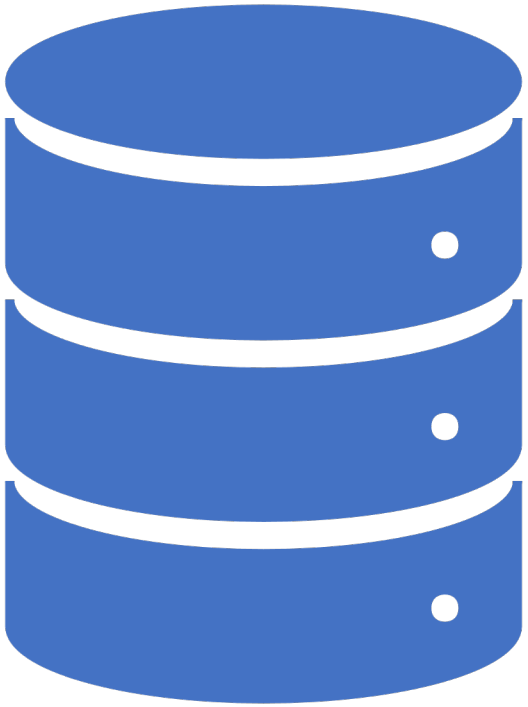


Seminar 1

Mapping Entity-Relationship Diagrams to Relational Schemas

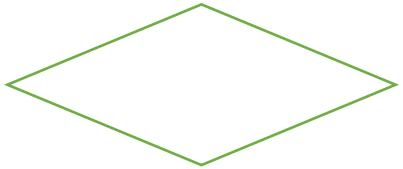
SQL – Data Definition Language (DDL)

Entity-Relationship (ER) Model

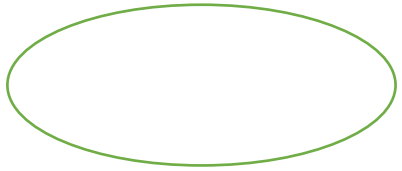
The Entity-Relationship (ER) diagram

- entity set

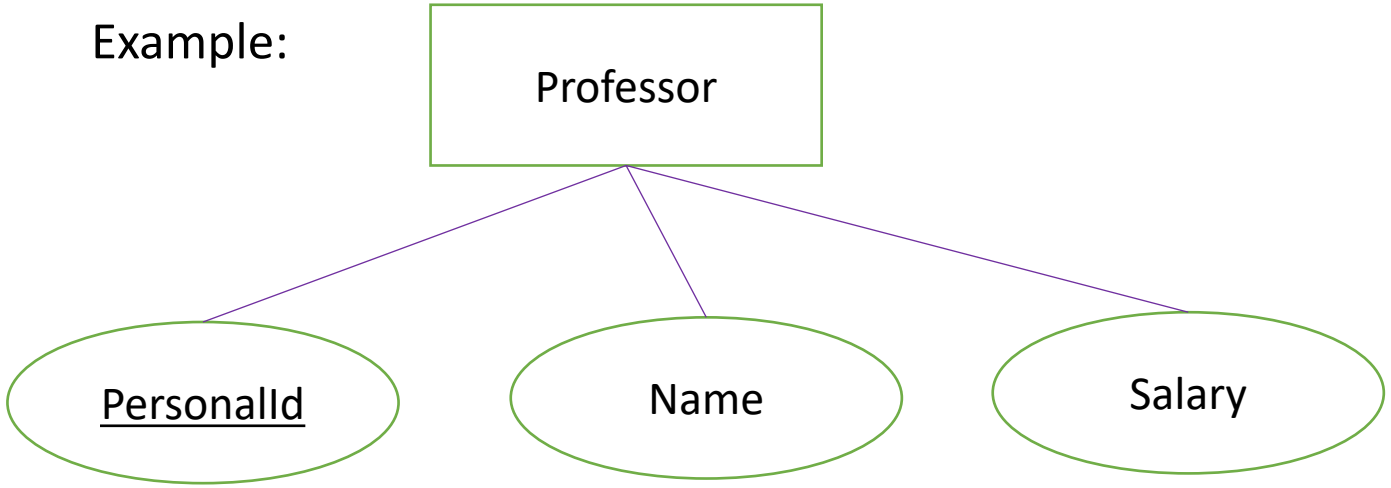
- relationship set



- attribute



Example:

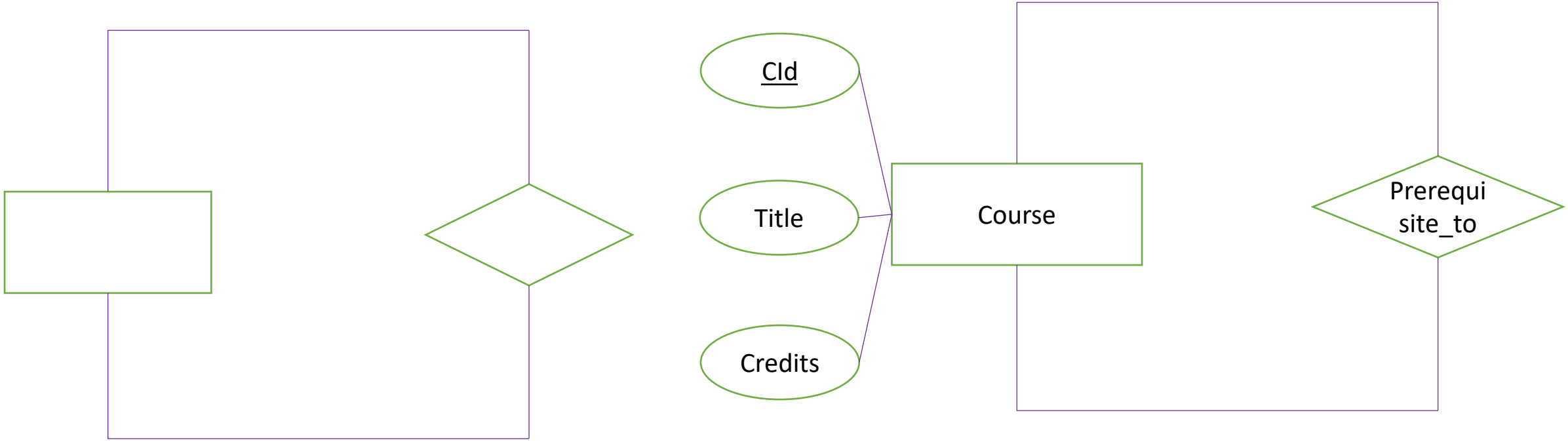


PersonalId – **primary key** for the *Professor* entity set

The degree of a relationship set

- the number of entity sets that participate in the relationship set

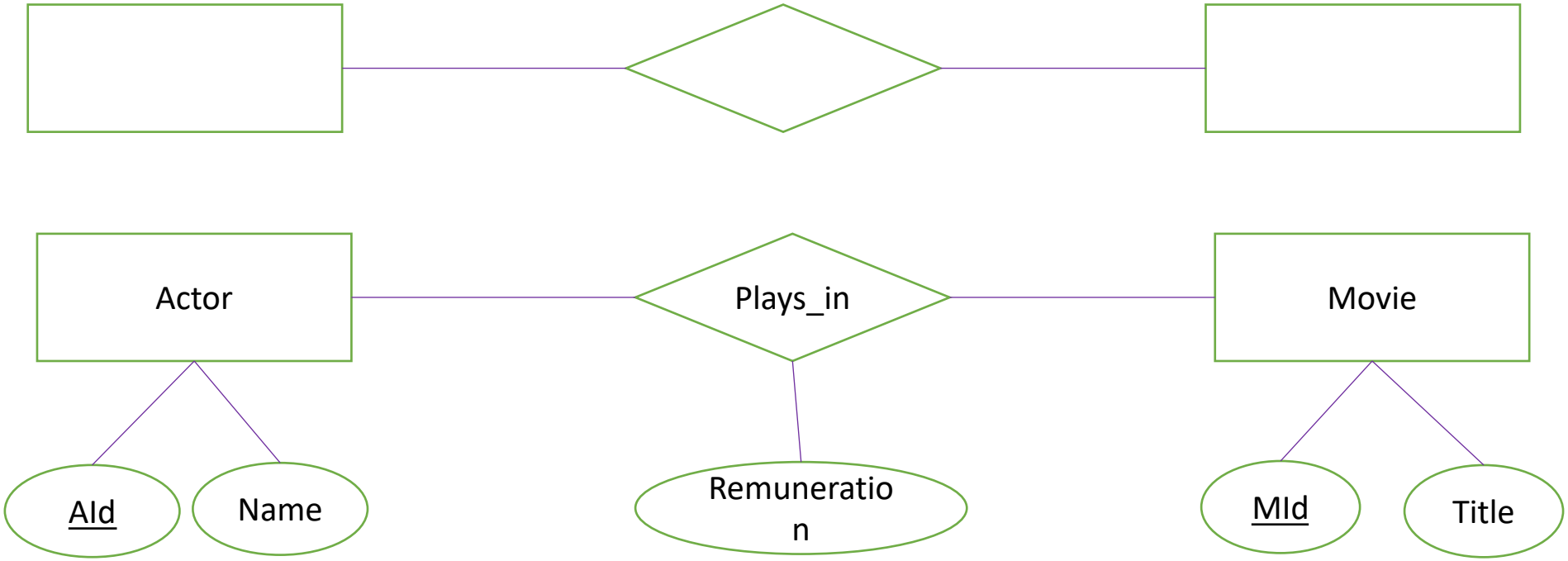
- Unary relationship set (degree = 1)



The degree of a relationship set

- the number of entity sets that participate in the relationship set

- Binary relationship set (degree = 2)

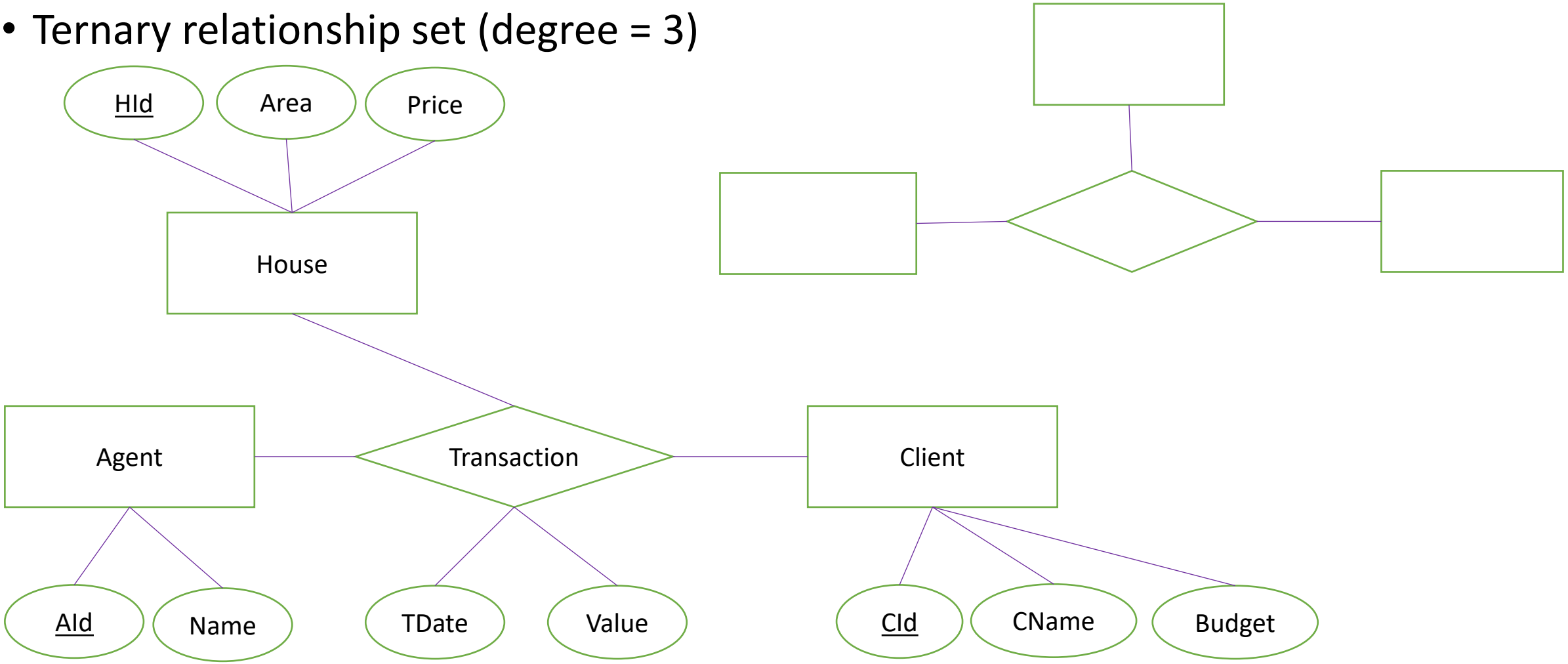


Remuneration – descriptive attribute for the *Plays_in* relationship set

The degree of a relationship set

- the number of entity sets that participate in the relationship set

- Ternary relationship set (degree = 3)

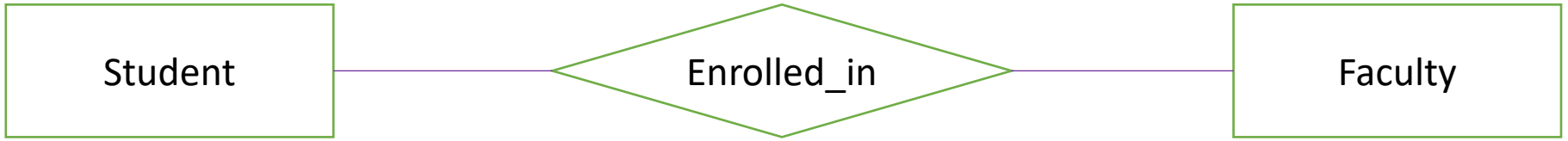


Mapping cardinalities – binary relationship sets

- 1 – 1 (1 to 1)

- 1 – n (1 to many)

- m – n (many to many)



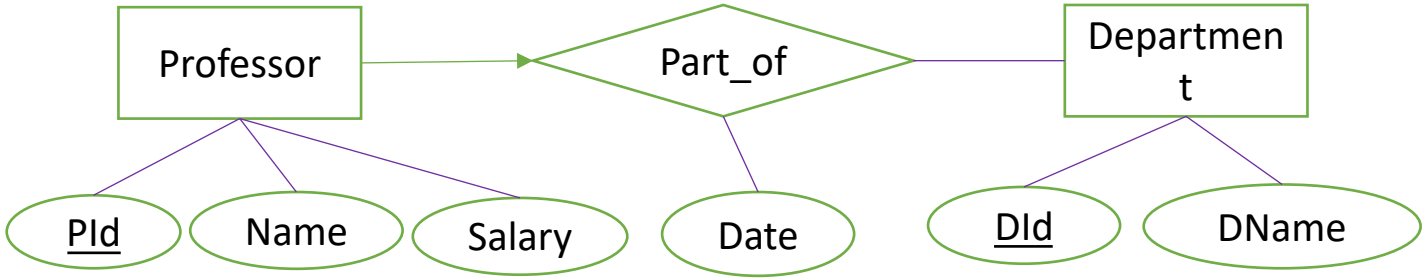
Relational Model

Translation from *Entity-Relationship (ER)* model to *Relational* model

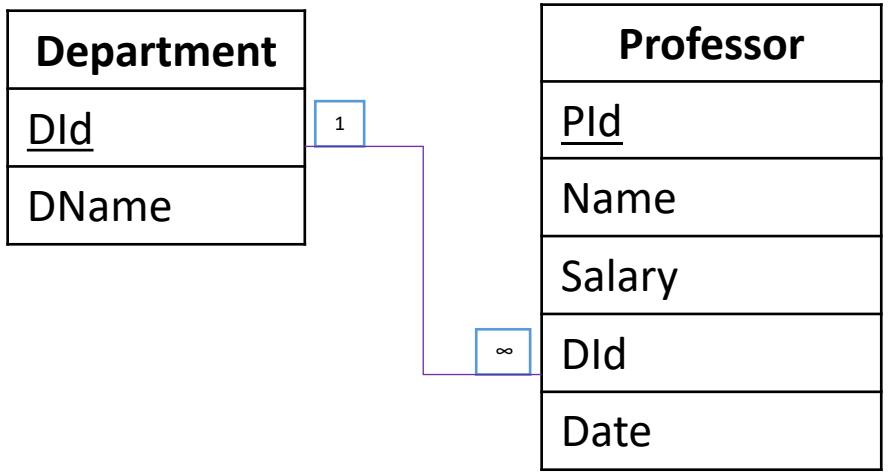
- entity set → relation

- The *name of the entity set* becomes the ***name of the relation***.
- The *attributes of the entity set* become the ***attributes of the relation***.
- The *primary key of the entity set* becomes the ***primary key of the relation***.

1 – n (or n - 1) relationship set → relation

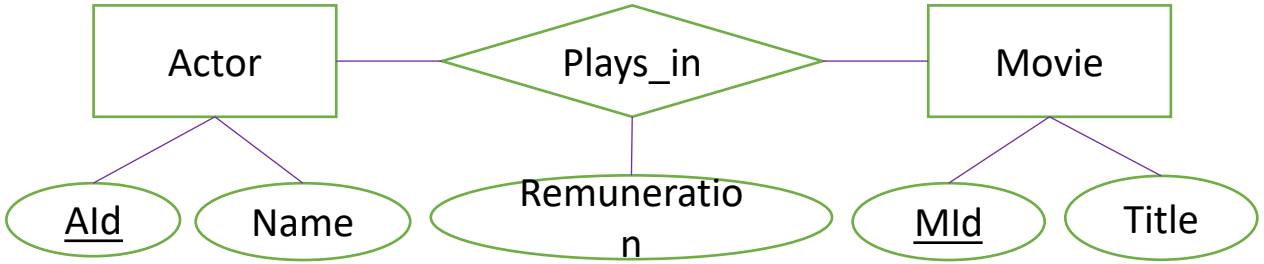


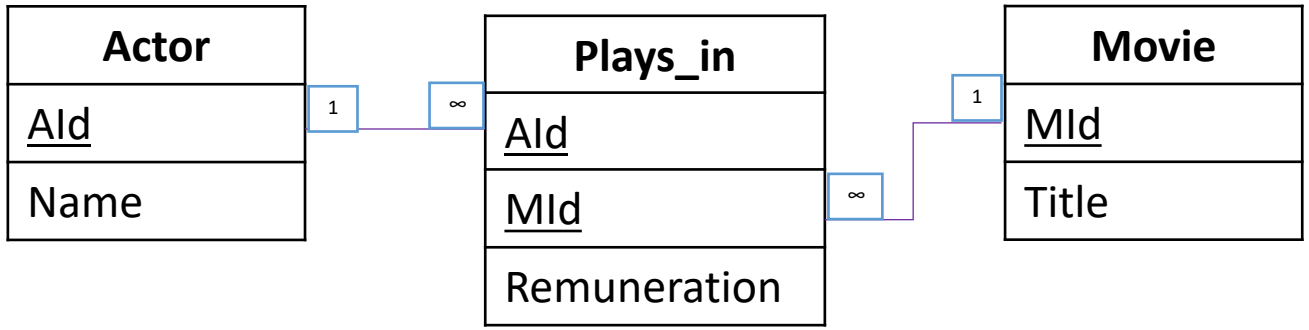
- No need of an additional relation.



- The *Professor* entity set is the *n (many)* side of the *Part_of* relationship set and the *Department* entity set is the *1* side of the *Part_of* relationship set.
- The relationship set data is included in the *Professor* relation (corresponding to the entity set that lies on the *n* side of the relationship set); the relation will store every *professor's department* along with the *Data* when he/she started being *part of it*.
- The primary key in the *Department* relation (corresponding to the entity set that lies on the *1* side of the relationship set) becomes a *foreign key* in the *Professor* relation.

m – n relationship set → relation





- The *name of the relationship set* becomes the *name of the relation* (e.g. Actor, Plays_in, Movie).
- The *relationship set descriptive attributes* become *attributes in the relation* (e.g. Name, Remuneration, Title).
- The *primary key attributes from each entity set*, that are part of the relationship set (e.g. AId, MId):
  - become *attributes in the relation*;
  - are *foreign keys in the relation*;
  - can become the *primary key of the (intermediate) relation*.

SQL – Data Definition Language (DDL)

- DDL contains commands to create / modify / delete databases, tables and indexes, establish relationships between tables and constraints.
- Examples for databases: CREATE / ALTER / DROP DATABASE
- Examples for tables: CREATE / ALTER / DROP TABLE
- Examples for indexes: CREATE / ALTER / DROP INDEX

- **CREATE DATABASE** – allows to create a database
  - ***CREATE DATABASE database_name***
    - Example: CREATE DATABASE Cinema;
- **ALTER DATABASE** – allows to modify a database
    - Example: ALTER DATABASE Cinema
      MODIFY Name = Theater;
- **DROP DATABASE** – allows to drop a database
    - Example: DROP DATABASE Theater;

SQL – Data Definition Language (DDL)

- **CREATE TABLE** – allows to create a table in a database
  - ***CREATE TABLE table_name***
    (
    column_name1 data_type,
    column_name2 data_type,
    …
    );
    - Example: CREATE TABLE Student
      (
      SId INT,
      Name VARCHAR(30),
      Surname VARCHAR(40),
      City VARCHAR(20)
      );

- **DROP TABLE** – allows to drop a table from a database
  - **DROP TABLE table_name**
    - Example: DROP TABLE Student

- **ALTER TABLE** – allows to modify the structure of a table from a database

  - Add a new column

ALTER TABLE table_name
ADD column_name data_type;

Example: ALTER TABLE Student
ADD Dob DATE;

  - Modify the type of a column

ALTER TABLE table_name
ALTER COLUMN column_name data_type;

Example: ALTER TABLE Student
ALTER COLUMN Dob DATETIME;

  - Delete a column

ALTER TABLE table_name
DROP COLUMN column_name;

Example: ALTER TABLE Student
DROP COLUMN Dob;

SQL – Data Definition Language (DDL)

- In SQL language, each column, (local) variable, expression, parameter, has a data type.
- A data type is an attribute that specify the type of values allowed for that object (e.g. TINYINT, INT, SMALLINT, BIGINT, DECIMAL, FLOAT, REAL, MONEY, NCHAR, CHAR, VARCHAR, NVARCHAR, DATE, DATETIME, TIME).
- The type / domain of each attribute / field is enforced by the DBMS whenever tuples / records are added or modified.

- Create the *Movie* table (relation):

CREATE TABLE Movie
(MId CHAR(10),
Title VARCHAR(70),
Release_Year TINYINT,
Run_Time INT,
Box_Office DECIMAL(12, 2))

- Create the *MovieActor* table (relation):

CREATE TABLE MovieActor
(MId CHAR(10),
AId CHAR(10),
Remuneration DECIMAL(12, 2))
- Stored data – which *Actors* perform in which *Movies*, with their corresponding *Remuneration*.

- Drop the *Movie* table (relation):

DROP TABLE Movie
- Both the schema information and the tuples in the table are removed.

- Alter the schema of *Movie* table by adding a new field:

ALTER TABLE Movie
ADD Synopsis VARCHAR(500)
- Every tuple in the current instance is extended with the *NULL* value for the new added field.
- The statement assumes that a table *Movie* exists.

- Alter the schema of *Movie* table by removing a field:

ALTER TABLE Movie
DROP COLUMN Run_Time

SQL – Data Definition Language (DDL)

- **CONSTRAINTS** – are used to ensure the integrity of the data that are introduced in a table.
- The data integrity can be ensured in declarative mode in the definition of the table or in procedural mode in stored procedures or triggers.
- The constraints can be declared in CREATE TABLE statement or ALTER TABLE statement.

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

NOT NULL constraint

- By default a table allows the NULL value.
- The *NOT NULL* constraint should be used to avoid NULL values on a column.
- If a column has *NOT NULL* constraint set on it, a value must be specified on that column.

Example: - on CREATE TABLE statement:

```
CREATE TABLE Student(
SId INT NOT NULL,
Name VARCHAR(30),
Surname VARCHAR(40),
City VARCHAR(20))
```

- on ALTER TABLE statement (plus change of column type):

```
ALTER TABLE Movie
ALTER COLUMN Release_Year SMALLINT NOT NULL
```

SQL – Data Definition Language (DDL)

UNIQUE constraint

- Can be defined on the columns in which the duplicates are not allowed (the NULL value is allowed).
- Can be used on one or multiple columns from a table; a table can contain multiple UNIQUE constraints.
- A UNIQUE constraint defined on multiple columns from a table, ensure the uniqueness of the values of those columns related to each tuple.
- By defining a UNIQUE constraint, it is also created automatically an UNIQUE corresponding index.

Example: - on CREATE TABLE statement on one column:

```
CREATE TABLE Student(
SId INT UNIQUE,
Name VARCHAR(30),
Surname VARCHAR(40),
City VARCHAR(20))
```

- on ALTER TABLE statement on one column:

```
ALTER TABLE Student
ADD UNIQUE(SId)
```

- on CREATE TABLE statement on multiple columns:

```
CREATE TABLE Student(
SId INT NOT NULL,
Name VARCHAR(30),
Surname VARCHAR(40),
City VARCHAR(20),
CONSTRAINT uk_Student UNIQUE(SId, Name))
```

- on ALTER TABLE statement on multiple columns:

```
ALTER TABLE Student
ADD CONSTRAINT uk_Student UNIQUE(SId, Name)
```

SQL – Data Definition Language (DDL)

DROP CONSTRAINT

- Allows to eliminate each constraint.

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name;
```

Example:

```
ALTER TABLE Student
DROP CONSTRAINT uk_Student;
```

PRIMARY KEY

- Each table must have one primary key.
- A primary key identify unique each tuple / record from the table.
- A primary key does not allow the duplicates or the NULL values on the column in which it is defined.
- Can be defined on a single column or on a combination of columns.
- If the primary key is defined on a combination of columns, then the combination of values should be unique.
- It can be defined only one PRIMARY KEY constraint in a table.

SQL – Data Definition Language (DDL)

PRIMARY KEY

Example: - on CREATE TABLE statement on one column:

```
CREATE TABLE Student(
SId INT PRIMARY KEY,
Name VARCHAR(30),
Surname VARCHAR(40),
City VARCHAR(20))
```

- on CREATE TABLE statement on multiple columns:

```
CREATE TABLE Student(
SId INT,
Name VARCHAR(30),
Surname VARCHAR(40),
City VARCHAR(20),
CONSTRAINT PK_Student PRIMARY KEY(SId, Name))
```

Example: - the *MovieActor* table creation statement with the primary key declaration

```
CREATE TABLE MovieActor
(MId CHAR(10),
AId CHAR(10),
Remuneration DECIMAL(12, 2),
PRIMARY KEY(MId, AId))
```

- Multiple candidate key can be declared using UNIQUE; one of them is chosen as the primary key.
- The primary key {MId, AId} corresponds to the constraint: For a given Actor and a given Movie, there is a single Remuneration; there are no 2 tuples in the relation with identical values in both MId and AId fields.

SQL – Data Definition Language (DDL)

PRIMARY KEY

Example: This is an example of how not to define keys; designating *{AId}* as the primary key corresponds to the constraint: an Actor can only perform in one movie; whereas choosing *{MId, Remuneration}* as a candidate key corresponds to the constraint: no two Actors can get the same Remuneration for a given Movie; such constraints prevent the storage of database instances that can arise in practice.

```
CREATE TABLE MovieActor
(MId CHAR(10),
AId CHAR(10),
Remuneration DECIMAL(12, 2),
PRIMARY KEY(AId),
UNIQUE(MId, Remuneration))
```

Example: - add a PRIMARY KEY constraint

```
ALTER TABLE Student
ADD CONSTRAINT PK_Student PRIMARY KEY(SId, Name)
```

- drop a PRIMARY KEY constraint

```
ALTER TABLE Student
DROP CONSTRAINT pk_Student;
```

- To define a PRIMARY KEY after a table creation statement, the column or columns that will be included in the PRIMARY KEY must have a NOT NULL constraint defined on them.

Example: - set NOT NULL (MId must be NOT NULL)

```
ALTER TABLE Movie
ALTER COLUMN MId CHAR(10) NOT NULL
```

- set PRIMARY KEY

```
ALTER TABLE Movie
ADD CONSTRAINT PK_Movie PRIMARY KEY(MId)
```

SQL – Data Definition Language (DDL)

FOREIGN KEY

- It is pointing to a PRIMARY KEY from another table; it has the same data type, the same values (not necessary all of them or only once) as the primary key and it is NOT NULL.

Table *Group*

GId	NoOfStudents
822	27
921	29

Table *Student*

SId	Name	Surname	City	**GId**
1	Olanescu	Dan	Alba-Iulia	***822***
2	Petre	Alina	Dej	***822***
3	Hora	Bogdan	Satu-Mare	***921***

- The column **Gid** from Table *Student* points to column **Gid** from Table *Group*.
- The column **Gid** from Table *Student* is **FOREIGN KEY**, and the column **Gid** from Table *Group* is PRIMARY KEY.
- The FOREIGN KEY constraint is used to prevent actions that could distroy the relationship between the 2 tables involved and also to prevent the introduction of other values, on the FOREIGN KEY position, from the ones that correspond to the PRIMARY KEY.
- No modifications are allowed on the table that contains the PRIMARY KEY, if these ones are involving the relationship with the FOREIGN KEY from the other table.

SQL – Data Definition Language (DDL)

FOREIGN KEY

Table *Group*

GId	NoOfStudents
822	27
921	29

```
CREATE TABLE Group(
GId INT PRIMARY KEY,
NoOfStudents INT)
```

Example: on CREATE TABLE statement

```
CREATE TABLE Student(
SId INT PRIMARY KEY,
Name VARCHAR(30),
Surname VARCHAR(40),
City VARCHAR(20),
GId INT,
CONSTRAINT fk_Student FOREIGN KEY(GId) REFERENCES Group(GId))
```

Table *Student*

SId	Name	Surname	City	**GId**
1	Olanescu	Dan	Alba-Iulia	***822***
2	Petre	Alina	Dej	***822***
3	Hora	Bogdan	Satu-Mare	***921***

```
CREATE TABLE Student(
SId INT PRIMARY KEY,
Name VARCHAR(30),
Surname VARCHAR(40),
City VARCHAR(20),
GId INT FOREIGN KEY REFERENCES Group(GId))
```

Example: on ALTER TABLE statement

```
ALTER TABLE Student
ADD FOREIGN KEY(GId)
REFERENCES Group(GId)
```

```
ALTER TABLE Student
ADD CONSTRAINT fk_Student FOREIGN KEY(GID)
REFERENCES Group(GId)
```

SQL – Data Definition Language (DDL)

FOREIGN KEY

Example:
CREATE TABLE MovieActor
(MId CHAR(10),
AId CHAR(10),
Remuneration DECIMAL(12, 2),
PRIMARY KEY(MId, AId),
FOREIGN KEY(MId) **REFERENCES** Movie(MId))

MId foreign key is equivalent with: the MovieActor table can store Actors only for the Movies that appear in the Movie table.

- FOREIGN KEY (foreign_key_attribute) REFERENCES primary_key_table (primary_key_attribute)
- CONSTRAINT fk_name_constraint FOREIGN KEY(foreign_key_attribute) REFERENCES primary_key_table (primary_key_attribute)

Example: adds a foreign key to the MovieActor table (table Actor is assumed to exist with {AId} as primary key):
create table Actor(
AId char(10) primary key,
Name varchar(50))

ALTER TABLE MovieActor
ADD CONSTRAINT fk_MovieActor FOREIGN KEY(AId) REFERENCES Actor(AId)

Example: removes a foreign key from the MovieActor table:
ALTER TABLE MovieActor
DROP fk_MovieActor

SQL – Data Definition Language (DDL)

FOREIGN KEY

In the case of update / delete operations, in order to enforce referential integrity constraints, the system can execute 4 actions:

- NO ACTION – the update / delete is not allowed if it violates the specified integrity constraint (default option);
- CASCADE – the update / delete is allowed on the parent table, but it also generates updates / deletes on the child table;
- SET NULL – the foreign key column values are replaced with *NULL* (only if they are nullable);
- SET DEFAULT–the foreign key column values are replaced with their default values (specified with DEFAULT); if a column is nullable and doesn't have a DEFAULT definition, *NULL* will be considered as the default value for the column.

Example: with actions

```
CREATE TABLE Student(
SId INT PRIMARY KEY,
Name VARCHAR(30),
Surname VARCHAR(40),
City VARCHAR(20),
GId INT FOREIGN KEY REFERENCES Group(GId)
ON DELETE CASCADE
ON UPDATE CASCADE)
```

```
CREATE TABLE MovieActor
(
MId CHAR(10),
AId CHAR(10),
Remuneration DECIMAL(12, 2),
PRIMARY KEY(MId, AId),
FOREIGN KEY(MId) REFERENCES Movie(MId)
ON DELETE CASCADE
ON UPDATE NO ACTION
)
```

SQL – Data Definition Language (DDL)

CHECK constraint

- It is used to define the values domain that can be used for a specified column.
- Can be defined on one or multiple columns.

Example: - on CREATE TABLE statement on one column

```
CREATE TABLE Student(
SId INT PRIMARY KEY CHECK(SId>0),
Name VARCHAR(30),
Surname VARCHAR(40),
City VARCHAR(20))
```

- on CREATE TABLE statement on multiple columns

```
CREATE TABLE Student(
SId INT PRIMARY KEY,
Name VARCHAR(30),
Surname VARCHAR(40),
City VARCHAR(20),
CONSTRAINT ck_SId CHECK(SId>0 AND
City IN ('Cluj Napoca', 'Brasov'))
```

Example:

```
CREATE TABLE Movie
(MId CHAR(10),
Title VARCHAR(70),
Release_Year TINYINT,
Run_Time INT,
Box_Office DECIMAL(12, 2),
CONSTRAINT PK_Movie PRIMARY KEY(MId),
CONSTRAINT Year_Range
CHECK(Release_Year >= 1905 AND Release_Year <= 2018))
```

Example: on ALTER TABLE statement

```
ALTER TABLE Student
ADD CHECK (SId>0)
```

Exemple: on ALTER TABLE statement with name for CHECK constraint

```
ALTER TABLE Student
ADD CONSTRAINT ck_Student
CHECK (SId>0 AND City IN ('Cluj Napoca', 'Brasov'))
```

SQL – Data Definition Language (DDL)

DEFAULT constraint

- It is used to insert a *default* value in a column.
- The *default* value will be inserted for all the new tuples / records, if there is no other value specified.
- Can be used to insert system values obtained from the execution of different functions.

Example: on CREATE TABLE statement

```
CREATE TABLE Student(
SId INT PRIMARY KEY,
Name VARCHAR(30),
Surname VARCHAR(40),
City VARCHAR(20),
EnrollmentDate DATE DEFAULT GETDATE())
```

- on ALTER TABLE statement

```
ALTER TABLE Student
ADD CONSTRAINT df_Student DEFAULT 'Alba-Iulia' FOR City
```

- DROP a DEFAULT constraint

```
ALTER TABLE Student
DROP CONSTRAINT df_Student
```

Example: on MovieActor table

```
ALTER TABLE MovieActor
ADD DEFAULT 0 FOR Remuneration
```