Vers le modèle physique

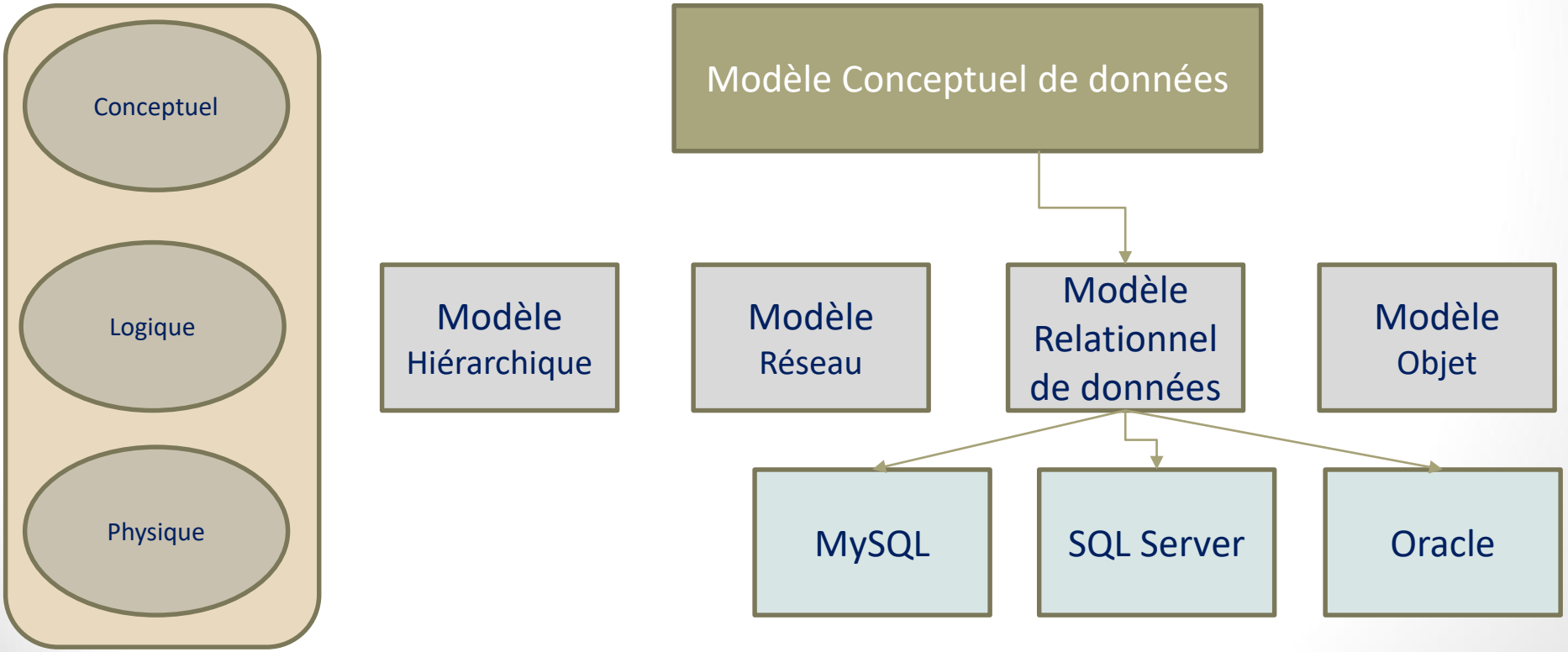
Principes de dérivation

Plan du cours

- Du MCD au modèle physique
- Terminologie
- Principes de dérivation diagramme de classe → modèle physique

Le niveau logique

- Le niveau conceptuel est abstrait
- Avec le niveau logique, on se rapproche des tables tout en restant indépendant du SGBD



Quelques définitions

- Modèle relationnel de données : Déf Gilles Roy
- Un modèle logique de données spécifiant un schéma pour une base de données relationnelle soit : les tables, les champs de chaque table et leurs propriétés, la clé primaire des tables, les clés étrangères assurant les liaisons entre les tables et les contraintes d'intégrité portant sur ces liaisons

- Avantages
  - Indépendant du SGBD, tout en étant plus proche des tables
  - Etape de vérification / validation modèle
  - Typage et estimation volumétrie de la base
  - Précisions sur les contraintes, notamment UNIQUE et CHECK

Terminologie

Mathématique	Conceptuel	Base de données Relationnelle	Système de gestion de fichier
Relation	Entité	Table	Fichier
Tuple	Occurrence	Ligne	Enregistrement
Attribut	Attribut	Nom de colonne	Champ
Clé	Identifiant	Clé primaire	Clé enregistrement

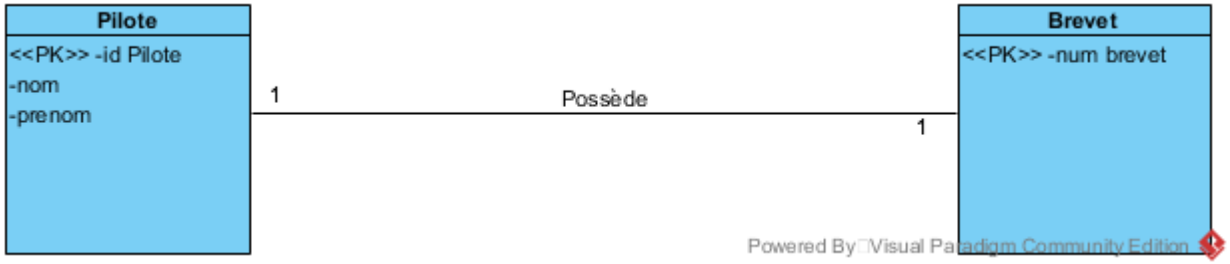
- Idéalement, on n'utilise pas les termes d'un niveau lorsqu'on travaille à un autre niveau (plus facile à dire qu'à faire …)

Principes de modélisation

- Un modèle de données relationnel doit être ***normalisé***
- Chaque entité (classe) UML donne lieu à une table physique
- Identifiant (diagramme classe) → clef primaire artificielle + clef métier
- Règles de dérivation des associations

Règles de dérivation des associations – 1 à 1

- 1 .. 1 aux 2 bouts



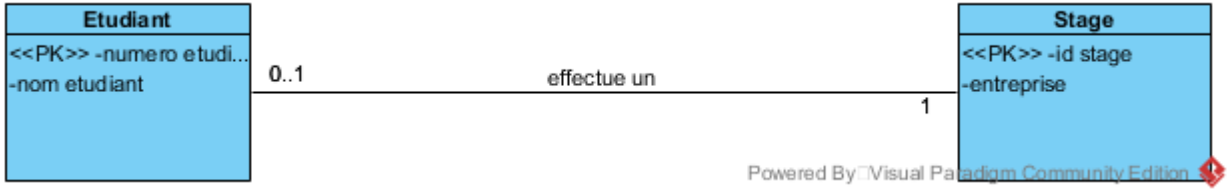

```
CREATE TABLE PILOTE
(
   CLEF_PILOTE INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
   CLEF_BREVET INT NOT NULL FOREIGN KEY REFERENCES BREVET(CLEF_BREVET),
   ID_PILOTE VARCHAR(30) ,
   NOM_PILOTE VARCHAR(30) ,
   PRENOM_PILOTE VARCHAR(30)
)
;
```

```
CREATE TABLE BREVET
(
   CLEF_BREVET INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
   NUMERO_BREVET VARCHAR(50)
)
;
```

Ici la clef étrangère peut être d'un côté ou de l'autre

Règles de dérivation des associations – 1 à 1

- 1 .. 1 d'un côté, 0.. 1 de l'autre




```
CREATE TABLE ETUDIANT
(
   CLEF_ETUDIANT INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
   CLEF_STAGE INT NOT NULL FOREIGN KEY REFERENCES STAGE(CLEF_STAGE),
   NUMERO_ETUDIANT VARCHAR(30) ,
   NOM_ETUDIANT VARCHAR(50)
)
;

INSERT INTO ETUDIANT  VALUES (1,'ETU_ENC_SIO_2018_SLAM_001','ROBERT');
INSERT INTO ETUDIANT  VALUES (NULL,'ETU_ENC_SIO_2018_SLAM_002','DUPOND');
```

```
CREATE TABLE STAGE
(
   CLEF_STAGE INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
   ID_STAGE VARCHAR(50) ,
   NOM_ENTREPRISE VARCHAR(50)
)
;

INSERT INTO STAGE  VALUES ('STAGE2018_09_01_001','Orange');
```

Ici la clef étrangère est du côté Etudiant (pour éviter valeurs nulles dans les données)

Règles de dérivation des associations – 1 à *

- 1 .. 1 ou 0..1 d'un côté, 0..* ou 1..* de l'autre


```
CREATE TABLE EMPLOYEUR
(
   ID_EMPLOYEUR INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
   RAISON_SOCIALE VARCHAR(60)
);
```

```
CREATE TABLE PERIODE_EMPLOI
(
   ID_PERIODE INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
   ID_EMPLOYEUR INT NOT NULL FOREIGN KEY REFERENCES EMPLOYEUR(ID_EMPLOYEUR),
   DATE_EMBAUCHE DATE,
   DATE_DEPART DATE
);
```

Ici la clef étrangère est du côté du 1..* (ou 0..*)

Règles de dérivation des associations – * à *

- 0..* ou 1..* d'un côté, 0..* ou 1..* de l'autre


```
CREATE TABLE POSTE (
CLEF_POSTE INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
NUM_AFFICHAGE VARCHAR(50) NOT NULL,
NOM_POSTE VARCHAR(60) ) ;
```

```
CREATE TABLE CANDIDAT (
CLEF_CANDIDAT INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
NUMERO_CANDIDAT VARCHAR(50) NOT NULL,
NOM_CANDIDAT VARCHAR(50) NOT NULL ,
PRENOM_CANDIDAT VARCHAR(50) ,
NUMERO_TELEPHONE VARCHAR(14),
DATE_NAISSANCE DATE
 );
```

```
CREATE TABLE CANDIDAT_POSTE (
CLEF_CANDIDAT_POSTE INT IDENTITY(1,1) NOT NULL ,
CLEF_CANDIDAT INT NOT NULL FOREIGN KEY REFERENCES CANDIDAT(CLEF_CANDIDAT),
CLEF_POSTE INT NOT NULL FOREIGN KEY REFERENCES POSTE(CLEF_POSTE) ) ;
```

Table d'association **obligatoire** (ici CANDIDAT_POSTE)

Règles de dérivation des associations – * à *

- 0..* ou 1..* d'un côté, 0..* ou 1..* de l'autre


```
CREATE TABLE POSTE (
CLEF_POSTE INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
NUM_AFFICHAGE VARCHAR(50) NOT NULL,
NOM_POSTE VARCHAR(60) ) ;
```

```
CREATE TABLE CANDIDAT (
CLEF_CANDIDAT INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
NUMERO_CANDIDAT VARCHAR(50) NOT NULL,
NOM_CANDIDAT VARCHAR(50) NOT NULL ,
PRENOM_CANDIDAT VARCHAR(50) ,
NUMERO_TELEPHONE VARCHAR(14),
DATE_NAISSANCE DATE
 );
```

```
CREATE TABLE CANDIDAT_POSTE_VARIANTE (
CLEF_CANDIDAT INT NOT NULL FOREIGN KEY REFERENCES CANDIDAT(CLEF_CANDIDAT),
CLEF_POSTE INT NOT NULL FOREIGN KEY REFERENCES POSTE(CLEF_POSTE) ,
CONSTRAINT PK_CANDIDAT_POSTE PRIMARY KEY (CLEF_CANDIDAT, CLEF_POSTE)
) ;
```

Table d'association **obligatoire** (ici CANDIDAT_POSTE)

Règles de dérivation des associations avec Classe d'association

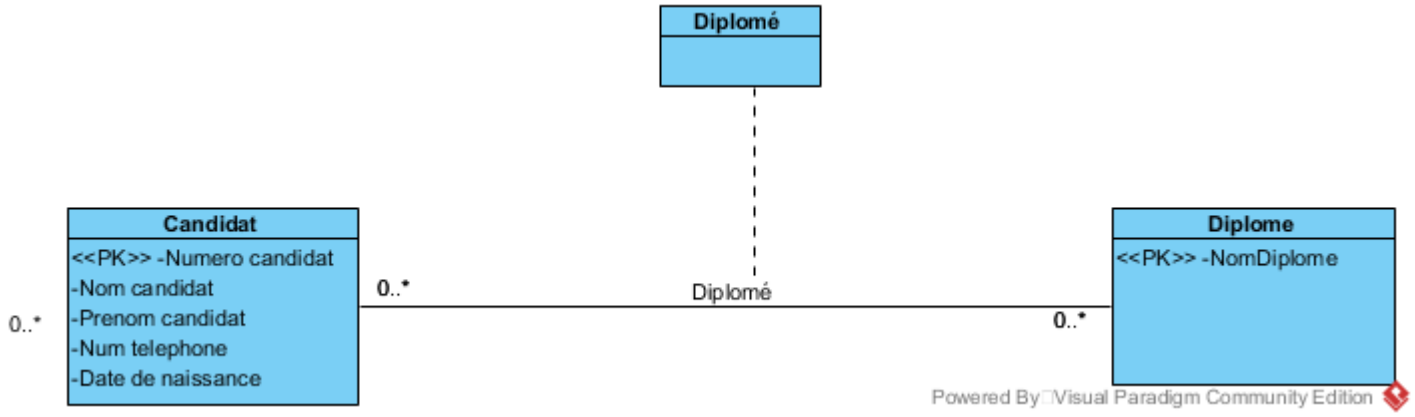


Table d'association **obligatoire,** déjà fournie par la classe Association (entité faible)

Règles de dérivation des associations avec Classe d'association

```
CREATE TABLE EST_DIPLOME (
CLEF_EST_DIPLOME INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
CLEF_DIPLOME INT NOT NULL FOREIGN KEY REFERENCES DIPLOME(CLEF_DIPLOME),
CLEF_CANDIDAT INT NOT NULL FOREIGN KEY REFERENCES CANDIDAT(CLEF_CANDIDAT)) ;
```

```
CREATE TABLE CANDIDAT (
CLEF_CANDIDAT INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
NUMERO_CANDIDAT VARCHAR(50) NOT NULL,
NOM_CANDIDAT VARCHAR(50) NOT NULL ,
PRENOM_CANDIDAT VARCHAR(50) ,
NUMERO_TELEPHONE VARCHAR(14),
DATE_NAISSANCE DATE
 );
```

```
CREATE TABLE DIPLOME (
CLEF_DIPLOME INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
NOM_DIPLOME VARCHAR(50) NOT NULL) ;
```

Table d'association **obligatoire,** déjà fournie par la classe Association (entité faible), ici EST_DIPLOME

Reste à voir

- Règles de dérivation concernant l'héritage
- Traduction diagramme de classe UML → langage Java ou PHP