Chapter 01 /Part 03: Transition from Logical to SQL 2

Dr. Djakhdjakha L.

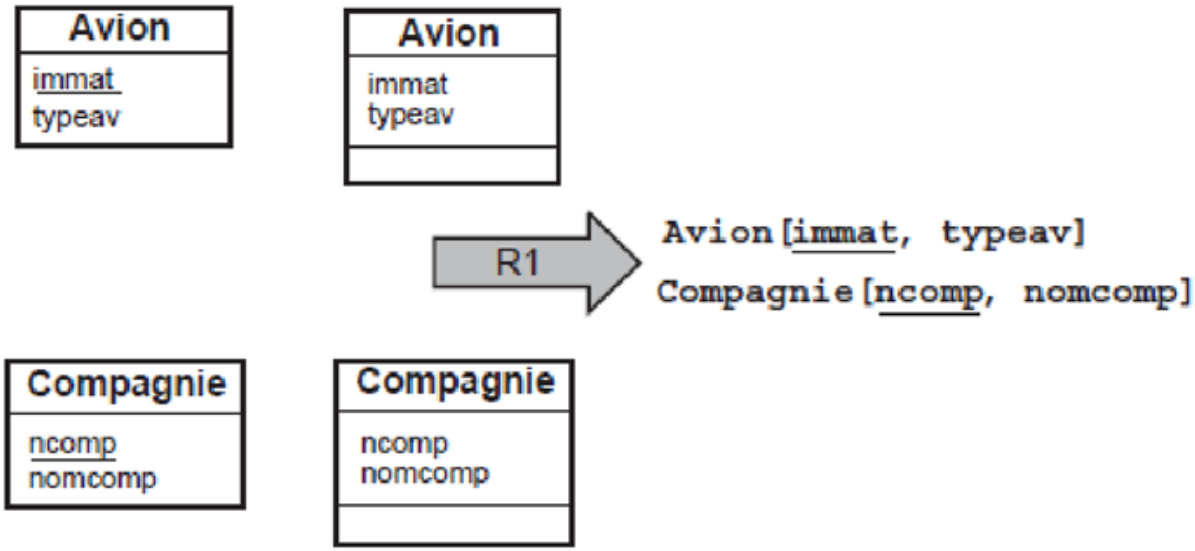
March 2025

Transition from Logical to SQL 2

- **Principle**
- A relation becomes a table, and its attributes become the columns of the table.
- The primary key is defined on the columns translated from the relation's identifier using the PRIMARY KEY constraint.

Entity/Class Transformation

- **Rule 1:**
- Each entity becomes a relation.
- The entity identifier becomes the primary key of the relation.
- Each class in the UML diagram becomes a relation.



- Select an attribute of the class to act as an identifier.
- If no attribute is suitable, add one to ensure the relation has a primary key.

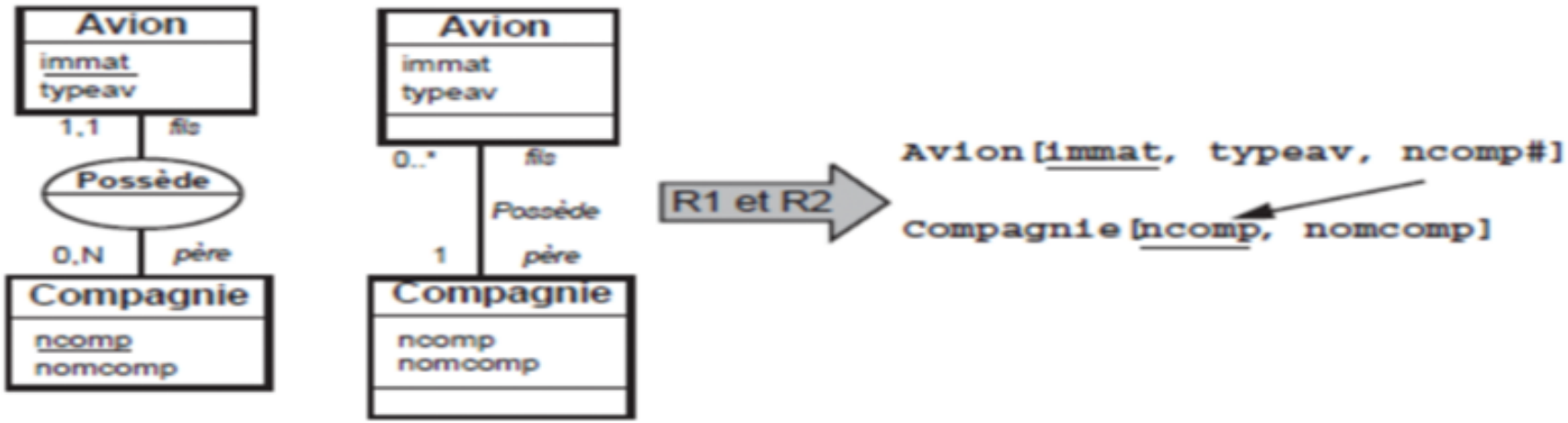
Entity/Class Transformation

```
1  create table pilote
2  (brevet varchar(8),
3  nom varchar(30),
4  age int,
5  constraint pk_pilote primary key(brevet)
6  );
```

Association Transformation

- Foreign keys are translated using FOREIGN KEY... REFERENCES... constraints.

One-to-Many Associations

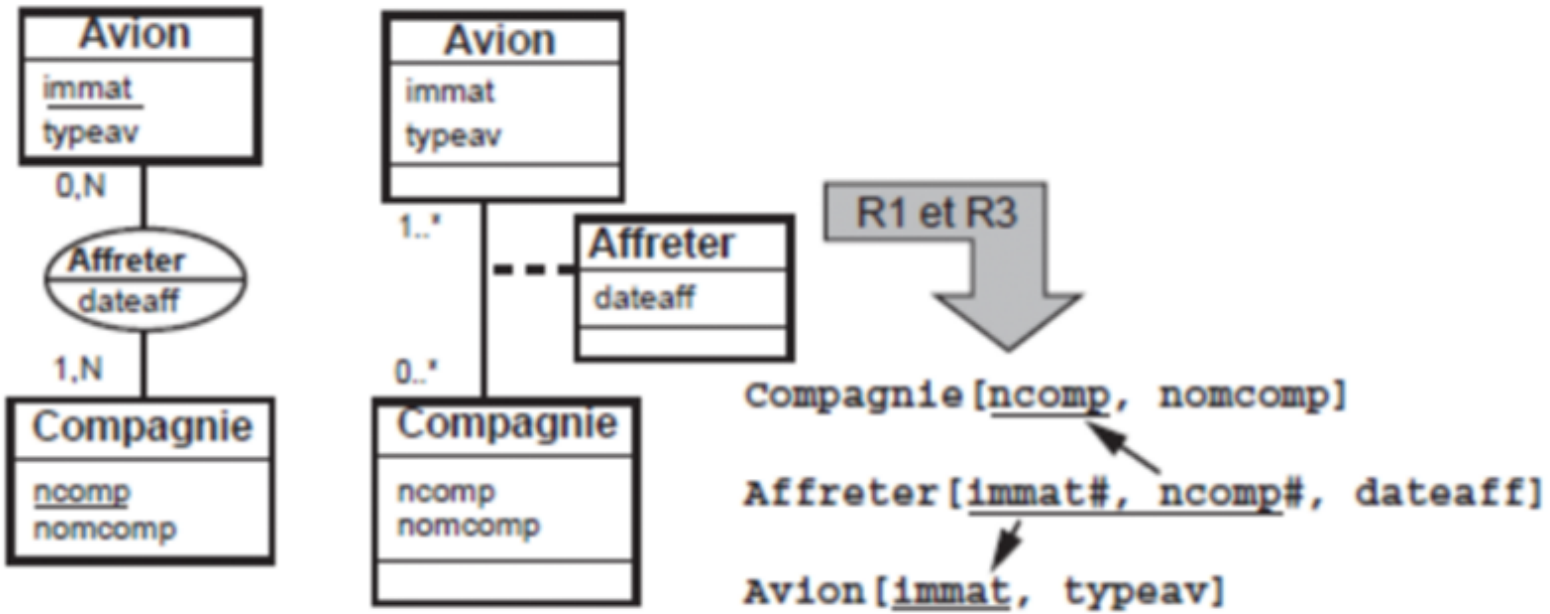


One-to-Many Associations

```sql
create table compagnie
(comp varchar(4),
nomcomp varchar(30),
constraint pk_compagnie primary key (comp)
);

create table avion
(na varchar(2),
typea varchar (4),
cap int,
comp varchar(4),
constraint pk_avion primary key (na),
constraint fk_avion_comp_compagnie foreign key (comp) references compagnie (comp),
constraint nn_avion_comp check (comp is not null)
);
```

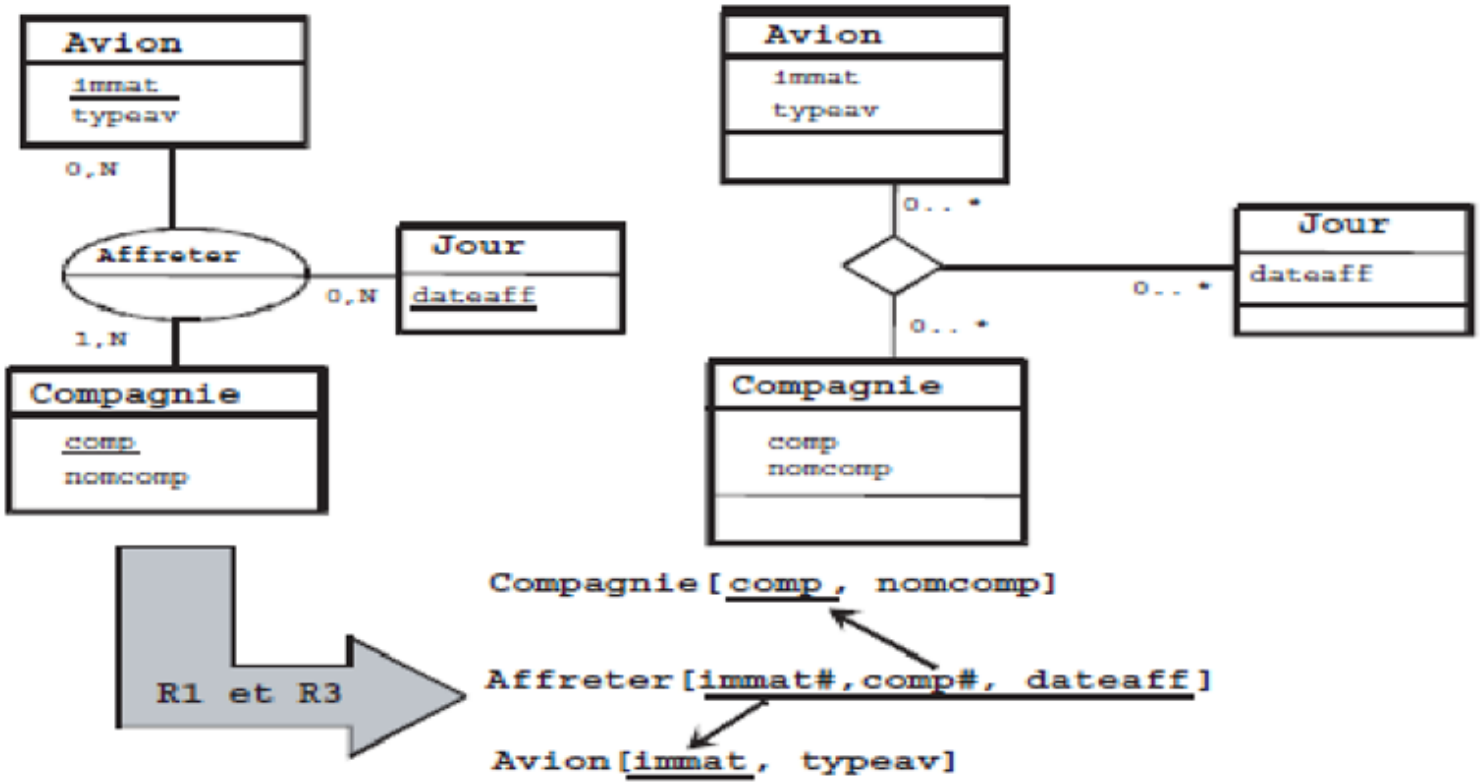
Many-to-Many and N-ary Associations



Many-to-Many and N-ary Associations

```
1  create table compagnie
2  (comp varchar(4), nomcomp varchar(30),
3  constraint pk_compagnie primary key (comp)
4  );
5  create table avion
6  (immat varchar(6), typav varchar (10),
7  constraint pk_avion primary key (immat)
8  );
9  create table affreter
10 (immat varchar(6), comp varchar(4), dateaff date,
11 constraint pk_affreter primary key (immat, comp),
12 constraint fk_affreter_immat_avion foreign key (immat) references avion (immat),
13 constraint fk_affreter_comp_compagnie foreign key (comp) references compagnie (comp)
14 );
```

Many-to-Many and N-ary Associations

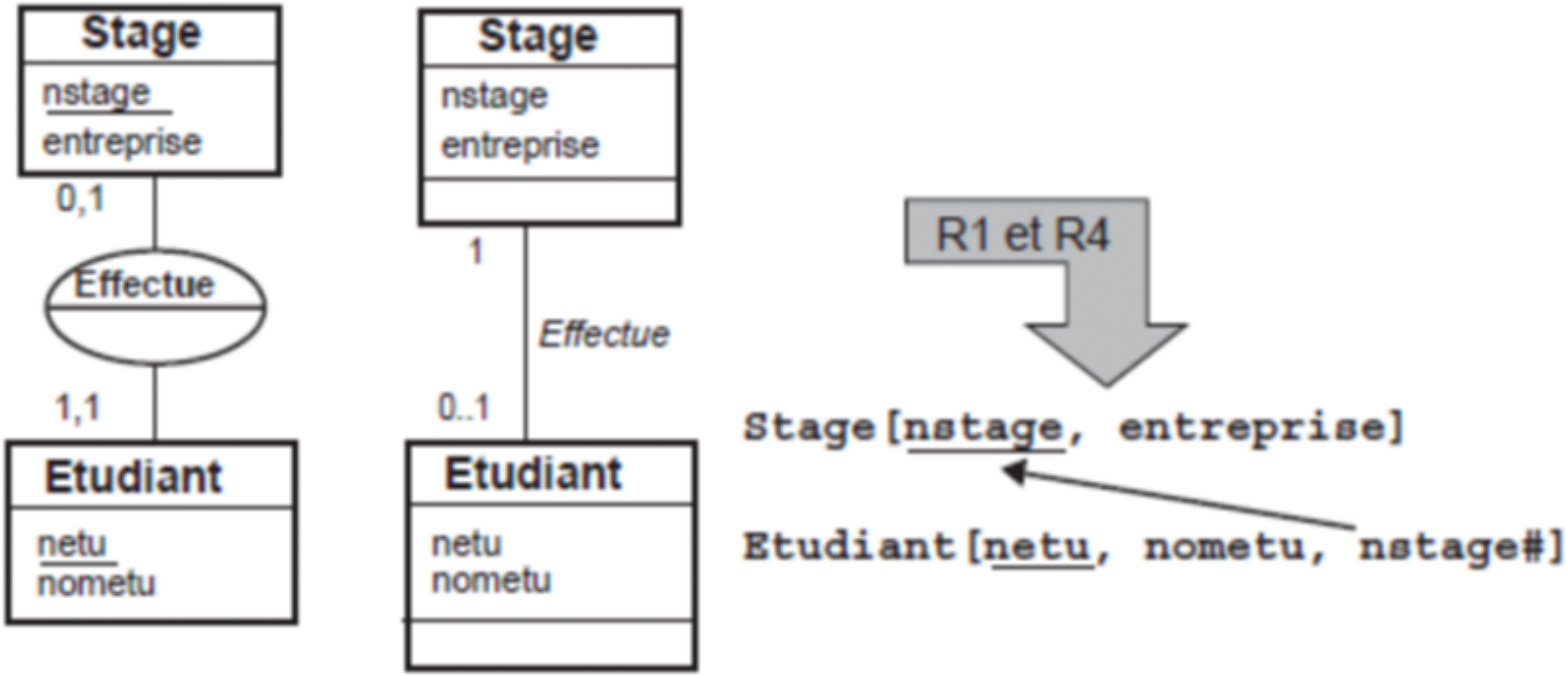


Many-to-Many and N-ary Associations

```
1  create table compagnie
2  (comp varchar(4), nomcomp varchar(30),
3  constraint pk_compagnie primary key (comp)
4  );
5  create table avion
6  (immat varchar(6), typav varchar (10),
7  constraint pk_avion primary key (immat)
8  );
9  create table affreter
10 (immat varchar(6), comp varchar(4), dateaff date,
11 constraint pk_affreter primary key (immat, comp, dateaff),
12 constraint fk_affreter_immat_avion foreign key (immat) references avion (immat),
13 constraint fk_affreter_comp_compagnie foreign key (comp) references compagnie (comp)
14 );
```

One-to-One Associations

- **Rule 4:**
- To avoid NULL values in the database, add a foreign key attribute in the relation derived from the entity with a minimum cardinality of zero.
- If both minimum cardinalities are zero, a choice is made between the two relations.
- If both are one, merging the two entities (classes) is recommended.



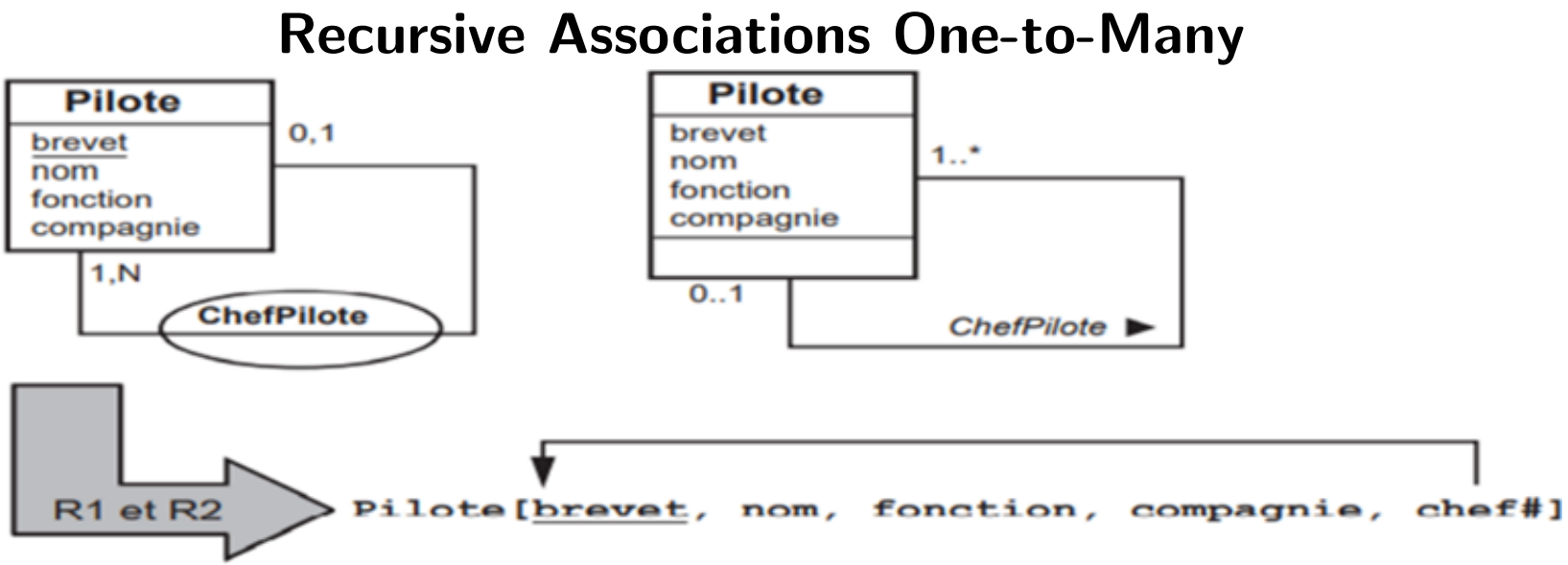
One-to-One Associations

- It is preferable to place the foreign key in the **Student** table to avoid null values in the database.
- The specificity of the SQL script lies in the definition of the last two constraints, which represent the **two minimum cardinalities (both at 1)** of the **Effectue** association.
- These constraints express the following:
  - A student must be assigned to only **one internship** (**NOT NULL** constraint).
  - An internship can be assigned to only **one student** (**UNIQUE** constraint).
- The **NOT NULL** constraint prevents null values in the *nstage* column.
- The **UNIQUE** constraint ensures that a value in the *nstage* column cannot appear for multiple students.

One-to-One Associations

```
1  create table stage
2  (nstage varchar(4), entreprise varchar(30),
3  constraint pk_stage primary key(nstage));
4  create table etudiant
5  (netu varchar(2), nometu varchar(30),
6  nstage varchar(4),
7  constraint pk_etudiant primary key (netu),
8  constraint fk_etudiant_nstage_stage foreign key (nstage) references stage(nstage),
9  constraint nn_etudiant_nstage check (nstage is not null),
10 constraint unique_etudiant_nstage Unique (nstage));
```

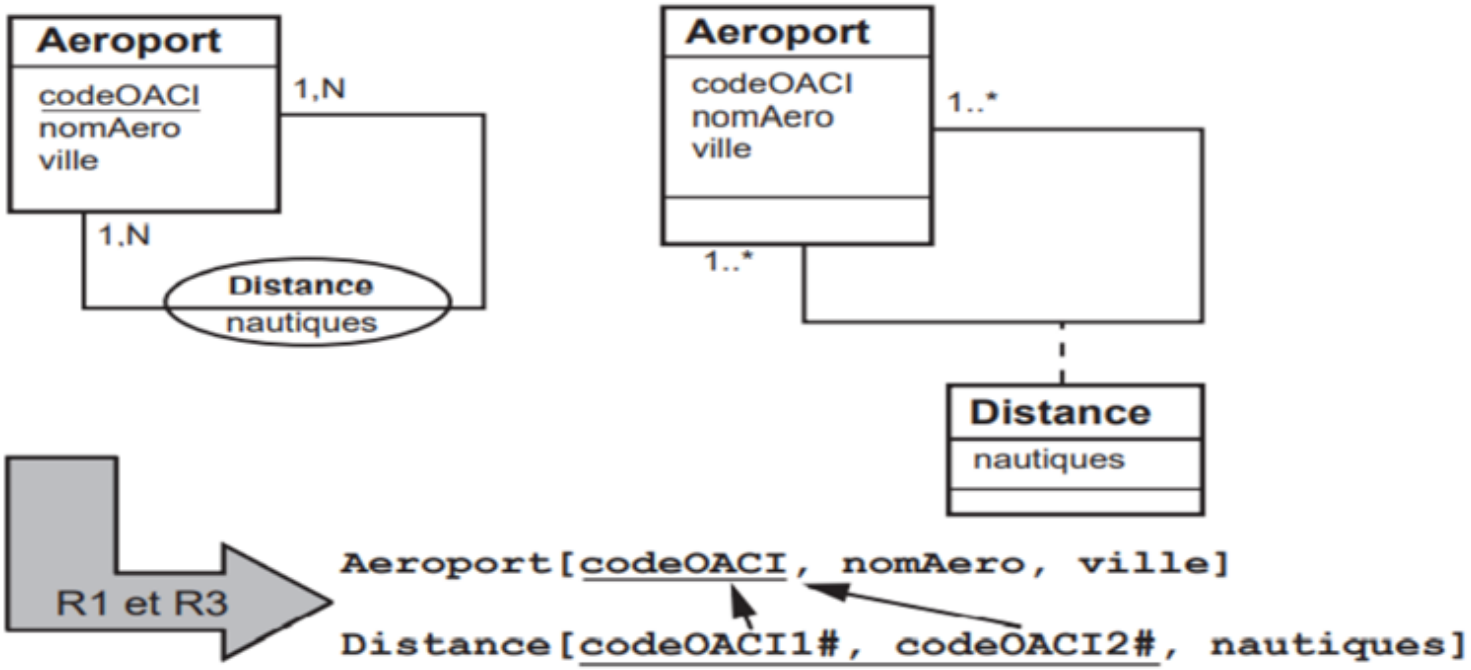
Recursive Associations



```sql
create table pilote
(brevet varchar(8),
nom varchar(30),
fonction varchar(4),
compagine varchar(4),
chef varchar(8),
constraint pk_pilote primary key (brevet),
constraint fk_pilote_chef_pilote foreign key(chef) references pilote (brevet)
);
```

Recursive Associations

Recursive Associations Many-to-Many



Frame Title

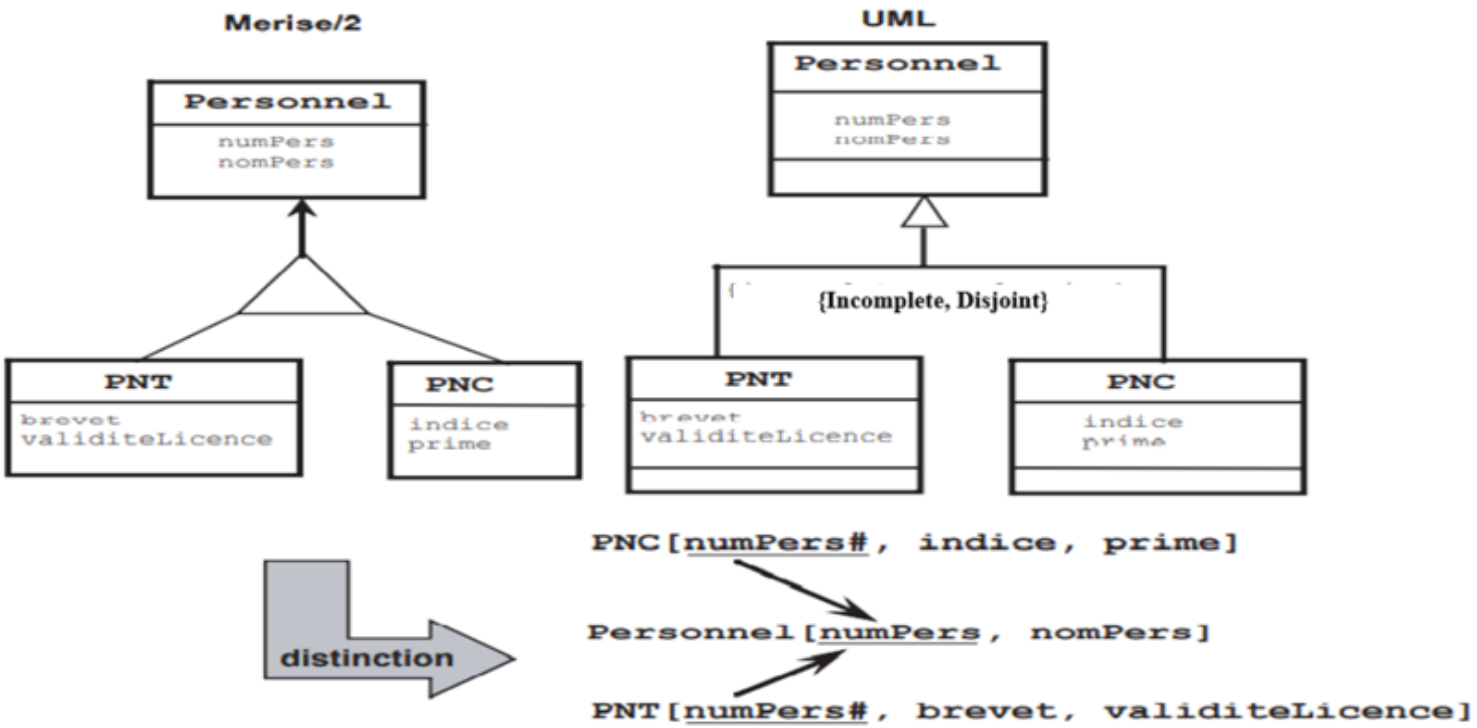
```sql
create table aeroport
(codeOACI varchar(8),
nomAero varchar(30),
ville varchar(20),
constraint pk_Aeroport primary key (codeOACI)
);
create table distance
(OACI1 varchar(8), OACI2 varchar(8),
nautiques int,
constraint pk_distance primary key (OACI1, OACI2),
constraint fk_distance_aeroport1 foreign key (OACI1) references aeroport (codeOACI),
constraint fk_distance_aeroport2 foreign key (OACI2) references aeroport (codeOACI)
);
```

Inheritance Transformation

- Decomposition by Distinction
- Push-down Decomposition
- Push-up Decomposition

Decomposition by Distinction

- Transform the superclass into a relation.
- Transform each subclass into a relation.
- The primary key of the superclass migrates into the subclass relation(s) and becomes both a primary and foreign key.

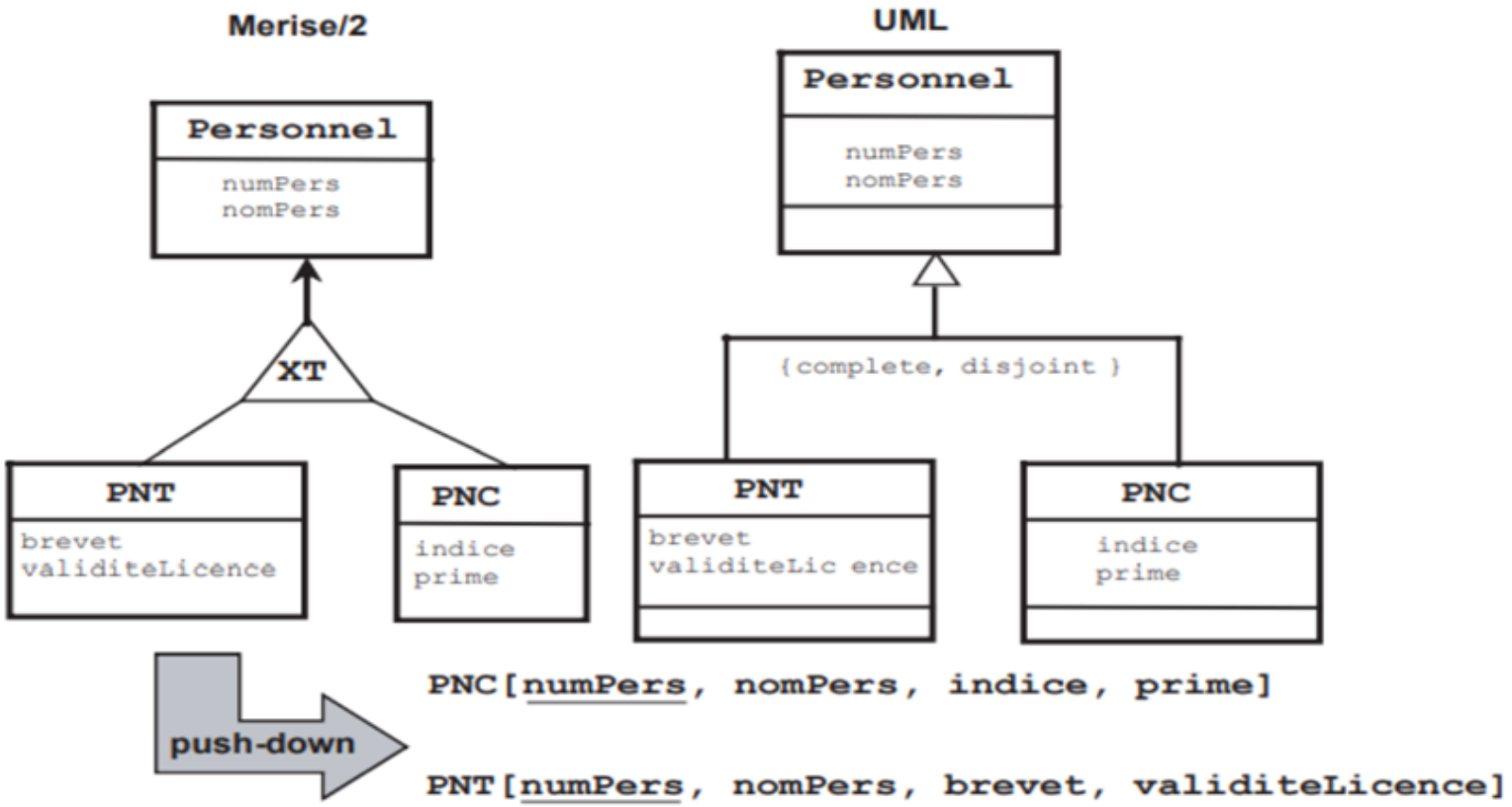


SQL2

```
1  create table personnel
2  (numPers int, nomPers varchar(20),
3  constraint pk_Personnel primary key (numPers)
4  );
5  drop table PNC;
6  create table PNC
7  (numPers int, indice int, prime float,
8  constraint pk_PNC primary key (numPers),
9  constraint fk_PNC_Personnel foreign key (numPers) references personnel (numPers)
10 );
11 create table PNT
12 (numPers int, brevet varchar(10),
13 validiteLicence Date,
14 constraint pk_PNT primary key(numPers),
15 constraint fk_PNT_Personnel foreign key (numPers) references personnel (numPers)
16 );
```

Push-down Decomposition

- If a totality or partition constraint exists, the superclass relation may not be translated.
- All attributes migrate into the subclass relation(s).



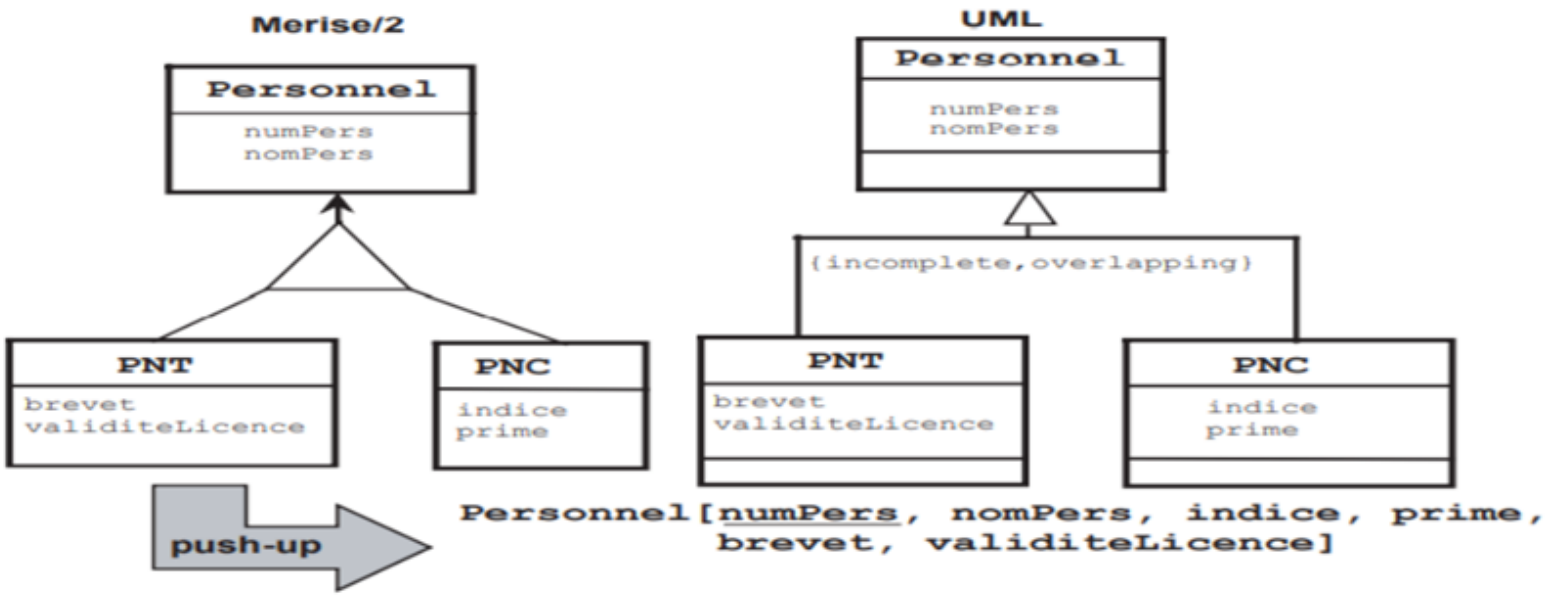
SQL2

```sql
create table PNC
(numPers int, nomPers varchar(20),
indice int, prime float,
constraint pk_PNC primary key(numPers)
);

create table PNT
(numPers int, validiteLicense date,
constraint pk_PNT primary key(numPers)
);
```

Push-up Decomposition

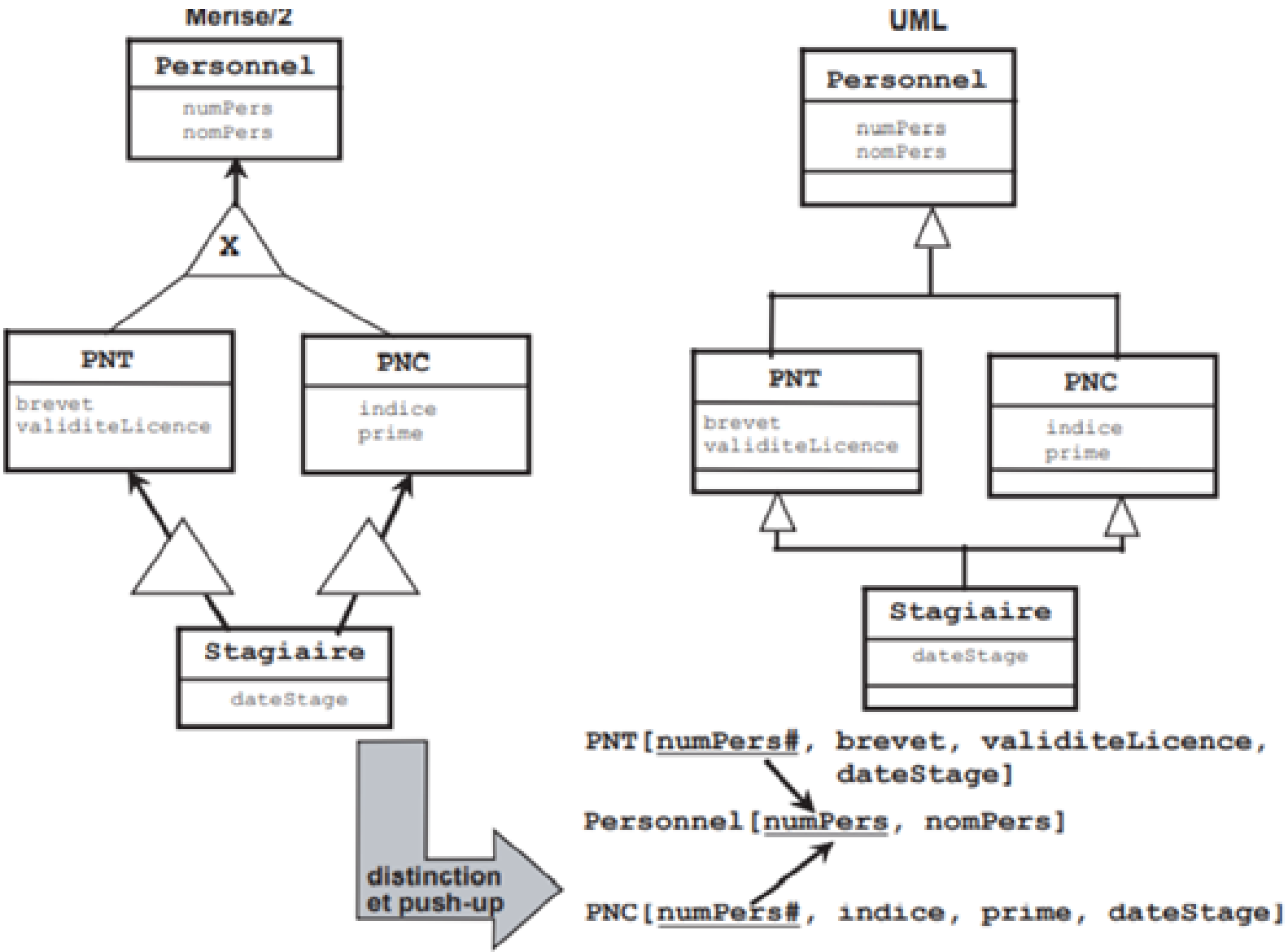
- Remove subclass relations and migrate their attributes into the superclass relation.



SQL2

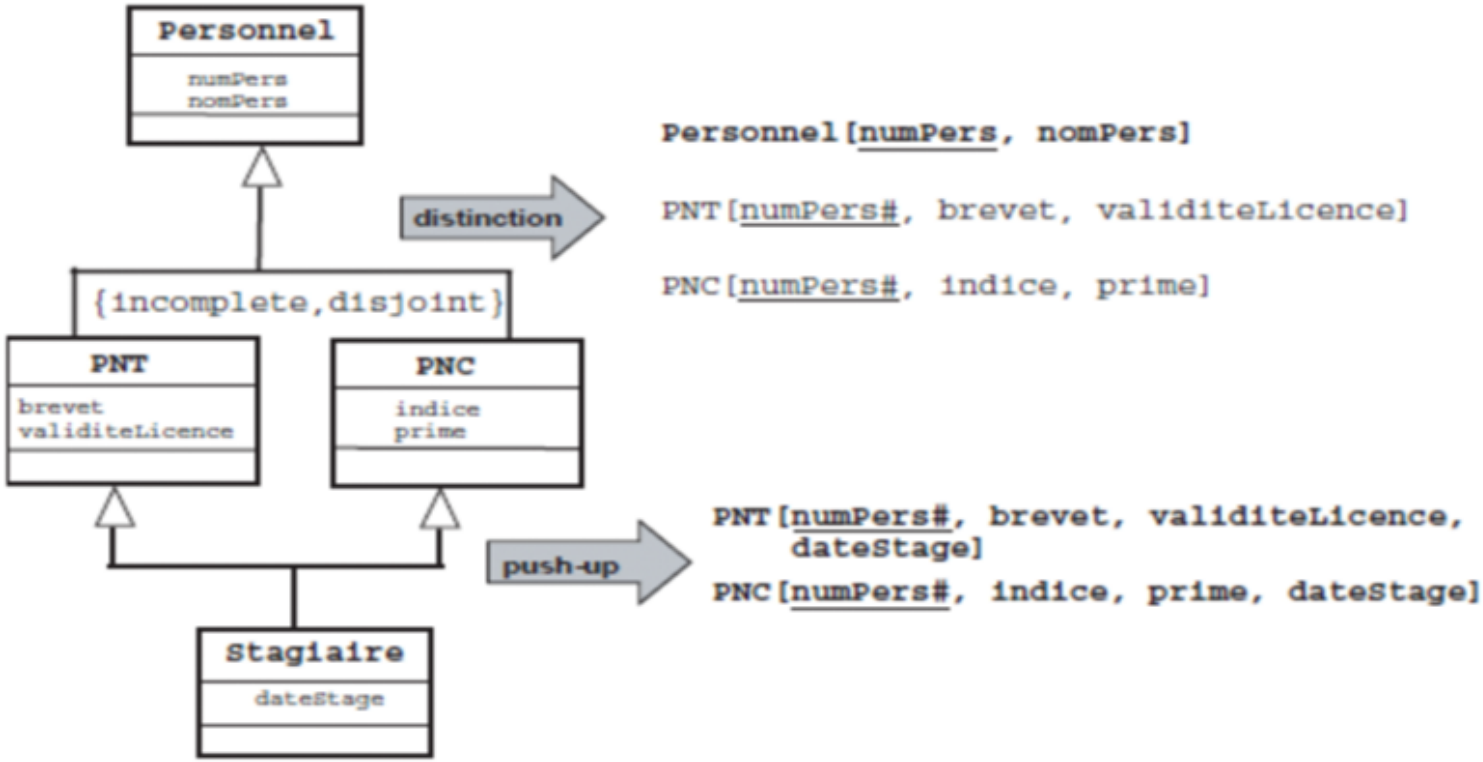
```sql
create table Personnel
(numPers int, nomPers varchar(20),
indice int, prime float,
brevet varchar(10),
validiteLicense date,
constraint pk_Personnel primary key(numPers)
);
```

Multiple Inheritance



Multiple Inheritance

Transformation of multiple inheritance associations.



SQL2

```sql
create table Personnel
(numPers int, nomPers varchar(20),
constraint pk_Personnel primary key(numPers)
);
create table PNC
(numPers int, nomPers varchar(20),
indice int, prime float, datesatge date,
constraint pk_PNC primary key(numPers),
constraint fk_PNC_Personnel foreign key(numPers) references Personnel (numPers)
);

create table PNT
(numPers int, brevet varchar(10),
validiteLicense date, datesatge date,
constraint pk_PNT primary key(numPers),
constraint fk_PNT_Personnel foreign key(numPers) references Personnel (numPers)
);
```