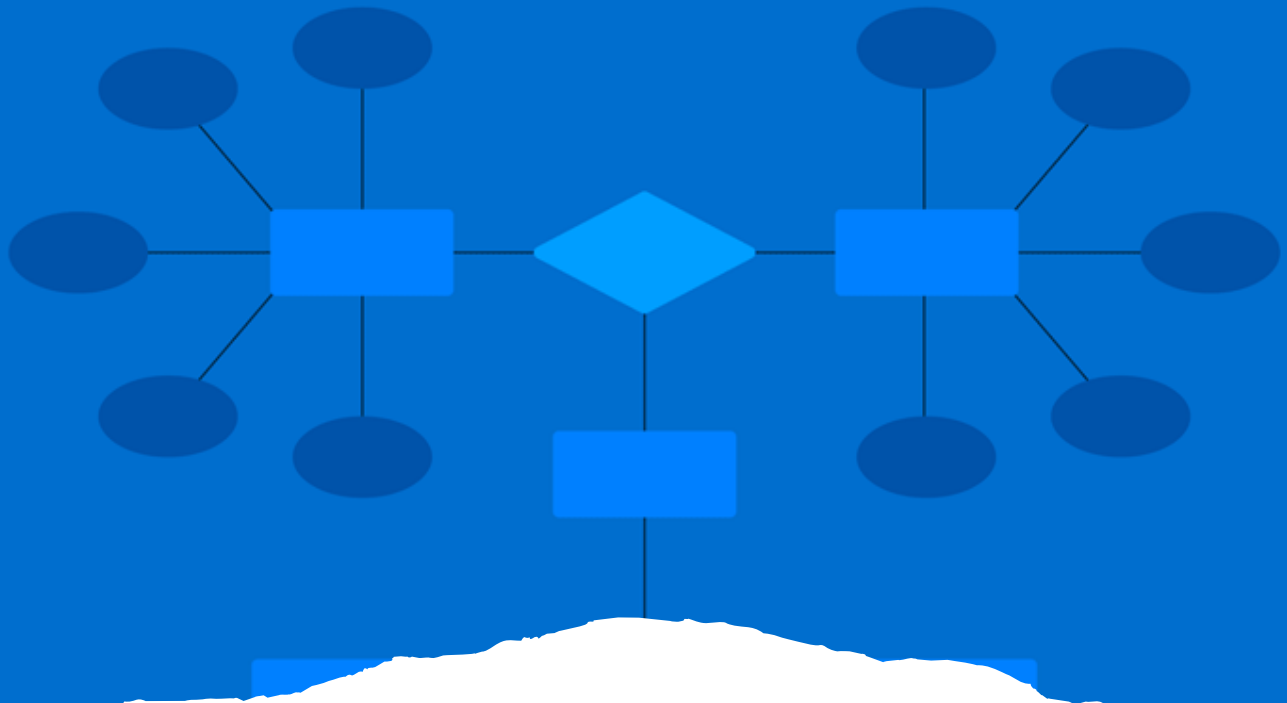
ENTITY RELATIONSHIP MODEL



COM106: Introduction to Databases

Mapping to the Relational Model

Mapping to the Relational Model

We have been studying the process of designing a database :

1. Creating an E-R Diagram based on the results of a data investigation (E-R Modelling) ✓
2. Using the mapping (translation rules) to create a relational schema (a set of linked tables)
3. Checking that each table is correctly designed (i.e. in 3rd Normal Form) which may involve decomposition (splitting into more tables)

Steps 1 & 2 are part of ER modelling; 3 is Normalisation

Mapping to the Relational Model involves :

Creating a relational table for each entity type

Linking the tables to reflect the relationships identified in the ER Diagram

This is achieved using the mapping rules which specify for a particular cardinality ratio (and sometimes a particular participation constraint) how the tables should be linked (to allow joins in SQL)

Mapping to the Relational Model

There are three main steps to the mapping process

1. **Create a Relation** (or table) for each Entity Type
   Entity identifier becomes the primary key

2. **Link the Tables** using the Relational Mapping Rules
   either - *Copy primary keys to other tables as foreign keys*
   or - *Create relationship relations (linking tables)*

3. **Complete Relations**
   Remove derived attributes (calculated from existing attributes)
   Add 'active attributes' to relationship relations
   Remove unnecessary tables

Links (or common attributes) are built between entity types to express the relationships identified in the E-R model

We will consider the two basic approaches in turn:

1. *Key Copying*
2. *Link Table Creation*

Mapping to the Relational Model

Key Copying

The Primary Key (**PK**) of one table is copied (or posted) to another table as a Foreign Key (**FK**) to express a relationship

For example, consider **DEPARTMENT** **Employs** **EMPLOYEE** **(1:N)**

With tables **DEPARTMENT** (dno, dname) and **EMPLOYEE** (eno, ename, salary)

DEPARTMENT

dno	dname
1	Sales
2	Admin

EMPLOYEE

eno	ename	salary	dno
101	Dopey	21000	1
102	Grumpy	25000	2
103	Happy	19000	2
104	Sneezy	22000	1
105	Sleepy	25000	2

We could post the **PK** from **DEPARTMENT** (**dno**) to **EMPLOYEE**

Adding **dno** as a **FK** in **EMPLOYEE** expresses the **1:N** relationship from **DEPARTMENT** to **EMPLOYEE**

This allows **DEPARTMENT** and **EMPLOYEE** to be joined to connect **EMPLOYEE** records with corresponding **DEPARTMENT** records

The tables are now :

* *non-standard notation; used here to denote a FK*

DEPARTMENT (**dno**, dname) and
EMPLOYEE (eno, ename, salary, **dno***)

Mapping to the Relational Model

Link Table Creation

For example, consider SUPPLIER Ships PART (M:N)

With Tables: SUPPLIER (sno, supp_name) PART (pno, part_name)

To add the Ships relationship (M:N) - create a link table called SHIPS.

The PK of the new SHIPS table is a composite of those from SUPPLIER and PART

i.e., SHIPS(sno*, pno*, quantity)

SUPPLIER

sno	supp_name
135	Camco
159	J&J Parts

PART

pno	part_name
P13	Flange Plate
P28	Sprocket Cover
P42	Speed Limiter

SHIPS

sno	pno	quantity
135	P13	130
135	P42	57
159	P13	93
159	P28	75
159	P42	10

The link table (SHIPS) may also contain '*active attributes*' (normally amounts or dates) such as, e.g., 'quantity' which belong only in a linking table.

Also, to express the M:N Ships relationship, the SHIPS link table contains duplicate sno values AND duplicate pno values.

Effectively SUPPLIER to SHIPS is 1:N AND PART to SHIPS is 1:N

Mapping to the Relational Model

Comparison of Mapping Approaches

Key Copying

does not involve extra tables (no extra join overhead)

not possible for M:N relationships

Link Tables

uses extra linking table (extra join overhead)

necessary for M:N relationship

reduced null values in foreign keys of some 1:1 or 1:N relationships (where certain participation constraints are partial)

Where possible, null values (blank value) should be reduced by design

1:1 Mapping Rule

Copy **PK** of either relation (*but not both!*) to the other as **FK**

For example, consider the relationship **EMPLOYEE** **Assigned** **CAR**

EMPLOYEE (enum, ename) **CAR** (regnum, make, model)

Copy primary key in either direction

EMPLOYEE (enum, ename, regnum*) **CAR** (regnum, make, model)

or **EMPLOYEE** (enum, ename) **CAR** (regnum, make, model, enum*)

Mapping to the Relational Model

1:1 Mapping rule - Example

A **1:1** relationship is implemented as a **special form of a 1:N** relationship where the **FK MUST contain UNIQUE values**

*In a **1:N** relationship a FK **SHOULD NOT have unique values***

For example, in **EMPLOYEE** **Assigned** **CAR** *(with enum as FK in **CAR**)*

EMPLOYEE

enum	ename
101	Tom
103	Dick
104	Harry

CAR

regnum	make	model	enum
GJZ 1498	Ford	Focus	104
EFZ 5391	Renault	Scenic	101
SEZ 7980	Ford	Fiesta	103

Note that **enum** in **CAR** (**FK**) must have **unique** values

Could also have used **regnum** as FK in EMPLOYEE – again, must have **unique** values

If (***and only if***) regnum is the **only** data to be held about **CAR**, then the relational schema can be simplified by . . . .

deciding that **CAR** is not an entity (i.e., should not be modelled) and adding **regnum** (the unique identifier) as an attribute in **EMPLOYEE**

Mapping to the Relational Model

1:N Mapping rule

Copy primary key from the **'1'** relation to the **'N'** relation as **foreign key**

For example, consider the relationship **WARD Contains PATIENT (1:N)**

WARD (wardnum , wname) **PATIENT** (patnum , pname)

Copy PK of the '1' table (**WARD**) to the 'N' table (**PATIENT**) as FK

i.e., **WARD** (wardnum , wname) **PATIENT** (patnum , pname, **wardnum***)

In **PATIENT**, the FK (**wardnum**) should contain duplicate values – i.e. there should be more than one patient in each ward

M:N Mapping Rule

Create a **Link Table** using primary keys from **both** tables as the **composite primary key**

- Normally use the relationship name as the Link Table Name
- PK of Link Table is **composite**
- Other '**active attributes**' may be added if appropriate

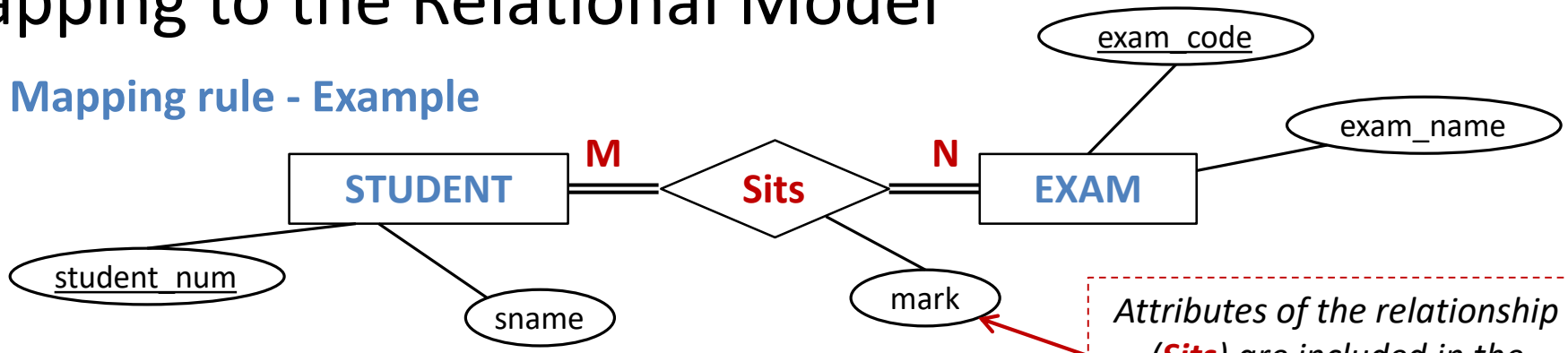
For example, consider the relationship **STUDENT Sits EXAM (M:N)**

STUDENT (student_num, sname) **EXAM** (exam_code, exam_name)

Link Table **SITS** is created - **SITS** (student_num*, exam_code*, mark)

Mapping to the Relational Model

M:N Mapping rule - Example



*Attributes of the relationship (**Sits**) are included in the Link Table (**SITS**)*

*Alternatively, could use a unique identifier (**id**) as the **PK** in the Link Table (**SITS**) instead of using a composite key*

STUDENT

student_num	sname
12345	J Goodall
23456	P McCann

EXAM

exam_code	exam_name
COM140	Computer Technology
COM106	Database
COM101	Programming

SITS

id	Student_num	exam_code	mark
1	12345	COM140	56
2	12345	COM106	62
3	12345	COM101	48
4	23456	COM140	69
5	23456	COM106	50

Mapping to the Relational Model

When the relational mapping rules have been applied, and the tables are linked, all that remains in constructing the relational model is to **complete the relations**:

1. Remove Derived Attributes

Derived attributes are those which can be **calculated** from other existing attribute(s).

They do not need to be stored (can be calculated in reports, queries etc.)

Examples of Derived Attributes:

*Total Cost = Unit Cost * Number Ordered*

Pension Contribution = %age of Salary

In practice derived attributes are normally omitted earlier in the design process

2. Add 'Active Attributes' to Relationship Relations

Active attributes tend to be **amounts or dates/time**

We have already seen an example when mapping the **Sits** relationship between **STUDENT** and **EXAM**

The 'mark' attribute of the **Sits** relationship is added to the **SITS** linking table

See also the projemp database used in the SQL practicals;

The **WORKS** table has the nonkey attributes '**role**' & '**duration**' –

WORKS (eno*, pno*, **role**, **duration**)

Mapping to the Relational Model

3. Remove Unnecessary Tables

Sometimes tables contain attributes found in other tables and do not link to more than one other table. It may be possible to **eliminate** such tables thereby reducing the join overhead.

For example, consider the relationship **EMPLOYEE** **Awarded** **QUALIFICATION** **(M:N)**

Applying the mapping rules gives:

EMPLOYEE (eno , ename) **AWARDED** (eno*, **qual***) **QUALIFICATION** (**qual**)

Since attribute **qual** is also found in **AWARDED** –
QUALIFICATION is unnecessary and can be **eliminated**

Mapping to the Relational Model

Summary of the steps in Mapping to the Relational Model

Mapping rules are applied to an E-R model to create a **relational schema** (a set of linked tables)

1. Create a Relation (or table) for each Entity Type

2. Link the Tables using the Relational Mapping Rules

1:1 relationship - Copy PK into other table as FK (in any direction but NOT both)

1:N relationship - Copy PK from the '1' table into the 'N' table as FK

M:N relationship - Create link table (using both PKs); PK on new Link Table is composite

3. Complete the relations

Remove derived attributes

Add 'active attributes' to relationship relations

Remove unnecessary tables

The result is a **Relational Schema** (a set of linked tables), such as , for example, the projemp relational schema used in the labs:

DEPT (dno, dname, location)
EMP (eno, ename, salary, age, supno, dno*)
WORKS (eno*, pno*, role)
PROJ (pno, pname, ptype, budget)