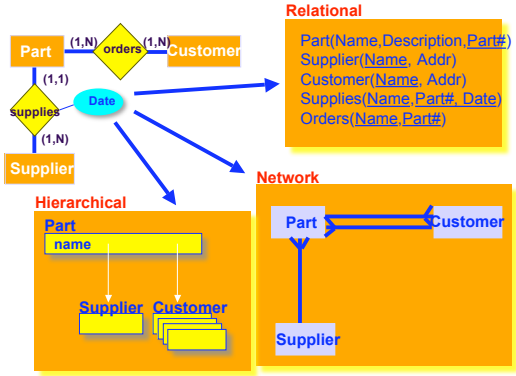
Week 10: Database Design

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

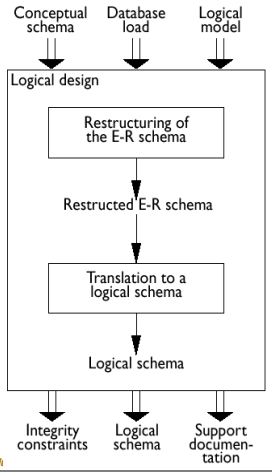
Designing a Database Schema



(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:
  - not all the constructs of the Entity-Relationship model can be translated naturally into the relational model;
  - 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.

Database Design Process



Logical Design Steps

- It is helpful to divide the design into two steps:
  - ***Restructuring of the Entity-Relationship schema***, based on criteria for the optimization of the schema and the simplification of the following step;
  - ***Translation into the logical model***, based on the features of the logical model (in our case, the relational model).

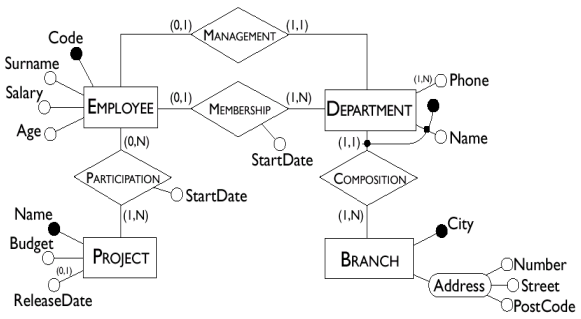
Performance Analysis

- 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).
- There are many other cost models depending on use and type of DB
  - ✓ Warehouse (OLAP - On-Line Analysis Processing)
  - ✓ Operational DB (OLTP - On-Line Transaction

Employee-Department Example

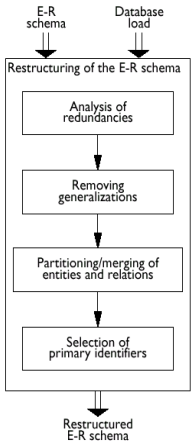


Typical Operations

- 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.
- Need operations and their volume/frequency

Workload Design

- During initial design and requirements analysis
  - ✓Estimate operations and frequency
  - ✓Gross estimates at best
- After database is operational
  - ✓Tools record actual workload characteristics

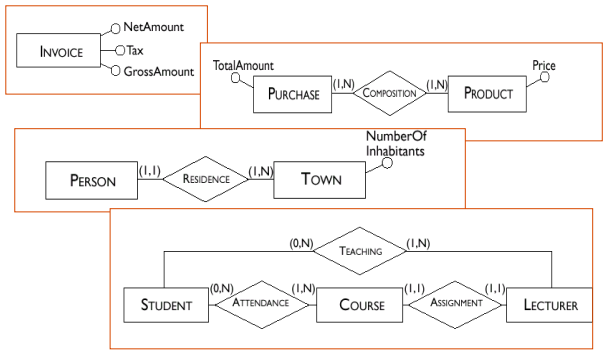


Analysis Steps

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.

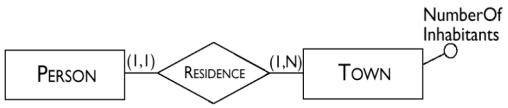
Examples of Redundancies



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.

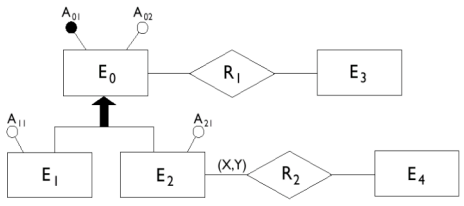
Cost Comparison: An Example



In this schema the attribute *NumberOfInhabitants* is redundant.

Removing Generalizations

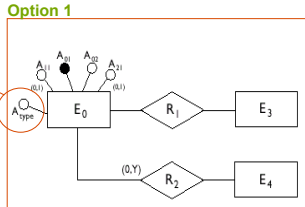
- The relational model does not allow direct representation of generalizations that may be present in an E-R diagram.
- For example, here is an ER schema with generalizations:



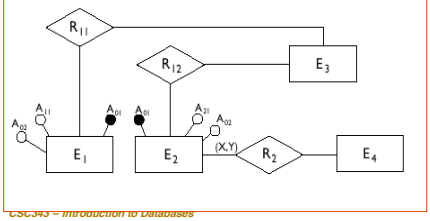
Possible Restructurings

Note!

Option 1

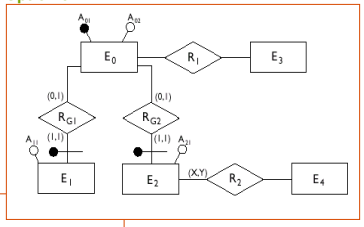


Option 2

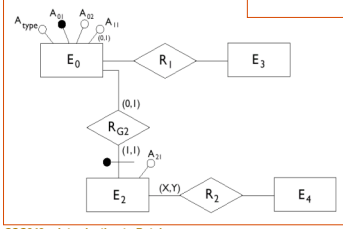


...Two More...

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)

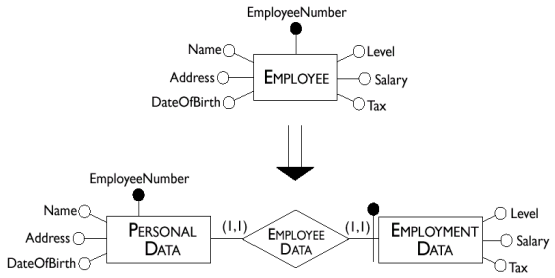
Partitioning and Merging of Entities and Relationships

- 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.

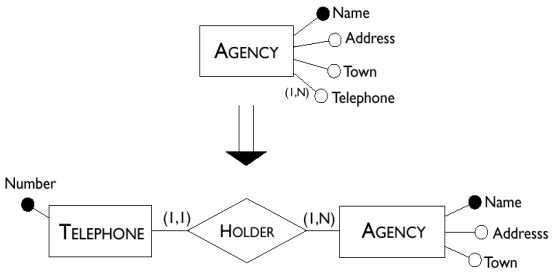
- The same criteria with those discussed for redundancies are valid in making a decision about this type of restructuring.

Example of Partitioning

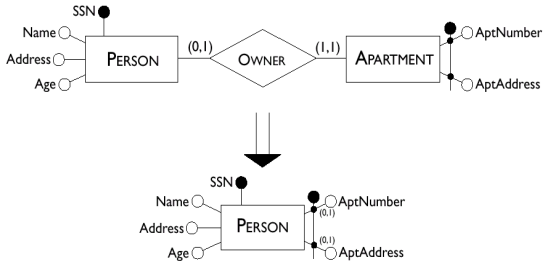


Recall this is a weak entity

Deletion of Multi-Valued Attribute

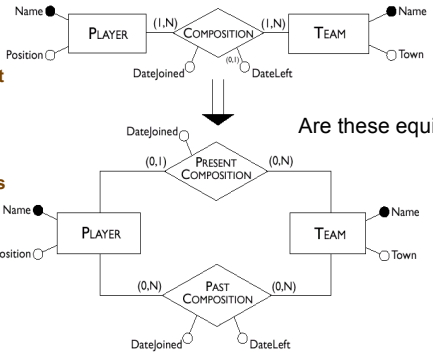


Merging Entities



Two candidate keys

Partitioning of a Relationship



Suppose that composition represents current and past compositions of a team

Are these equivalent?

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

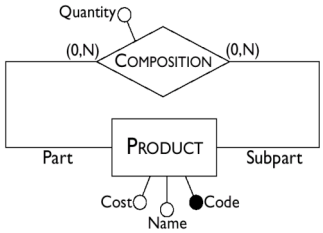
Translation into a Logical Schema

- 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.

Many-to-Many Relationships

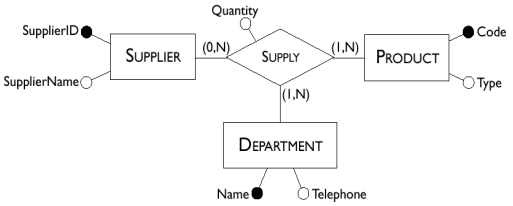
Employee(<u>Number</u>, Surname, Salary)
Project(<u>Code</u>, Name, Budget)
Participation(<u>Number</u>, <u>Code</u>, StartDate)

Many-to-Many Recursive Relationships



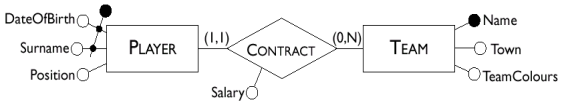
Product(<u>Code</u>, Name, Cost)
Composition(<u>Part</u>, <u>SubPart</u>, Quantity)

Ternary Relationships



Supplier(<u>SupplierID</u>, SupplierName)
Product(<u>Code</u>, Type)
Department(<u>Name</u>, Telephone)
Supply(<u>Supplier</u>, <u>Product</u>, <u>Department</u>, Quantity)

One-to-Many Relationships

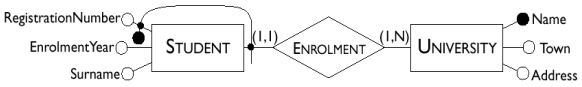


Player(<u>Surname</u>, <u>DateOfBirth</u>, Position)
Team(<u>Name</u>, Town, TeamColours)
Contract(<u>PlayerSurname</u>, <u>PlayerDateOfBirth</u>, Team, Salary)

OR

Player(<u>Surname</u>, <u>DateOfBirth</u>, Position, TeamName, Salary)
Team(<u>Name</u>, Town, TeamColours)

Weak Entities



Student(<u>RegistrationNumber</u>, <u>University</u>, Surname, EnrolmentYear)
University(<u>Name</u>, Town, Address)

One-to-One Relationships

Head(<u>Number</u>, Name, Salary, Department, StartDate)
Department(<u>Name</u>, Telephone, Branch)

OR

Head(<u>Number</u>, Name, Salary, StartDate)
Department(<u>Name</u>, Telephone, HeadNumber, Branch)

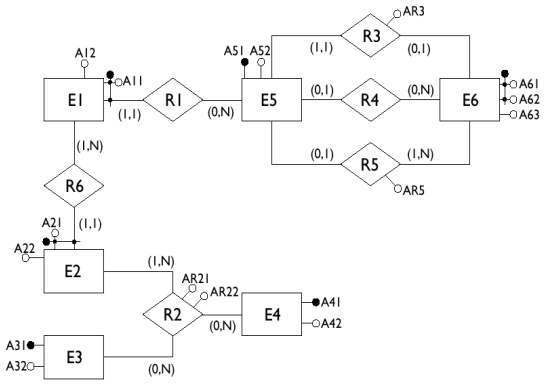
Optional One-to-One Relationships

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

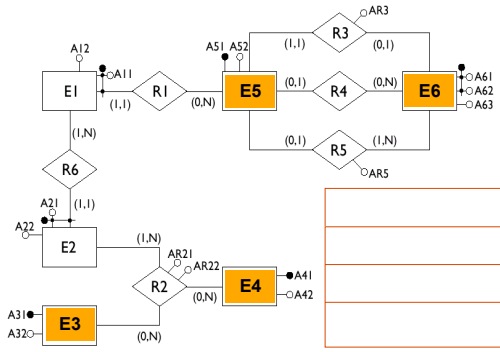
Or, if both entities are optional

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

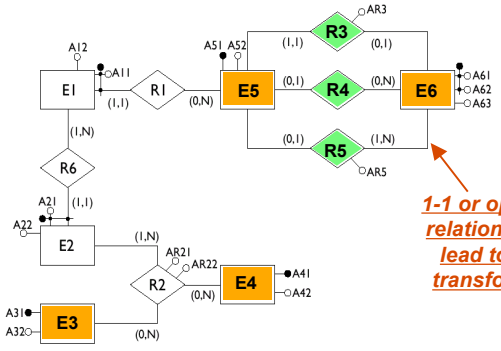
A Sample ER Schema



Entities with Internal Identifiers

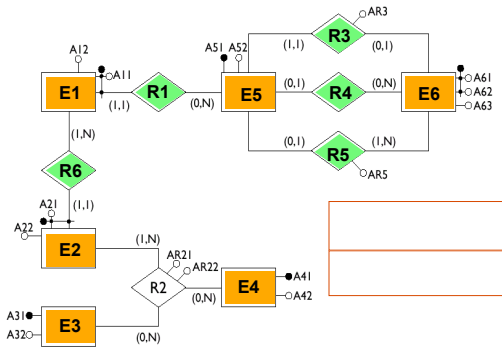


1-1 and Optional 1-1 Relationships

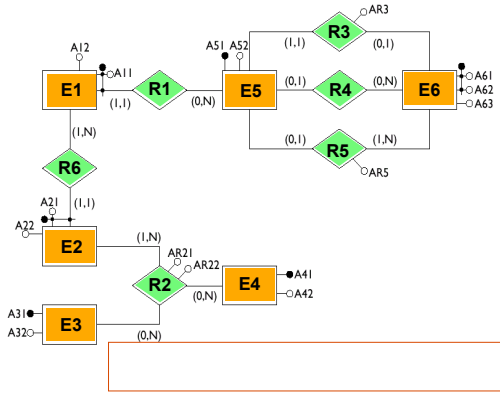


1-1 or optional 1-1 relationships can lead to messy transformations

Weak Entities

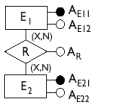
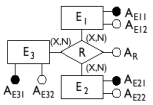
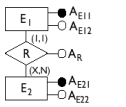


Many-to-Many Relationships

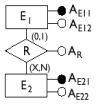
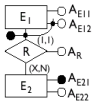


Homework - Fill in a Translation

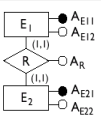
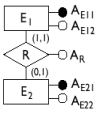
Summary of Transformation Rules

Type	Initial schema	Possible translation
Binary many-to-many 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 many-to-many 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 relationship with mandatory participation		$E_1(\underline{A_{E11}}, A_{E12}, A_{E21}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

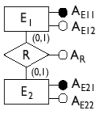
...More Rules...

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

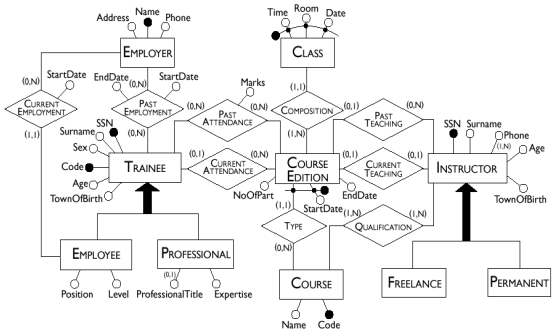
…Even More Rules...

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

…and the Last One...

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

The Training Company Revisited



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]

Logical Design with a CASE Tool