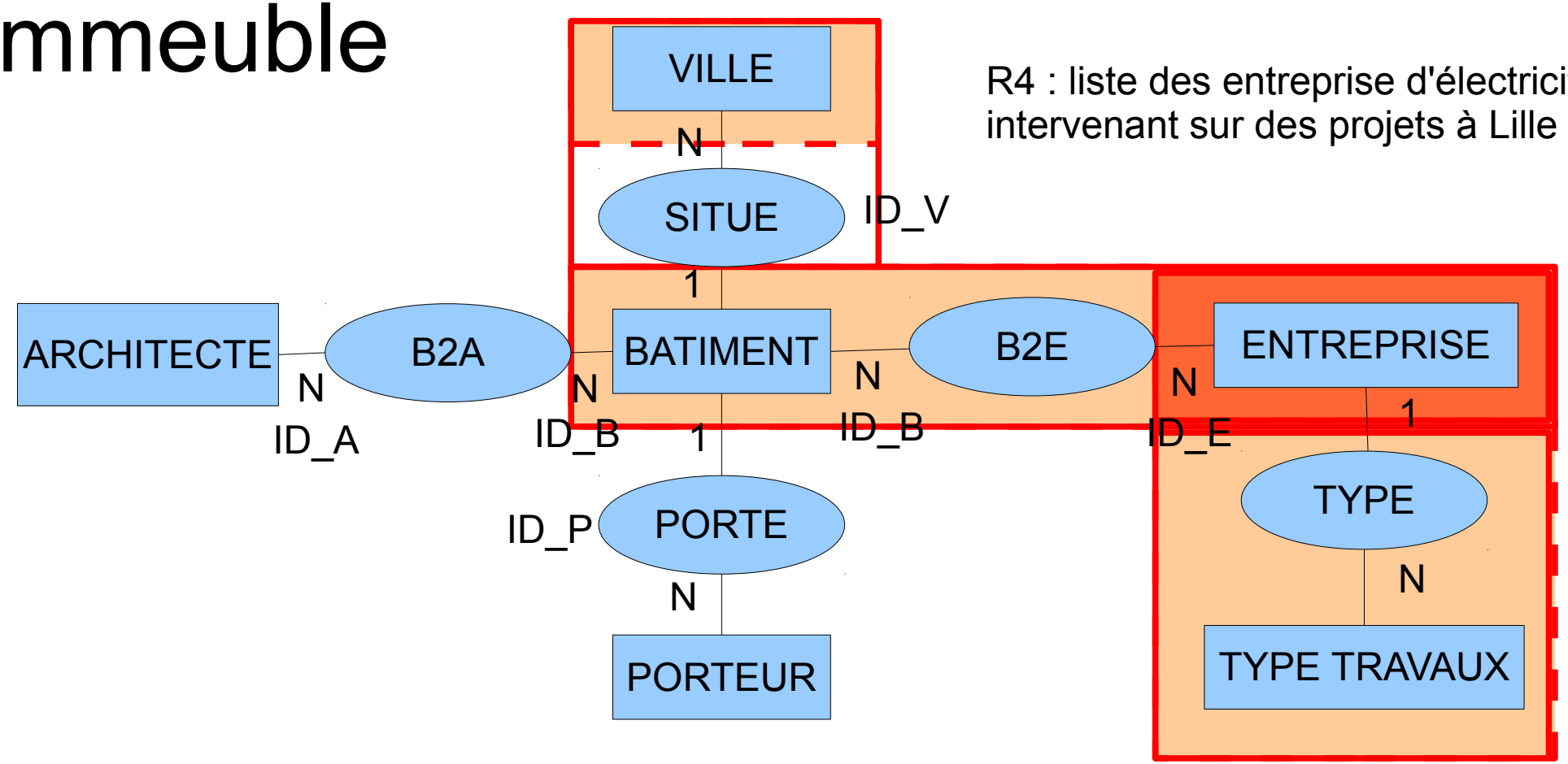
Immeuble



R4 : liste des entreprise d'électricité intervenant sur des projets à Lille

```
SELECT ENT.NOM FROM ENT, TT, B2E, BAT, VILLE
WHERE TT.NOM = 'Electricité' AND VILLE.NOM = 'Lille'
AND TT.ID_TT = ENT.ID_TT
AND ENT.ID_E = B2E_ID_E
```

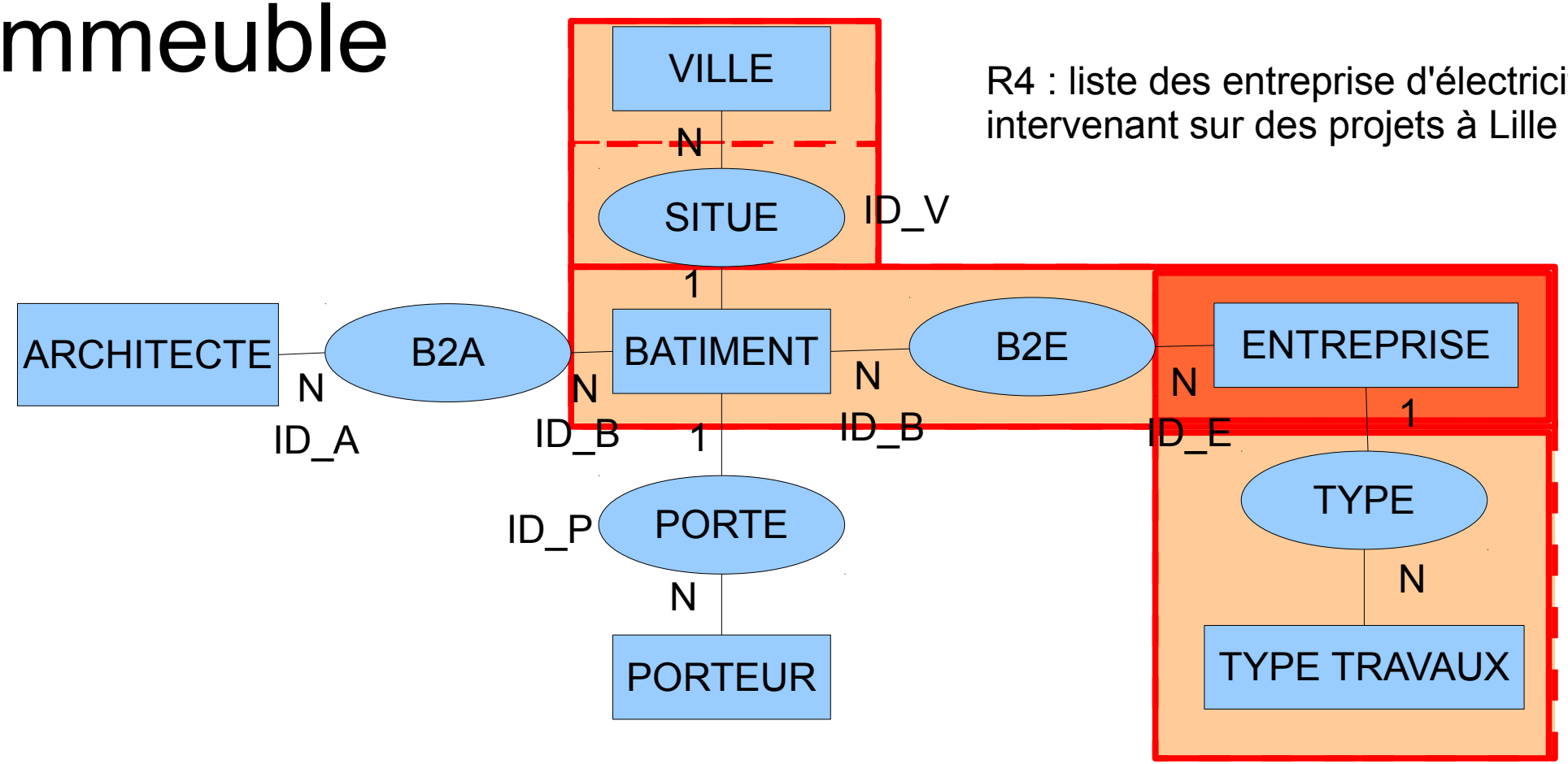
Immeuble



R4 : liste des entreprise d'électricité intervenant sur des projets à Lille

```
SELECT ENT.NOM FROM ENT, TT, B2E, BAT, VILLE
WHERE TT.NOM = 'Electricité' AND VILLE.NOM = 'Lille'
AND TT.ID_TT = ENT.ID_TT
AND ENT.ID_E = B2E_ID_E
AND B2E.ID_B = BAT.ID_B
```

Immeuble



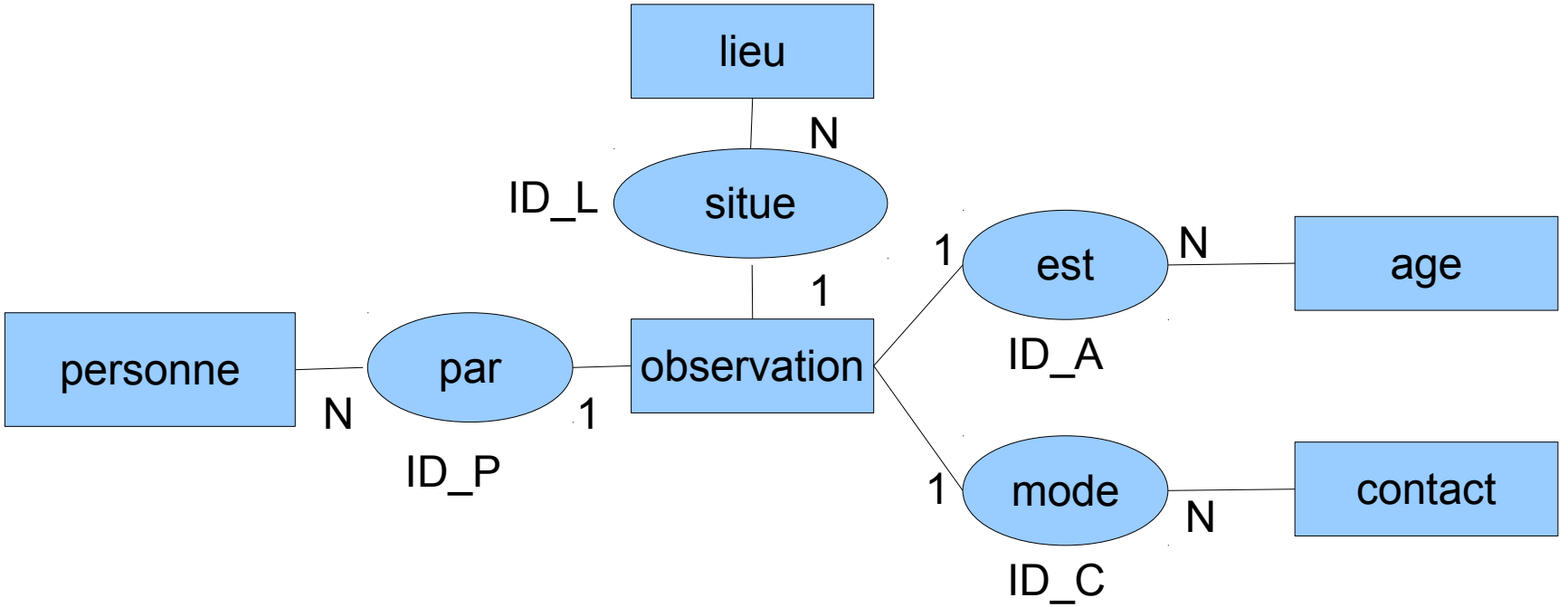
R4 : liste des entreprise d'électricité intervenant sur des projets à Lille

```
SELECT ENT.NOM FROM ENT, TT, B2E, BAT, VILLE
WHERE TT.NOM = 'Electricité' AND VILLE.NOM = 'Lille'
AND TT.ID_TT = ENT.ID_TT
AND ENT.ID_E = B2E_ID_E
AND B2E.ID_B = BAT.ID_B
AND BAT.ID_V = VILLE.ID_V
```

ANNEXES

4 – Étude des lérots

Lérots



- PERSONNE(ID_PERSONNE, NOM)
- LIEU(ID_LIEU, COMMUNE, DEP)
- AGE(ID_AGE, NOM)
- CONTACT(ID_CONTACT, NOM)
- OBSERVATION(ID_O, ID_P, ID_L, ID_C, ID_A, DATE, EFFECTIF)

Observation de Lérots

- 1) Liste des observateurs

  SELECT NOM FROM PERSONNE

Observation de Lérots

- 2) Dates des observations réalisées à Lille

```
SELECT O.DATE
FROM OBSERVATION O, LIEU L
WHERE L.COMMUNE = 'LILLE'
AND L.ID_LIEU = O.ID_LIEU
```

Observation de Lérots

- 3) Effectif pour chaque observation visuelle

```
SELECT O.EFFECTIF
FROM OBSERVATION O, CONTACT C
WHERE C.NOM = 'VISU'
AND C.ID_CONTACT = O.ID_CONTACT
```

Observation de Lérots

- 4) Liste des personnes ayant observé des adultes en 2006

```
SELECT P.NOM
FROM PERSONNE P, OBSERVATION O, AGE A
WHERE A.NOM = 'ADULT'
AND YEAR(O.DATE) = 2006
AND O.ID_AGE = A.ID_AGE
AND O.ID_PERSONNE = P.ID_PERSONNE
```

Observation de Lérots

- 5) Tableau identique à celui donné en énoncé

  SELECT P.NOM, O.DATE, L.COMMUNE, L.DEP, A.NOM, O.EFFECTIF, C.NOM

  FROM PERSONNE P, OBSERVATION O, LIEU L, AGE A, CONTACT C

  WHERE O.ID_PERSONNE = P.ID_PERSONNE

  AND O.ID_LIEU = L.ID_LIEU

  AND O.ID_CONTACT = C.ID_CONTACT

  AND O.ID_AGE = A.ID_AGE

Observation de Lérots

- 6) Nombre d'observateurs

  SELECT COUNT(NOM) FROM PERSONNE

Observation de Lérots

- 7) Nombre d'observations sans données pour l'age

  Codage des données manquantes :

  NA (Not Available) ou Missing Value ou Non Renseigné

```
SELECT COUNT( A.NOM)
FROM OBSERVATION O, AGE A
WHERE A.NOM = 'NA'
AND A.ID_AGE = O.ID_AGE
```

Observation de Lérots

- 8) Nombre de Lérots observés par département

```
SELECT SUM(O.EFFECTIF), L.DEPARTEMENT
FROM OBSERVATION O, LIEU L
WHERE  O.ID_LIEU = L.ID_LIEU
GROUP BY L.DEPARTEMENT
```

Observation de Lérots

- 9) Nombre de Lérots observés par département et par age

```
SELECT SUM(O.EFFECTIF), L.DEPARTEMENT, A.NOM
FROM OBSERVATION O, LIEU L, AGE A
WHERE  O.ID_LIEU = L.ID_LIEU
AND O.ID_AGE = A.ID_AGE
GROUP BY L.DEPARTEMENT, A.NOM
```