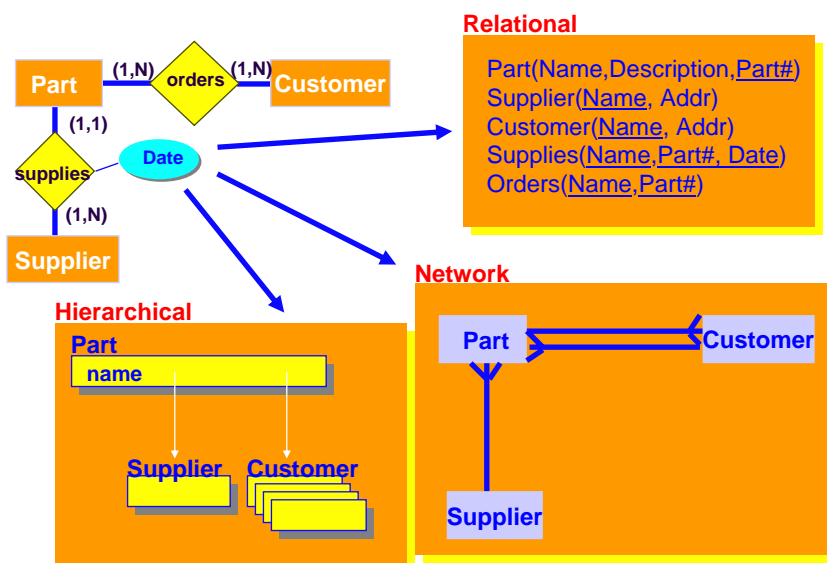


# Week 11: Database Design

Database Design  
From an ER Schema to a Relational One  
Restructuring an ER schema  
Performance Analysis  
Analysis of Redundancies, Removing  
Generalizations  
Translation into a Relational Schema



## Designing a Database Schema

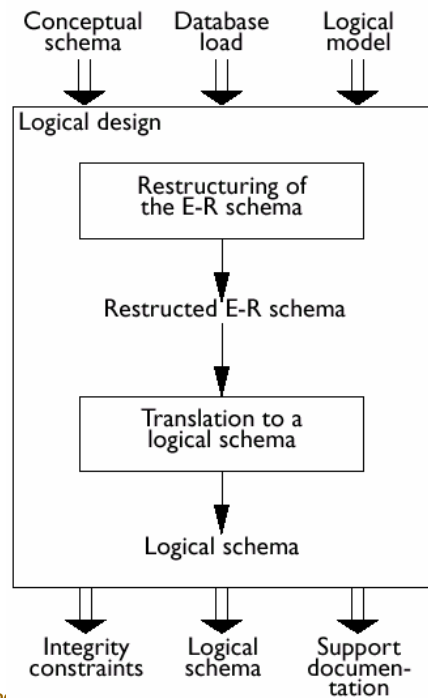


## (Relational) Database Design

- Given a conceptual schema (ER, but could also be a UML), generate a logical (relational) schema.
- This is **not** just a simple translation from one model to another for two main reasons:
  1. not all the constructs of the Entity-Relationship model can be translated naturally into the relational model;
  2. the schema must be restructured in such a way as to make the execution of the projected operations as efficient as possible.
- The topic is covered in section 3.5 of the textbook. This lecture unit uses material from other textbooks as well.

CSC343 – Introduction to Databases

Database Design — 3



CSC343 – Intro

Database Design — 4

## Database Design Process

## Logical Design Steps

It is helpful to divide the design into two steps:

1. **Restructuring of the Entity-Relationship schema**, based on criteria for the optimization of the schema and the simplification of the following step;
2. **Translation into the logical model**, based on the features of the logical model (in our case, the relational model).

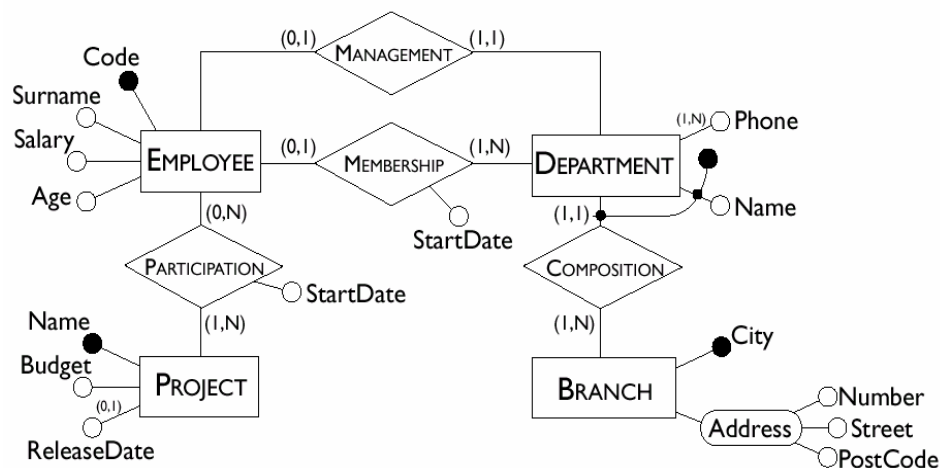
## Performance Analysis

- An ER schema is restructured to optimize:
  - ✓ **Cost of an operation** (evaluated in terms of the number of occurrences of entities and relationships that are visited during the execution of an operation);
  - ✓ **Storage requirements** (evaluated in terms of number of bytes necessary to store the data described by the schema).
- In order to study these parameters, we need to know:
  - ✓ Projected volume of data;
  - ✓ Projected operation characteristics.

## Cost Model

- The cost of an operation is measured in terms of the number of disk accesses required. A **disk access** is, generally, orders of magnitude more expensive than in-memory accesses, or CPU operations.
- For a coarse estimate of cost, we assume that
  - ✓ a Read operation (for one entity or relationship) requires 1 disk access;
  - ✓ A Write operation (for one entity or relationship) requires 2 disk accesses (read from disk, change, write back to disk).

## Employee-Department Example



## Typical Operations

- Operation 1: Assign an employee to a project.
- Operation 2: Find an employee record, including her department, and the projects she works for.
- Operation 3: Find records of employees for a department.
- Operation 4: For each branch, retrieve its departments, and for each department, retrieve the last names of their managers, and the list of their employees.
- **Note:** For UML class diagrams, these would be operations associated with persistent database classes.

## Tables of Volumes and Operations

The volume of data and the general characteristics of the operations can be summed up using two special tables.

**Table of volumes**

| Concept       | Type | Volume |
|---------------|------|--------|
| Branch        | E    | 10     |
| Department    | E    | 80     |
| Employee      | E    | 2000   |
| Project       | E    | 500    |
| Composition   | R    | 80     |
| Membership    | R    | 1900   |
| Management    | R    | 80     |
| Participation | R    | 6000   |

**Table of operations**

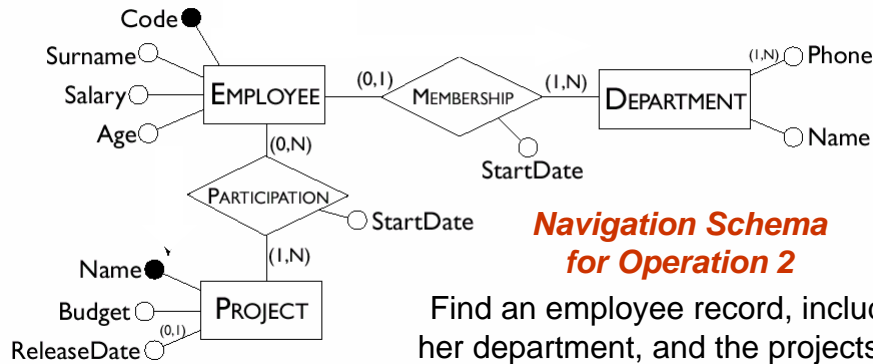
| Operation   | Type | Frequency   |
|-------------|------|-------------|
| Operation 1 | I    | 50 per day  |
| Operation 2 | I    | 100 per day |
| Operation 3 | I    | 10 per day  |
| Operation 4 | B    | 2 per day   |

I - Interactive

B - Batch

## Navigation Schema

A *navigation schema* starts from the inputs to an operation and moves (via arrows) towards its outputs.



## Table of Accesses

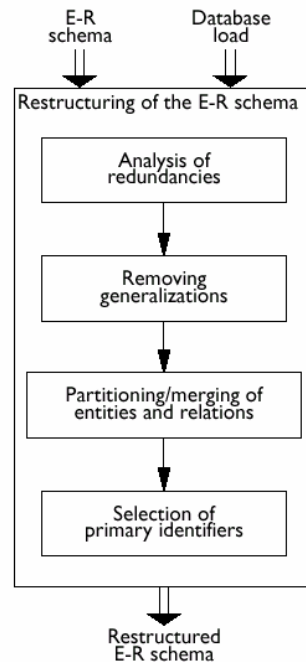
This table evaluates the cost of an operation, using the table of volumes and the navigation schema.

**Operation 2**

| Concept       | Type         | Accesses | Type |
|---------------|--------------|----------|------|
| Employee      | Entity       | 1        | R    |
| Membership    | Relationship | 1        | R    |
| Department    | Entity       | 1        | R    |
| Participation | Relationship | 3        | R    |
| Project       | Entity       | 3        | R    |

Average # of participations and projects per employee

Type:  
R – Read,  
W - Write,  
RW - Read&Write.

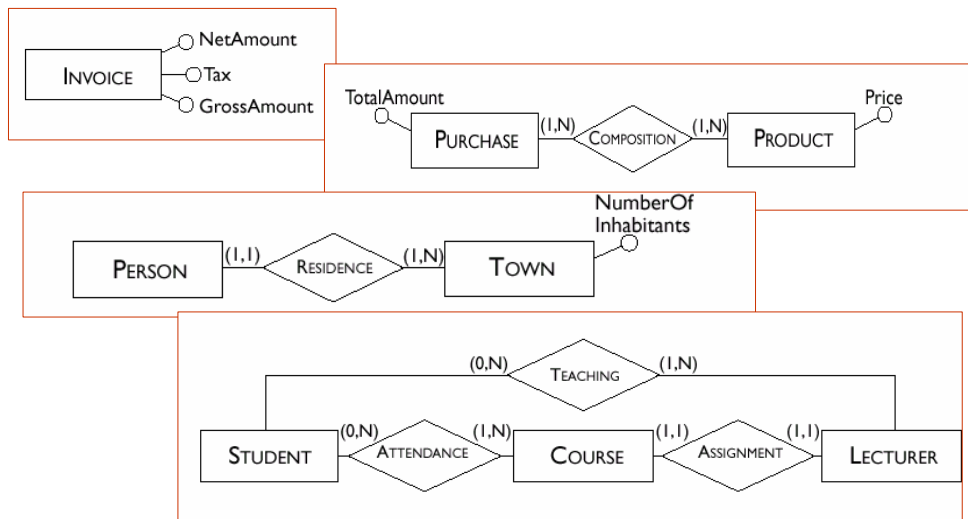


## Analysis Steps

## Analysis of Redundancies

- A redundancy in a conceptual schema corresponds to a piece of information that can be derived (that is, obtained through a series of retrieval operations) from other data in the database.
- An Entity-Relationship schema may contain various forms of redundancy.

## Examples of Redundancies



CSC343 – Introduction to Databases

Database Design — 15

## Deciding About Redundancies

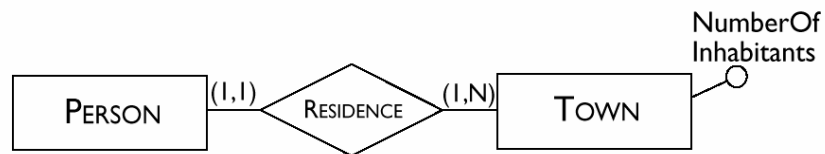
- The presence of a redundancy in a database may be
  - ✓ **an advantage:** a reduction in the number of accesses necessary to obtain the derived information;
  - ✓ **a disadvantage:** because of larger storage requirements, (but, usually at negligible cost) **and the necessity to carry out additional operations in order to keep the derived data consistent.**
- The decision to maintain or eliminate a redundancy is made by comparing the cost of operations that involve the redundant information and the storage needed, in the case of presence or absence of redundancy.

CSC343 – Introduction to Databases

Database Design — 16



## Cost Comparison: An Example



In this schema the attribute *NumberOfInhabitants* is redundant.

## Load and Frequency of Operations

Table of volumes

| Concept   | Type | Volume  |
|-----------|------|---------|
| Town      | E    | 200     |
| Person    | E    | 1000000 |
| Residence | R    | 1000000 |

Table of operations

| Operation   | Type | Frequency   |
|-------------|------|-------------|
| Operation 1 | I    | 500 per day |
| Operation 2 | I    | 2 per day   |

- **Operation 1:** add a new person with the person's town of residence.
- **Operation 2:** print all the data of a town (including the number of inhabitants).

## Table of Accesses, with Redundancy

### Operation 1

| Concept   | Type         | Accesses | Type |
|-----------|--------------|----------|------|
| Person    | Entity       | 1        | W    |
| Residence | Relationship | 1        | W    |
| Town      | Entity       | 1        | W    |

### Operation 2

| Concept | Type   | Accesses | Type |
|---------|--------|----------|------|
| Town    | Entity | 1        | R    |

## Table of Accesses, without Redundancy

### Operation 1

| Concept   | Type         | Accesses | Type |
|-----------|--------------|----------|------|
| Person    | Entity       | 1        | W    |
| Residence | Relationship | 1        | W    |

### Operation 2

| Concept   | Type         | Accesses | Type |
|-----------|--------------|----------|------|
| Town      | Entity       | 1        | R    |
| Residence | Relationship | 5000     | R    |

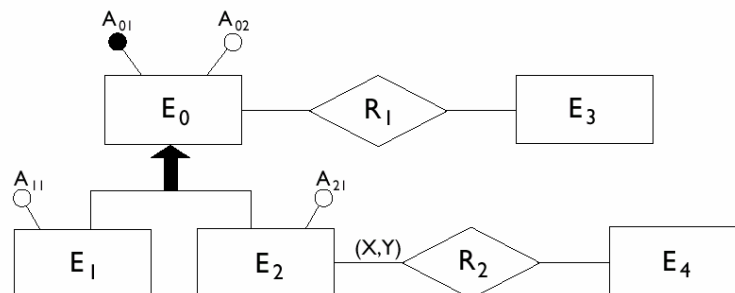
## Comparing the Cost of Operations

- Presence of redundancy:
  - ✓ Operation 1: 1,500 write accesses per day;
  - ✓ The cost of operation 2 is almost negligible;
  - ✓ Counting twice the write accesses, we have a total of 3,000 accesses a day.
- Absence of redundancy.
  - ✓ Operation 1: 1,000 write accesses per day;
  - ✓ Operation 2 however requires a total of 10,000 read accesses per day;
  - ✓ Counting twice the write accesses, we have a total of 12,000 accesses per day.

**Redundant data may improve performance!**

## Removing Generalizations

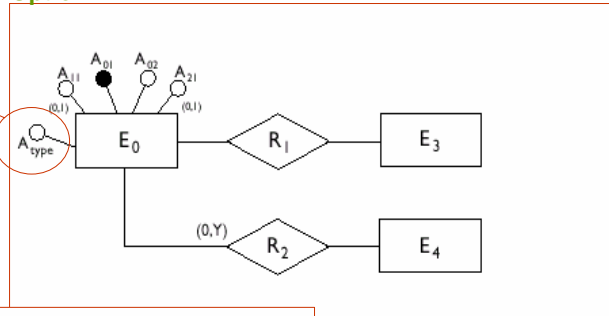
- The relational model does not allow direct representation of generalizations that may be present in an E-R diagram.
- For example, here is an ER schema with generalizations:



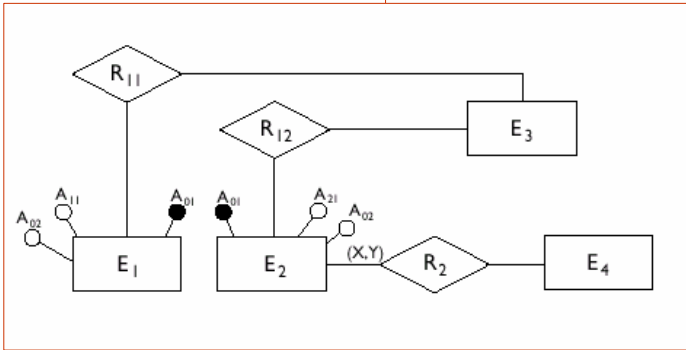
# Possible Restructurings

Note!

## Option 1

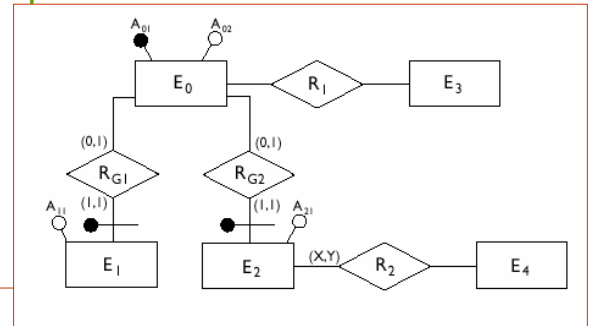


## Option 2

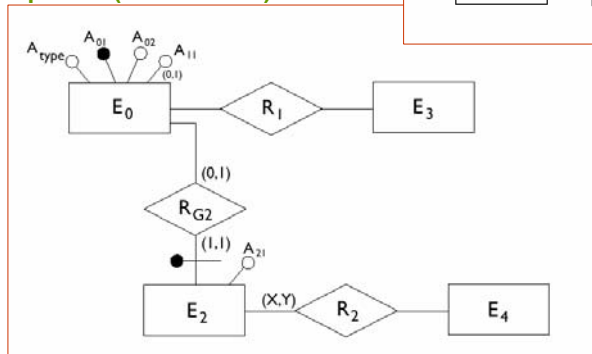


# ...Two More...

## Option 3



## Option 4 (combination)



## General Rules For Removing Generalization

- Option 1 is convenient when the operations involve the occurrences and the attributes of  $E_0$ ,  $E_1$  and  $E_2$  more or less in the same way.
- Option 2 is possible only if the generalization satisfies the coverage constraint (i.e., every instance of  $E_0$  is either an instance of  $E_1$  or  $E_2$ ) and is useful when there are operations that apply only to occurrences of  $E_1$  or  $E_2$ .
- Option 3 is useful when the generalization is not coverage-compliant and the operations refer to either occurrences and attributes of  $E_1$  ( $E_2$ ) or of  $E_0$ , and therefore make distinctions between child and parent entities.
- Available options can be combined (see option 4)

CSC343 – Introduction to Databases

Database Design — 25

## Partitioning and Merging of Entities and Relationships

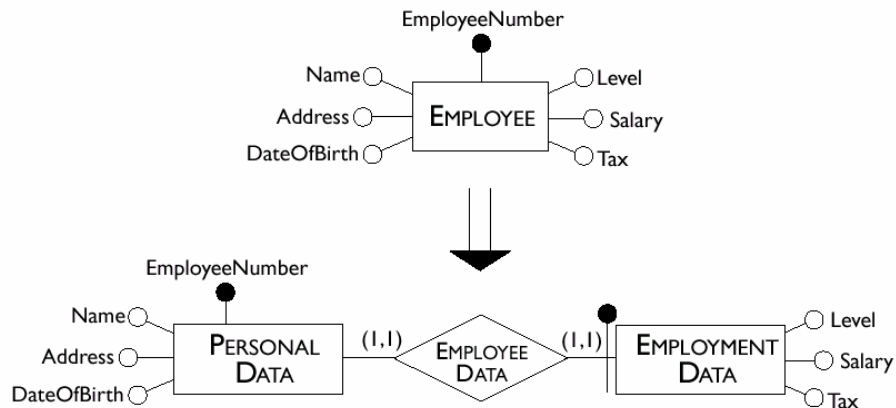
- Entities and relationships of an E-R schema can be partitioned or merged to improve the efficiency of operations, using the following principle:

***Accesses are reduced by separating attributes of the same concept that are accessed by different operations and by merging attributes of different concepts that are accessed by the same operations.***

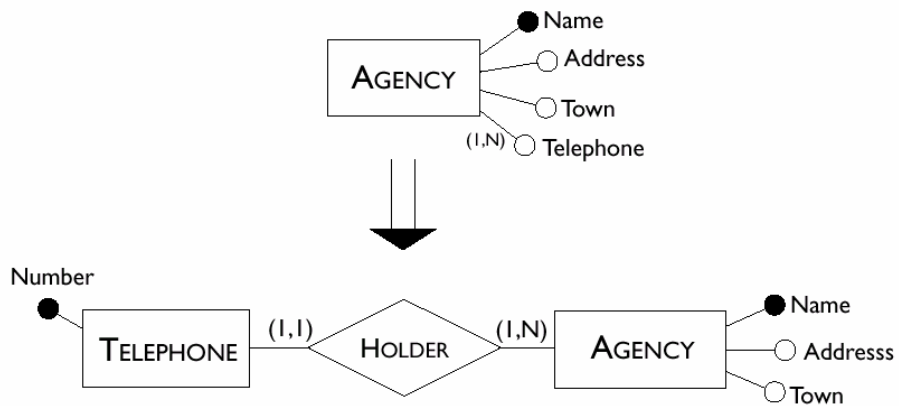
CSC343 – Introduction to Databases

Database Design — 26

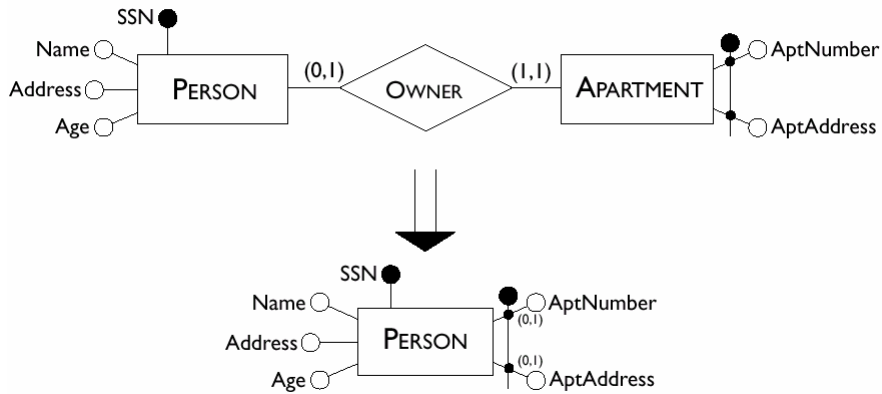
## Example of Partitioning



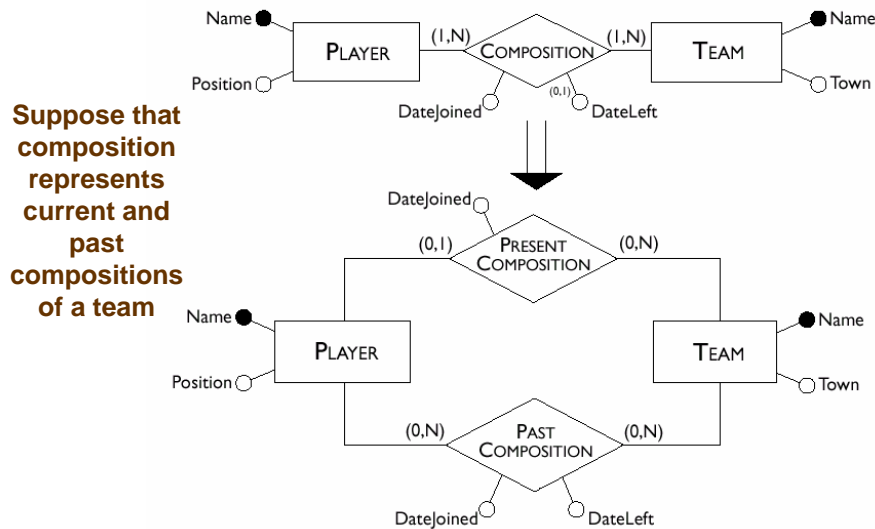
## Deletion of Multi-Valued Attribute



## Merging Entities



## Partitioning of a Relationship



## Selecting a Primary Key

- Every relation must have a unique primary key.
- The criteria for this decision are as follows:
  - ✓ Attributes with null values cannot form primary keys;
  - ✓ One/few attributes is preferable to many attributes;
  - ✓ Internal key preferable to external ones (weak entity);
  - ✓ A key that is used by many operations to access the instances of an entity is preferable to others.
- At this stage, if none of the candidate keys satisfies the above requirements, it may be best to introduce a new attribute (e.g., social insurance #, student #,...)

CSC343 – Introduction to Databases

Database Design — 31

## Translation into a Logical Schema

- The second step of logical design consists of a translation between different data models.
- Starting from an E-R schema, an equivalent relational schema is constructed. By “equivalent”, we mean a schema capable of representing the same information.
- We will deal with the translation problem systematically, beginning with the fundamental case, that of entities linked by many-to-many relationships.

CSC343 – Introduction to Databases

Database Design — 32



## Many-to-Many Relationships

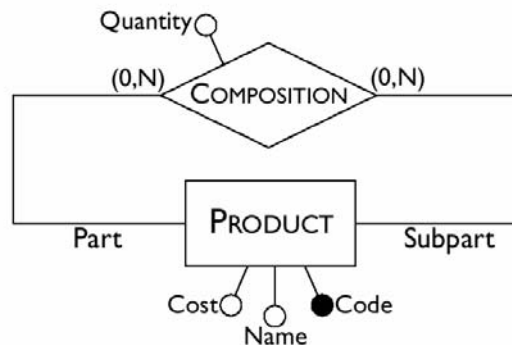


Employee(Number, Surname, Salary)

Project(Code, Name, Budget)

Participation(Number, Code, StartDate)

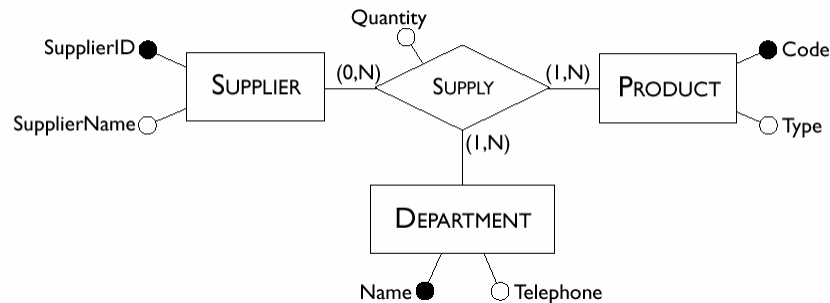
## Many-to-Many Recursive Relationships



Product(Code, Name, Cost)

Composition(Part, SubPart, Quantity)

## Ternary Relationships



Supplier(SupplierID, SupplierName)

Product(Code, Type)

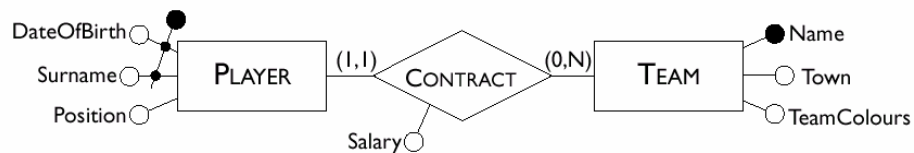
Department(Name, Telephone)

Supply(Supplier, Product, Department,  
Quantity)

CSC343 – Introduction to Databases

Database Design — 35

## One-to-Many Relationships



Player(Surname, DateOfBirth, Position)

Team(Name, Town, TeamColours)

Contract(PlayerSurname, PlayerDateOfBirth, Team,  
Salary)

OR

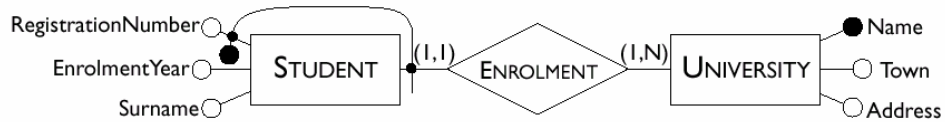
Player(Surname, DateOfBirth, Position, TeamName,  
Salary)

Team(Name, Town, TeamColours)

CSC343 – Introduction to Databases

Database Design — 36

## Weak Entities



Student(RegistrationNumber,  
University, Surname, EnrolmentYear)  
 University(Name, Town, Address)

## One-to-One Relationships



Head(Number, Name, Salary, Department,  
 StartDate)

Department(Name, Telephone, Branch)

OR

Head(Number, Name, Salary, StartDate)

Department(Name, Telephone, HeadNumber,  
 Branch)

## Optional One-to-One Relationships

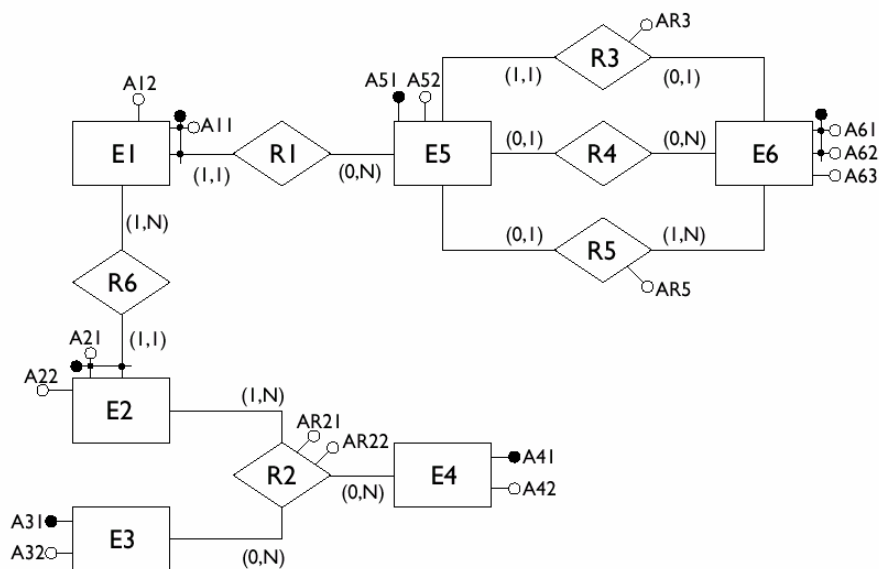


Employee(Number, Name, Salary)  
 Department(Name, Telephone, Branch, Head, StartDate)

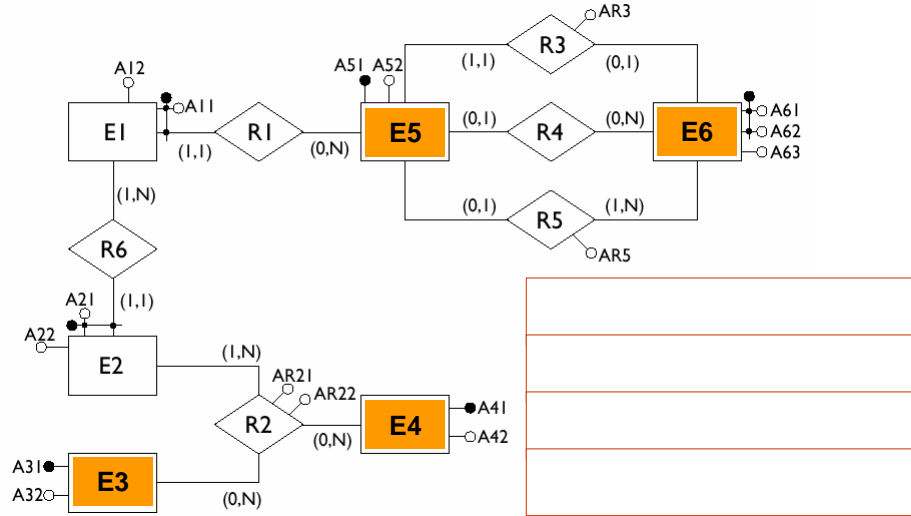
*Or, if both entities are optional*

Employee(Number, Name, Salary)  
 Department(Name, Telephone, Branch)  
 Management(Head, Department, StartDate)

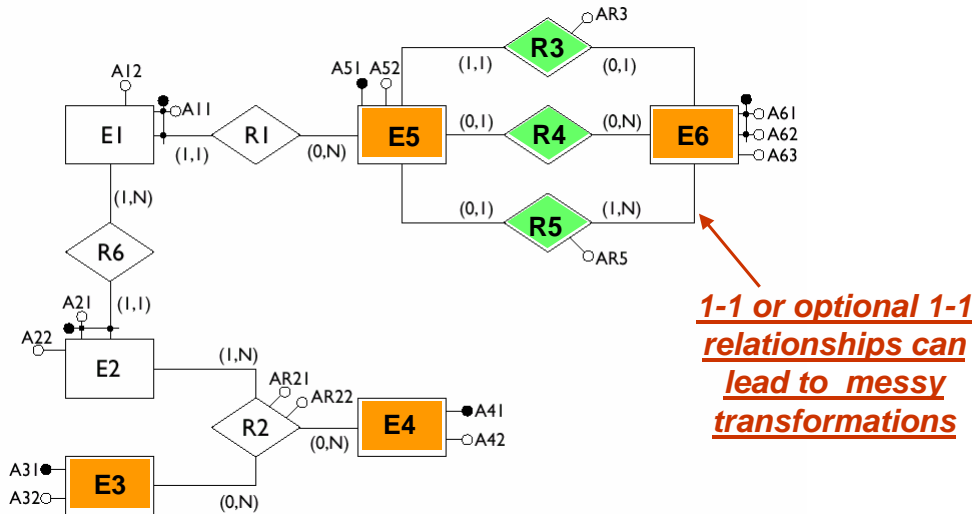
## A Sample ER Schema



## Entities with Internal Identifiers

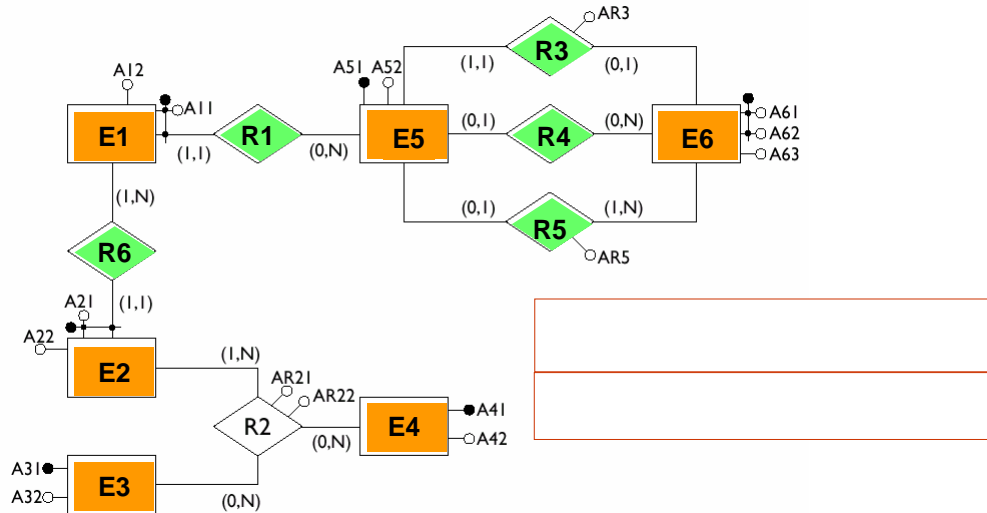


## 1-1 and Optional 1-1 Relationships

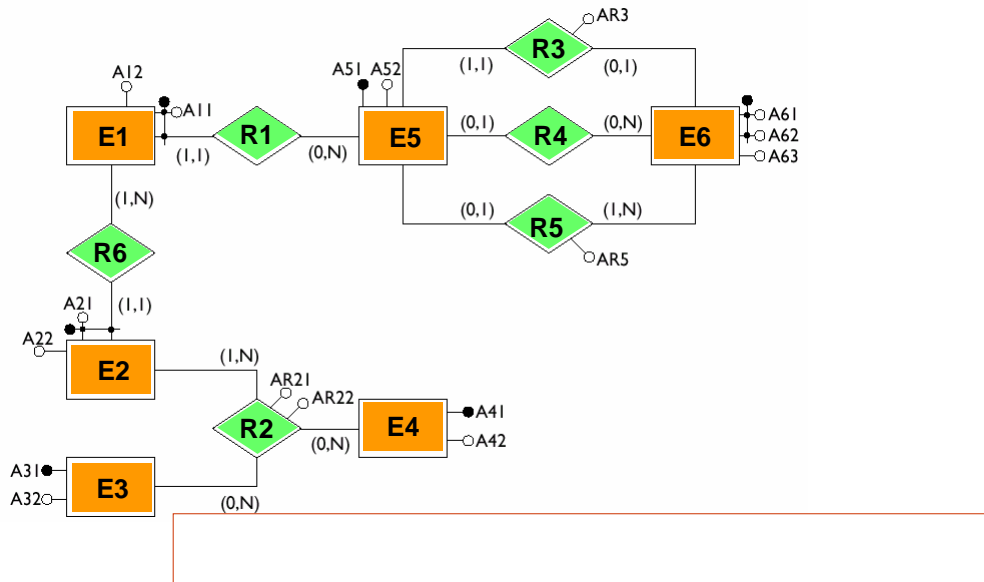


**1-1 or optional 1-1 relationships can lead to messy transformations**

# Weak Entities



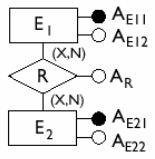
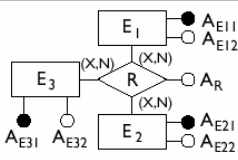
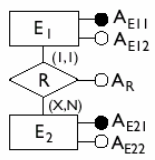
# Many-to-Many Relationships



## Result of the Translation

|  |
|--|
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

## Summary of Transformation Rules

| Type   | Initial schema  | Possible translation  |
|--|---|---|
| Binary<br>many-to-many<br>relationship                         |  | $E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$  |
| Ternary<br>many-to-many<br>relationship                        |  | $E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $E_3(\underline{A_{E31}}, A_{E32})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, \underline{A_{E31}}, A_R)$ |
| One-to-many<br>relationship with<br>mandatory<br>participation |  | $E_1(\underline{A_{E11}}, A_{E12}, A_{E21}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$   |

## ...More Rules...

| Type   | Initial schema | Possible translation   |
|--|----------------|--|
| One-to-many relationship with optional participation |                | $E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(A_{E11}, \underline{A_{E21}}, A_R)$ <p>Alternatively:</p> $E_1(\underline{A_{E11}}, A_{E21}, A_{E21}^*, A_R^*)$ $E_2(\underline{A_{E21}}, A_{E22})$ |
| Relationship with external identifiers               |                | $E_1(\underline{A_{E12}}, \underline{A_{E21}}, A_{E11}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$  |

## ...Even More Rules...

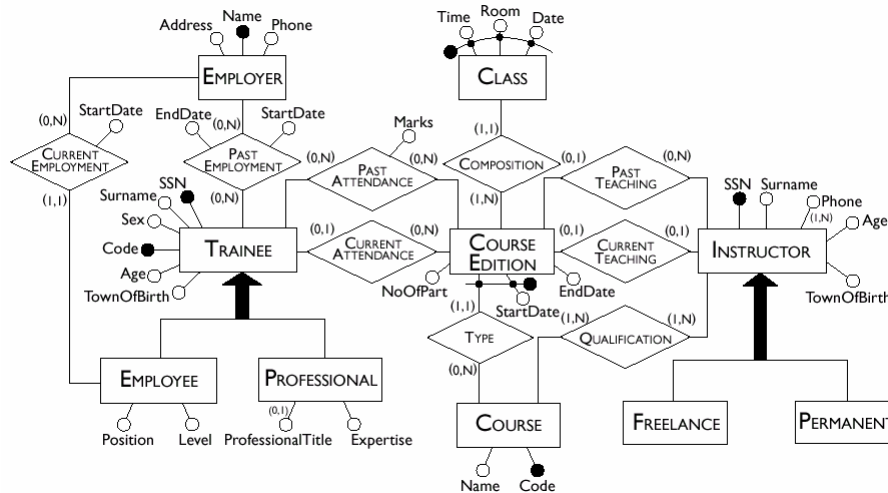
| Type   | Initial schema | Possible translation  |
|--|----------------|---|
| One-to-one relationship with mandatory participation for both entities |                | $E_1(\underline{A_{E11}}, \underline{A_{E12}}, \underline{A_{E21}}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$ <p>Alternatively:</p> $E_2(\underline{A_{E21}}, A_{E22}, \underline{A_{E11}}, A_R)$ $E_1(\underline{A_{E11}}, A_{E12})$ |
| One-to-one relationship with optional participation for one entity     |                | $E_1(\underline{A_{E11}}, \underline{A_{E12}}, \underline{A_{E21}}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$   |



## ...and the Last One...

| Type  | Initial schema | Possible translation   |
|---|----------------|--|
| One-to-one relationship with optional participation for both entities |                | $E_1(\underline{A_{E11}}, A_{E21})$ $E_2(\underline{A_{E21}}, A_{E22}, A_{E11}^*, A_R^*)$ <p>Alternatively:</p> $E_1(\underline{A_{E11}}, A_{E12}, A_{E21}^*, A_R^*)$ $E_2(\underline{A_{E21}}, A_{E22})$ <p>Alternatively:</p> $E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$ |

## The Training Company Revisited



## Operational Requirements, Revisited

- **operation 1:** insert a new trainee including all his or her data (to be carried out approximately 40 times a day);
- **operation 2:** assign a trainee to an edition of a course (50 times a day);
- **operation 3:** insert a new instructor, including all his or her data and the courses he or she is qualified to teach (twice a day);
- **operation 4:** assign a qualified instructor to an edition of a course (15 times a day);
- **operation 5:** display all the information on the past editions of a course with title, class timetables and number of trainees (10 times a day);
- **operation 6:** display all the courses offered, with information on the instructors who are qualified to teach them (20 times a day);
- **operation 7:** for each instructor, find the trainees all the courses he or she is teaching or has taught (5 times a week);
- **operation 8:** carry out a statistical analysis of all the trainees with all the information about them, about the editions of courses they have attended and the marks obtained (10 times a month).

CSC343 – Introduction to Databases

Database Design — 51

## Database Load

Table of volumes

| Concept           | Type | Volume |
|-------------------|------|--------|
| Class             | E    | 8000   |
| CourseEdition     | E    | 1000   |
| Course            | E    | 200    |
| Instructor        | E    | 300    |
| Freelance         | E    | 250    |
| Permanent         | E    | 50     |
| Trainee           | E    | 5000   |
| Employee          | E    | 4000   |
| Professional      | E    | 1000   |
| Employer          | E    | 8000   |
| PastAttendance    | R    | 10000  |
| CurrentAttendance | R    | 500    |
| Composition       | R    | 8000   |
| Type              | R    | 1000   |
| PastTeaching      | R    | 900    |
| CurrentTeaching   | R    | 100    |
| Qualification     | R    | 500    |
| CurrentEmployment | R    | 4000   |
| PastEmployment    | R    | 10000  |

CSC343 – Introduction to Databases

Table of operations

| Operation   | Type | Frequency    |
|-------------|------|--------------|
| Operation 1 | I    | 40 per day   |
| Operation 2 | I    | 50 per day   |
| Operation 3 | I    | 2 per day    |
| Operation 4 | I    | 15 per day   |
| Operation 5 | I    | 10 per day   |
| Operation 6 | I    | 20 per day   |
| Operation 7 | I    | 5 per day    |
| Operation 8 | B    | 10 per month |

Database Design — 52

## Access Tables

The attribute *NumberOfParticipants* in *CourseEdition* can be derived from relationships *CurrentAttendance*, *PastAttendance*.

Operation 2 with redundancy

| Concept        | Type | Acc | Type |
|----------------|------|-----|------|
| Trainee        | E    | 1   | R    |
| CurrentAtt'nce | R    | 1   | W    |
| CourseEdition  | E    | 1   | R    |
| CourseEdition  | E    | 1   | W    |

Operation 2 without redundancy

| Concept        | Type | Acc | Type |
|----------------|------|-----|------|
| Trainee        | E    | 1   | R    |
| CurrentAtt'nce | R    | 1   | W    |

Operation 5 with redundancy

| Concept       | Type | Acc | Type |
|---------------|------|-----|------|
| CourseEdition | E    | 5   | R    |
| Type          | R    | 5   | R    |
| Course        | E    | 1   | R    |
| Composition   | R    | 40  | R    |
| Class         | E    | 40  | R    |

Operation 5 without redundancy

| Concept       | Type | Acc | Type |
|---------------|------|-----|------|
| CourseEdition | E    | 5   | R    |
| Type          | R    | 5   | R    |
| Course        | E    | 1   | R    |
| Composition   | R    | 40  | R    |
| Class         | E    | 40  | R    |
| PastAtt'nce   | E    | 50  | R    |

## Analysis of Redundancy

- From the access tables we obtain (giving double weight to the write accesses):
  - ✓ presence of redundancy: for operation 2 we have 100 read disk accesses and 200 write disk accesses per day; for operation 5 we have 910 read accesses per day, for a total of 1,210 disk accesses per day;
  - ✓ without redundancy: for operation 2 we have 50 read accesses per day and 100 write accesses per day; for operation 5, we have 1,410 read accesses per day, for a total of 1,560 accesses per day.
- Thus, redundancy makes sense in this case, so we leave **NumberOfParticipants** as an attribute of the entity **CourseEdition**.

## Removing Generalizations

- For the generalization on instructors:
  - ✓ the relevant operations make no distinction between the child entities and these entities have no specific attributes;
  - ✓ we can therefore delete the child entities and add an attribute *Type* to the parent entity.
- For the generalization on trainees:
  - ✓ the relevant operations make no distinction between the child entities, but these entities have specific attributes;
  - ✓ we can therefore leave all the entities and add two relationships to link each child with the parent entity: in this way, we will have no attributes with possible null values on the parent entity and the dimension of the relations will be reduced.

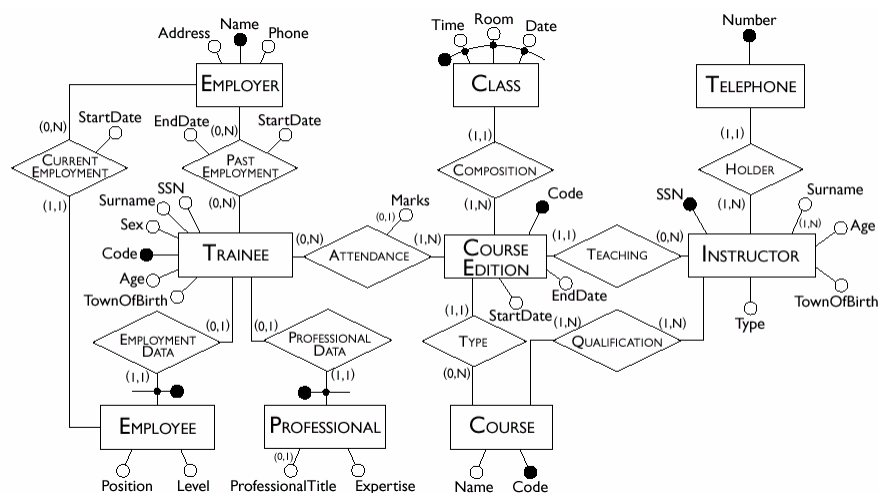
## Partitioning and Merging of Concepts

- The relationships *PastTeaching* and *PresentTeaching* can be merged since they describe similar concepts between which the operations make no difference. A similar consideration applies to the relationships *PastAttendance* and *PresentAttendance*.
- The multi-valued attribute *Telephone* can be removed from the *Instructor* entity by introducing a new entity *Telephone* linked by a one-to-many relationship to the *Instructor* entity.

## Choice of Main Identifiers

- **Trainee** entity:
  - ✓ there are two identifiers: the social security number and the internal code;
  - ✓ it is far preferable to choose the latter: a social security number will require several bytes whereas an internal code, which serves to distinguish between 5000 occurrences, requires a few bytes.
- **CourseEdition** entity:
  - ✓ it is identified externally by the **StartDate** attribute and by the **Course** entity;
  - ✓ we can see however that we can easily generate for each edition a code from the course code: this code is simpler and can replace the external identifier.

## After Restructuring



## Translation into the Relational Model

**CourseEdition**(Code, StartDate, EndDate, Course, Instructor)  
**Class**(Time, Room, Date, Edition)  
**Instructor**(SSN, Surname, Age, TownOfBirth, Type)  
**Telephone**(Number, Instructor)  
**Course**(Code, Name)  
**Qualification**(Course, Instructor)  
**Trainee**(Code, SSN, Surname, Age, TownOfBirth, Sex)  
**Attendance**(Trainee, Edition, Marks\*)  
**Employer**(Name, Address, Telephone)  
**PastEmployment**(Trainee, Employer, StartDate, EndDate)  
**Professional**(Trainee, Expertise, ProfessionalTitle\*)  
**Employee**(Trainee, Level, Position, Employer, StartDate)

CSC343 – Introduction to Databases

Database Design — 59

## Logical Design Using CASE Tools

- The logical design phase is partially supported by database design tools:
  - ✓ the translation to the relational model is carried out by such tools semi-automatically;
  - ✓ the restructuring step is difficult to automate and CASE tools provide little or no support for it.
- Most commercial CASE tools will generate automatically SQL code for the creation of the database.
- Some tools allow direct connection with a DBMS and can construct the corresponding database automatically.
- [**CASE** = Computer-Aided Software Engineering]

CSC343 – Introduction to Databases

Database Design — 60

The screenshot displays the Logic Works EF/wm/ERX interface. On the left, a logical data model is shown with the following tables and attributes:

- Employee\_Project**: Emp\_Id: NUMBER, Name: VARCHAR2(20)
- Employee**: Emp\_Id: NUMBER, Dept\_Id: NUMBER, Name: VARCHAR2(20), Salary: NUMBER, Age: NUMBER
- Department**: Dept\_Id: NUMBER, Name: VARCHAR2(20), Telephone: NUMBER
- Project**: Name: VARCHAR2(20), Budget: NUMBER, Deadline: DATE
- Manager**: Emp\_Id: NUMBER, Dept\_Id: NUMBER, Room: VARCHAR2(20), Name: VARCHAR2(20)
- Building**: Name: VARCHAR2(20), City: VARCHAR2(20), Address: VARCHAR2(20)

Relationships are indicated by lines connecting the tables. A primary key is shown on Emp\_Id in the Employee table. A foreign key relationship is shown between Dept\_Id in Employee and Dept\_Id in Department. Another foreign key relationship is shown between Dept\_Id in Manager and Dept\_Id in Department.

On the right, the 'ORACLE Schema Generation Report' is displayed, showing the following SQL code:

```

CREATE TABLE Employee (
  Emp_Id          NUMBER NOT NULL,
  Dept_Id        NUMBER NOT NULL,
  Name           VARCHAR2(20) NULL,
  Salary         NUMBER NULL,
  Age           NUMBER NULL,
  PRIMARY KEY (Emp_Id) );

CREATE TABLE Project (
  Name           VARCHAR2(20) NOT NULL,
  Budget        NUMBER NULL,
  Deadline      DATE NULL,
  PRIMARY KEY (Name) );

CREATE TABLE Employee_Project (
  Emp_Id          NUMBER NOT NULL,
  Name           VARCHAR2(20) NOT NULL,
  PRIMARY KEY (Emp_Id, Name) );

```

## Logical Design with a CASE Tool