Databases and Information Systems

Translation from ER to the Relational Model

Prof. Ing. Lelio Campanile, PhD

Data Analytics Bachelor
Università degli Studi della Campania Luigi Vanvitelli

Prerequisite: ER Schema Restructuring

Learning goals

- Understand the **second step** of logical design: translating a restructured ER schema into a relational schema
- Apply a **systematic set of translation rules**, one for each type of relationship
- Recognize how **cardinalities** determine the correct translation strategy
- Produce a complete **relational schema** with correctly assigned primary keys

From ER to the Relational Model

The two-step process — where we are

Step 1 — Restructuring (done)

- Analyse and remove redundancies
- Remove generalizations
- Partition or merge concepts
- Select primary identifiers

Step 2 — Translation (this lecture)

- Translate each entity into a **relation**
- Translate each relationship using the **appropriate rule**
- Assign **primary keys** to each relation
- The result is a complete **relational schema**

The input is a **restructured** ER schema — generalizations have already been removed.

The relational model — a quick reminder

Key concepts

- A **relation** (table) has a name and a set of **attributes** (columns)
- A **primary key** (PK) uniquely identifies each tuple — underlined by convention
- Attributes that **reference the PK of another relation** act as links between tables — we will formalise these as **foreign keys** in the next module on the relational model

Notation used in this lecture

`RELATION(pk_attr, other_attr, ref_attr)`

Underlined = primary key Plain = all other attributes (including references)

Translation Rules

Overview of the translation rules

Rule	Relationship type	Strategy
1	Many-to-many (N:M)	New relation with two reference attributes
2	One-to-many, mandatory (1,1) side	Reference attr. into the (1,1) entity
3	One-to-many, optional (0,1) side	Separate relation or nullable reference
4	One-to-one, both mandatory	Merge or reference attr. (either side)
5	One-to-one, one optional	Reference attr. into the mandatory side
6	One-to-one, both optional	Separate relation
7	Ternary (N:M:P)	New relation with three reference attributes
8	External identifier	Reference attr. as part of the PK

We will examine each rule with a diagram and a concrete example.

Rule 1 — Many-to-Many Relationship

Rule 1 — Many-to-many: the idea

When does this apply?

Both sides of the relationship have maximum cardinality **N**: neither entity can absorb the relationship as a simple reference attribute.

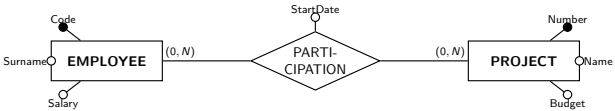
Translation strategy

Create a **new relation** for the relationship. Its primary key is the **combination** of the two entity PKs (plus any relationship attributes).

Schema

`E1(k1, a1)` `E2(k2, a2)` `R(k1, k2, ar)`

Rule 1 — Many-to-many: example



Relational schema

- EMPLOYEE(Code, Surname, Salary)
- PROJECT(Number, Name, Budget)
- PARTICIPATION(Code, Number, StartDate)

- **StartDate** is an attribute of the *relationship* — it moves into the new PARTICIPATION table
- Both Code and Number are references to the entity PKs and together form the composite PK of

Rule 2 — One-to-Many (Mandatory)

Rule 2 — One-to-many, mandatory participation: the idea

When does this apply?

One side has cardinality **(1,1)**: every instance of that entity *must* participate in the relationship exactly once.

Translation strategy

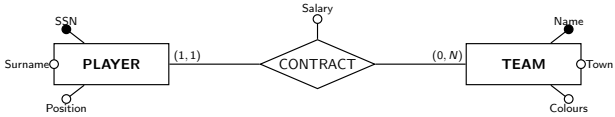
No new relation needed. Add the PK of the "many" side as a **reference attribute** directly into the "(1,1)" entity relation.
The relationship attribute (if any) also moves there.

Schema

`E1(k1, a1, k2, ar)` `E2(k2, a2)`

E1 has cardinality (1,1) — it absorbs the reference to E2

Rule 2 — One-to-many: example



Relational schema

- PLAYER(<u>SSN</u>, Surname, Position, Team, Salary)
- TEAM(<u>Name</u>, Town, Colours)

- Every player *must* belong to a team $(1, 1)$, so Team (reference to TEAM's PK) goes into PLAYER
- **Salary** (relationship attr.) also moves into PLAYER — it describes the specific player-team contract
- No separate table needed

Rule 3 — One-to-Many (Optional)

Rule 3 — One-to-many, optional participation: the idea

When does this apply?

The entity that would absorb the reference has cardinality **(0,1)**: some instances may *not* participate in the relationship.

Option A — nullable reference

Add the reference attribute into the (0,1) entity with a **nullable** value.

`E1(k1, a1, k2*, ar*)`
`E2(k2, a2)`

Simple, but introduces NULL values.

Option B — separate relation

Create a new table for the relationship (same as Rule 1).

`E1(k1, a1)`
`E2(k2, a2)`
`R(k1, k2, ar)`

No NULLs, but one extra join.

Choose **Option A** when optional participation is **rare** (few NULLs acceptable); **Option B** when it is **frequent** (many NULLs — keep the table clean).

Rules 4–6 — One-to-One Relationships

Rules 4–6 — One-to-one: overview

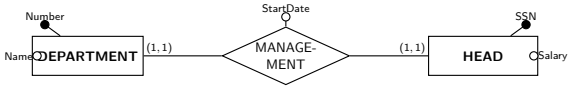
The three sub-cases

The correct translation depends on **which side(s) are optional**.

Cardinalities	Rule	Strategy
Both **(1,1)**	Rule 4	Reference attr. into either side (or merge)
One **(1,1)**, one **(0,1)**	Rule 5	Reference attr. into the **(1,1)** side
Both **(0,1)**	Rule 6	Separate relation (or nullable reference)

- The key principle: **avoid NULL values** whenever possible
- A reference attribute into the **(1,1)** side is always non-null — safe and efficient

Rule 4 — One-to-one, both mandatory



Option A — reference attr. into either side

```
DEPARTMENT(Number, Name,
  Head, StartDate)
HEAD(SSN, Salary)
```

Option B — merge the two entities

```
DEPT_HEAD(Number, Name,
  SSN, Salary, StartDate)
```

Option A is generally preferred — merging is only beneficial when the two entities are always accessed together.

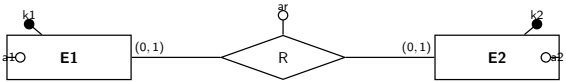
Rule 5 — One-to-one, one optional

Relational schema

```
DEPARTMENT(Number, Name,
  Head*, StartDate*)
EMPLOYEE(SSN, Salary)
```

- The reference to HEAD goes into **DEPARTMENT** (the mandatory side)
- Not every employee manages a department, so placing the reference in EMPLOYEE would produce NULL for most employees
- The $(1, 1)$ cardinality on DEPARTMENT guarantees the reference is always

Rule 6 — One-to-one, both optional



Preferred: separate relation

```
E1(k1, a1)
E2(k2, a2)
R(k1, k2, ar)
```

Alternative: nullable reference

```
E1(k1, a1, k2*, ar*)
E2(k2, a2)
```

When both sides are optional, NULLs are unavoidable with any reference strategy — a separate relation keeps the main tables clean.

Rule 7 — Ternary Relationship

Rule 7 — Ternary relationship: the idea

When does this apply?

A relationship connects **three entities simultaneously**. No binary decomposition can capture the same semantics.

Translation strategy

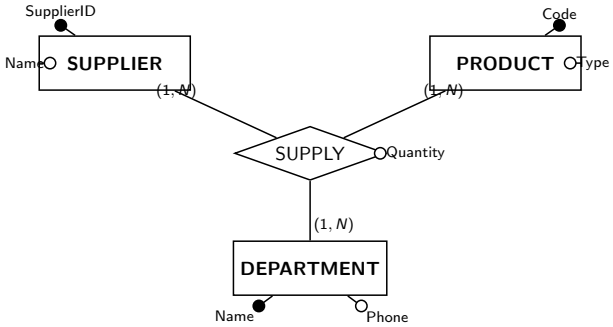
Always create a **new relation**. Its primary key is the **combination of all three entity PKs** (plus any relationship attributes).

Schema

E1(<u>k1</u>, a1) E2(<u>k2</u>, a2) E3(<u>k3</u>, a3)

R(<u>k1</u>, <u>k2</u>, <u>k3</u>, ar)

Rule 7 — Ternary relationship: example



Relational schema

- SUPPLIER(SupplierID, Name)
- PRODUCT(Code, Type)
- DEPARTMENT(Name, Phone)
- SUPPLY(SupplierID, Code, Dept, Quantity)

All three references to entity PKs form the composite PK of SUPPLY.

Rule 8 — External Identifier

Rule 8 — External identifier: the idea

When does this apply?

An entity has **no internal identifier** — it can only be identified *in combination* with another entity (the owner). In BCN notation this is an **identificatore esterno**.

Translation strategy

The PK of the weak entity is the **combination of its own partial key and the PK of the owner entity** (carried as a reference attribute). The (1,1) cardinality on the weak side ensures the reference is always present.

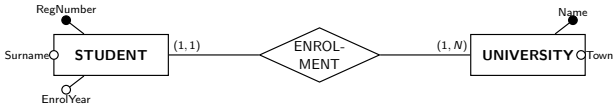
Schema

`OWNER(`<u>`k2`</u>`, a2)`

`WEAK(`<u>`k1`</u>`, `<u>`k2`</u>`, a1, ar)`

`k2` references OWNER and is also part of the PK of WEAK

Rule 8 — External identifier: example



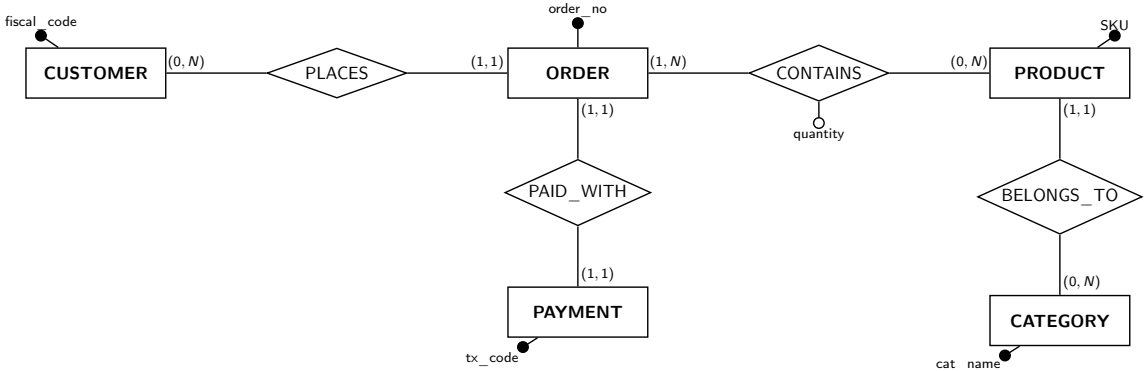
Relational schema

- UNIVERSITY(Name, Town)
- STUDENT(RegNumber, University, Surname, EnrolYear)

- University references UNIVERSITY's PK and is **also part of the PK** of STUDENT
- Registration numbers are only unique *within* a university
- The $(1, 1)$ cardinality guarantees a student always belongs to exactly

Putting it Together

A complete ER schema



Step-by-step translation

Entities → relations (always)

- CUSTOMER(fiscal_code, ...)
- ORDER(order_no, ...)
- PRODUCT(SKU, ...)
- PAYMENT(tx_code, ...)
- CATEGORY(cat_name, ...)

Relationships → apply rules

- **PLACES** $(0, N)$–$(1, 1)$:
  Rule 2 → reference into ORDER
- **CONTAINS** $(1, N)$–$(0, N)$:
  Rule 1 → new relation
- **PAID_WITH** $(1, 1)$–$(1, 1)$:
  Rule 4 → reference into ORDER
- **BELONGS_TO** $(1, 1)$–$(0, N)$:
  Rule 2 → reference into PRODUCT

Final relational schema

Complete schema — PKs underlined, reference attributes in plain text

- CUSTOMER(fiscal_code, name, email)
- CATEGORY(cat_name)
- PAYMENT(tx_code, method, amount)
- PRODUCT(SKU, description, price, cat_name) ← cat_name references CATEGORY [Rule 2]
- ORDER(order_no, date, fiscal_code, tx_code) ← fiscal_code ref. CUSTOMER; tx_code ref. PAYMENT [Rules 2,4]
- CONTAINS(order_no, SKU, quantity) ← N:M with relationship attribute [Rule 1]

Summary

Summary — translation rules at a glance

Rule	Condition	Result
1	N:M	New relation, composite PK
2	1:N, (1,1) side exists	Reference attr. into the (1,1) entity
3	1:N, (0,1) side only	Nullable reference *or* new relation
4	1:1, both (1,1)	Reference attr. into either side (or merge)
5	1:1, one (1,1)	Reference attr. into the (1,1) side
6	1:1, both (0,1)	New relation (preferred)
7	Ternary	New relation, three-way PK
8	External identifier	Reference attr. as part of PK

Key principle: always check the cardinalities first. They tell you exactly which rule to apply.

How to reason during translation

1. Translate **every entity** into a relation with its attributes and PK
2. For **each relationship**, read the cardinality pair
3. Select the **translation rule** from the table
4. Add the reference attribute (or new relation) accordingly
5. Move any **relationship attributes** into the appropriate relation
6. Check: no unnecessary NULLs, all reference attributes point to an existing PK

Questions for the class

- Why can't we always use Rule 1 (new relation) for *every* relationship?
- In Rule 3, when would you prefer the nullable reference over the separate relation?
- In Rule 5, why do we put the reference attribute into the **mandatory** side rather than the optional side?
- Can the same attribute be *both* a primary key and a reference to another relation? Give an example.
- In the e-commerce schema, why does `quantity` go into `CONTAINS` and not into `PRODUCT`?

In-class exercise

Given the following ER schema, produce the complete relational schema.

Domain: A hospital manages **patients**, **doctors**, and **wards**.

- Each **patient** (SSN, name, birthdate) is **admitted to** exactly one ward
- Each **ward** (ward_code, name, floor) is **supervised by** exactly one doctor
- A doctor (employee_id, name, specialisation) may supervise **at most one** ward
- Doctors **treat** patients; for each treatment store the **start_date**

For each relationship: identify the type, select the rule, write the schema.