**XX. Database Design**

**Databases and DBMS**

- Data Models, Hierarchical, Network, Relational
- Restructuring an ER schema
- Performance analysis
- Analysis of Redundancies, Removing generalizations
- Translation into a Relational Schema
- Normal Forms and Normalization

---

**Databases**

- A database is a collection of persistent data shared by a number of applications.
- Databases have been founded on the concept of data independence: Applications should not have to know the organization of the data or the access strategy employed to fetch the data. ⇒ Query facility, query optimization
- Databases also founded on the concept of data sharing: Applications should be able to work on the same data concurrently, without knowing of each others’ existence. ⇒ Database transactions

---

**Conventional Files vs Databases**

<table>
<thead>
<tr>
<th>Advantage</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Good for data integration</td>
<td>- High cost</td>
</tr>
<tr>
<td>- Allow for more flexible formats</td>
<td>- Data duplication</td>
</tr>
</tbody>
</table>

**The future is with databases!**

---

**Types of Databases**

- **Conventional databases** -- (relational, network, hierarchical) consist of records of many different record types (database looks like a collection of files)
- **Object-Oriented databases** -- database consists of objects (and possibly associated programs); database schema consists of classes (which can be objects too).
- **Multimedia databases** -- database can store formatted data (i.e., records) but also text, pictures, ...
- **Active databases** -- database includes event-condition-action rules
- ...more...

---

**Database Management Systems**

- **DML** -- data manipulation language
- **DDL** -- data definition language (allows definition of database schema)
- **4GL** -- fourth generation language, useful for declarative query processing, report generation, ...
The Hierarchical Data Model

Database consists of hierarchical record structures: a field may have as value a list of records; every record has at most one parent.

The Network Data Model

A database now consists of records with pointers (links) to other records. Offers a navigational view of a database.

cycles of links are allowed

The Relational Data Model

A database consists of sets of records or (equivalently) sets of tuples (relations) or (equivalently) tables; no links allowed in the database.

Every tuple is an element of exactly one relation and is identified uniquely by a primary key.

Comparing Data Models

The oldest DBMSs were hierarchical, dating back to the mid-60s. IMS (IBM product) is the most popular among them.

The network data model came next (early '70s). Emphasis on "navigating" -- chasing pointers -- around a database.

Network model was found to be in many respects too implementation-oriented, machine-dependent.

The relational model is the most recent arrival (early '80s) and it has taken over the database market. Relational databases are considered simpler than their hierarchical and network cousins because they don't allow any links/pointers (which are necessarily machine-dependent).

Designing a Database Schema

(Relational) Database Design

Given a class or Entity-Relationship diagram (or "schema") produced during requirements analysis, 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.
Logical Design Steps

It is helpful to divide the logical 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 two parameters:
  - 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 tuple) requires 1 disk access
  - A Write operation (for one tuple) 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 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>
<thead>
<tr>
<th>Concept</th>
<th>Type</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch</td>
<td>E</td>
<td>10</td>
</tr>
<tr>
<td>Department</td>
<td>E</td>
<td>80</td>
</tr>
<tr>
<td>Employee</td>
<td>E</td>
<td>2000</td>
</tr>
<tr>
<td>Project</td>
<td>E</td>
<td>500</td>
</tr>
<tr>
<td>Composition</td>
<td>R</td>
<td>80</td>
</tr>
<tr>
<td>Membership</td>
<td>R</td>
<td>1000</td>
</tr>
<tr>
<td>Management</td>
<td>R</td>
<td>80</td>
</tr>
<tr>
<td>Participation</td>
<td>R</td>
<td>6000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operation</th>
<th>Type</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation 1</td>
<td>I</td>
<td>50 per day</td>
</tr>
<tr>
<td>Operation 2</td>
<td>I</td>
<td>100 per day</td>
</tr>
<tr>
<td>Operation 3</td>
<td>I</td>
<td>10 per day</td>
</tr>
<tr>
<td>Operation 4</td>
<td>B</td>
<td>2 per day</td>
</tr>
</tbody>
</table>

I - Interactive  
B - Batch

---

**Table of Volumes and Operations**

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

<table>
<thead>
<tr>
<th>Concept</th>
<th>Type</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch</td>
<td>E</td>
<td>10</td>
</tr>
<tr>
<td>Department</td>
<td>E</td>
<td>80</td>
</tr>
<tr>
<td>Employee</td>
<td>E</td>
<td>2000</td>
</tr>
<tr>
<td>Project</td>
<td>E</td>
<td>500</td>
</tr>
<tr>
<td>Composition</td>
<td>R</td>
<td>80</td>
</tr>
<tr>
<td>Membership</td>
<td>R</td>
<td>1000</td>
</tr>
<tr>
<td>Management</td>
<td>R</td>
<td>80</td>
</tr>
<tr>
<td>Participation</td>
<td>R</td>
<td>6000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operation</th>
<th>Type</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation 1</td>
<td>I</td>
<td>50 per day</td>
</tr>
<tr>
<td>Operation 2</td>
<td>I</td>
<td>100 per day</td>
</tr>
<tr>
<td>Operation 3</td>
<td>I</td>
<td>10 per day</td>
</tr>
<tr>
<td>Operation 4</td>
<td>B</td>
<td>2 per day</td>
</tr>
</tbody>
</table>

I - Interactive  
B - Batch

---

**Navigation Schema for Operation 2**

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.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Type</th>
<th>Accesses</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td>Entity</td>
<td>1</td>
<td>R</td>
</tr>
<tr>
<td>Membership</td>
<td>Relationship</td>
<td>1</td>
<td>R</td>
</tr>
<tr>
<td>Department</td>
<td>Entity</td>
<td>1</td>
<td>R</td>
</tr>
<tr>
<td>Participation</td>
<td>Relationship</td>
<td>3</td>
<td>R</td>
</tr>
<tr>
<td>Project</td>
<td>Entity</td>
<td>3</td>
<td>R</td>
</tr>
</tbody>
</table>

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

Average # of participations and projects per employee

---

**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.
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.

Load and Operations for the Example

Table of volumes

<table>
<thead>
<tr>
<th>Concept</th>
<th>Type</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Town</td>
<td>E</td>
<td>200</td>
</tr>
<tr>
<td>Person</td>
<td>E</td>
<td>1000000</td>
</tr>
<tr>
<td>Residence</td>
<td>R</td>
<td>1000000</td>
</tr>
</tbody>
</table>

Table of operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Type</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation 1</td>
<td>500 per day</td>
<td></td>
</tr>
<tr>
<td>Operation 2</td>
<td>2 per day</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Concept</th>
<th>Type</th>
<th>Accesses</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>Entity</td>
<td>1</td>
<td>W</td>
</tr>
<tr>
<td>Residence</td>
<td>Relationship</td>
<td>5000</td>
<td>R</td>
</tr>
</tbody>
</table>

Operation 2
- print all the data of a town (including the number of inhabitants).

Table of Accesses, without Redundancy

<table>
<thead>
<tr>
<th>Concept</th>
<th>Type</th>
<th>Accesses</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>Entity</td>
<td>1</td>
<td>W</td>
</tr>
<tr>
<td>Residence</td>
<td>Relationship</td>
<td>1</td>
<td>W</td>
</tr>
</tbody>
</table>

Operation 2
- print all the data of a town (including the number of inhabitants).

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:

![ER Schema with Generalizations](image1)

Possible Restructurings

Option 1

Option 2

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 is total (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 of $E_2$.
- Option 3 is useful when the generalization is not total 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.
Deletion of Multi-Valued Attribute

Merging Entities

Partitioning of a Relationship

Selecting a Primary Identifier

Translation into a Logical Schema

Many-to-Many Relationships

Every relation must have a unique primary identifier.

The criteria for this decision are as follows.

- Attributes with null values cannot form primary identifiers;
- One or few attributes is preferable to many attributes;
- An internal identifier is preferable to an external one;
- An identifier that is used by many operations to access the occurrences of an entity is preferable to others.

At this stage, if none of the candidate identifiers satisfies the above requirements, it is possible to introduce a further attribute to the entity. This attribute will hold special values (often called codes) generated solely for the purpose of identifying occurrences of the entity.
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)

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)

Entities with External Identifiers

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

Entities with External Identifiers

Many-to-Many Relationships

Result of the Translation
Summary of Transformation Rules

<table>
<thead>
<tr>
<th>Type</th>
<th>Initial Schema</th>
<th>Possible Translucions</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-to-one relationship</td>
<td>$T_x \times T_y$</td>
<td>$T_x[A_1] 	imes T_y[A_2]$</td>
</tr>
<tr>
<td>One-to-many relationship</td>
<td>$T_x \times T_y$</td>
<td>$T_x[A_1] 	imes T_y[A_2]$</td>
</tr>
<tr>
<td>One-to-many relationship with optional participation</td>
<td>$T_x \times T_y$</td>
<td>$T_x[A_1] 	imes T_y[A_2]$</td>
</tr>
</tbody>
</table>

...More Rules...

<table>
<thead>
<tr>
<th>Type</th>
<th>Initial Schema</th>
<th>Possible Translucions</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-to-many relationship with optional participation</td>
<td>$T_x \times T_y$</td>
<td>$T_x[A_1] 	imes T_y[A_2]$</td>
</tr>
<tr>
<td>Relationship with optional identification</td>
<td>$T_x \times T_y$</td>
<td>$T_x[A_1] 	imes T_y[A_2]$</td>
</tr>
</tbody>
</table>

...Even More Rules...

<table>
<thead>
<tr>
<th>Type</th>
<th>Initial Schema</th>
<th>Possible Translucions</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-to-one relationship with inventory participation, or both entities</td>
<td>$T_x \times T_y$</td>
<td>$T_x[A_1] 	imes T_y[A_2]$</td>
</tr>
<tr>
<td>One-to-one relationship with optional participation for one entity</td>
<td>$T_x \times T_y$</td>
<td>$T_x[A_1] 	imes T_y[A_2]$</td>
</tr>
</tbody>
</table>

...and the Last One...

<table>
<thead>
<tr>
<th>Type</th>
<th>Initial Schema</th>
<th>Possible Translucions</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-to-one relationship with optional participation for one entity</td>
<td>$T_x \times T_y$</td>
<td>$T_x[A_1] 	imes T_y[A_2]$</td>
</tr>
</tbody>
</table>

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 times 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).
**Database Load**

### Table of volumes

<table>
<thead>
<tr>
<th>Concept</th>
<th>Type</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>E</td>
<td>8000</td>
</tr>
<tr>
<td>CourseEdition</td>
<td>E</td>
<td>1000</td>
</tr>
<tr>
<td>Course</td>
<td>E</td>
<td>200</td>
</tr>
<tr>
<td>Instructor</td>
<td>E</td>
<td>300</td>
</tr>
<tr>
<td>Freelance</td>
<td>E</td>
<td>250</td>
</tr>
<tr>
<td>Permanent</td>
<td>E</td>
<td>50</td>
</tr>
<tr>
<td>Trainee</td>
<td>E</td>
<td>5000</td>
</tr>
<tr>
<td>Employee</td>
<td>E</td>
<td>4000</td>
</tr>
<tr>
<td>Professional</td>
<td>E</td>
<td>1000</td>
</tr>
<tr>
<td>Employer</td>
<td>E</td>
<td>8000</td>
</tr>
<tr>
<td>PastAttendance</td>
<td>R</td>
<td>10000</td>
</tr>
<tr>
<td>CurrentAttendance</td>
<td>R</td>
<td>500</td>
</tr>
<tr>
<td>Composition</td>
<td>R</td>
<td>8000</td>
</tr>
<tr>
<td>Type</td>
<td>R</td>
<td>1000</td>
</tr>
<tr>
<td>PastTeaching</td>
<td>R</td>
<td>900</td>
</tr>
<tr>
<td>CurrentTeaching</td>
<td>R</td>
<td>100</td>
</tr>
<tr>
<td>Qualification</td>
<td>R</td>
<td>500</td>
</tr>
<tr>
<td>CurrentEmployment</td>
<td>R</td>
<td>4000</td>
</tr>
<tr>
<td>PastEmployment</td>
<td>R</td>
<td>10000</td>
</tr>
</tbody>
</table>

**Access Tables**

### Table of operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Type</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation 1</td>
<td>R</td>
<td>50 per day</td>
</tr>
<tr>
<td>Operation 2</td>
<td>I</td>
<td>50 per day</td>
</tr>
<tr>
<td>Operation 3</td>
<td>I</td>
<td>3 per day</td>
</tr>
<tr>
<td>Operation 4</td>
<td>I</td>
<td>15 per day</td>
</tr>
<tr>
<td>Operation 5</td>
<td>I</td>
<td>10 per day</td>
</tr>
<tr>
<td>Operation 6</td>
<td>I</td>
<td>20 per day</td>
</tr>
<tr>
<td>Operation 7</td>
<td>I</td>
<td>5 per day</td>
</tr>
<tr>
<td>Operation 8</td>
<td>B</td>
<td>10 per month</td>
</tr>
</tbody>
</table>

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

#### Operation 2 with redundancy

<table>
<thead>
<tr>
<th>Concept</th>
<th>Type</th>
<th>Acc</th>
<th>Acc Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trainee</td>
<td>E</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>CurrentAttendance</td>
<td>R</td>
<td>1</td>
<td>W</td>
</tr>
<tr>
<td>CourseEdition</td>
<td>E</td>
<td>1</td>
<td>R</td>
</tr>
<tr>
<td>PresentAttendance</td>
<td>R</td>
<td>1</td>
<td>W</td>
</tr>
</tbody>
</table>

#### Operation 5 with redundancy

<table>
<thead>
<tr>
<th>Concept</th>
<th>Type</th>
<th>Acc</th>
<th>Acc Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CourseEdition</td>
<td>E</td>
<td>5</td>
<td>R</td>
</tr>
<tr>
<td>Course</td>
<td>E</td>
<td>5</td>
<td>R</td>
</tr>
<tr>
<td>Type</td>
<td>R</td>
<td>5</td>
<td>R</td>
</tr>
<tr>
<td>Composition</td>
<td>R</td>
<td>40</td>
<td>R</td>
</tr>
<tr>
<td>Class</td>
<td>E</td>
<td>40</td>
<td>R</td>
</tr>
<tr>
<td>PastAttendance</td>
<td>R</td>
<td>50</td>
<td>R</td>
</tr>
</tbody>
</table>

**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 1210 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 1410 read accesses per day, for a total of 1560 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.
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.

What is a Good Relational Schema?

- Some relational schemas are "better" representations than others. What are the criteria we can use to decide whether a diagram is better than another? Should we have more/fewer relations as opposed to attributes?
  - Enter normal forms

An attribute \( a \) (functionally) depends on a set of attributes \( a_1, a_2, \ldots, a_n \) if these determine uniquely the value of \( a \) for every tuple of the relation where they appear together

\[ a_1, a_2, \ldots, a_n \rightarrow a \]

Example

- For the relation
  
  \[
  \text{Course}(\text{name}, \text{title}, \text{instrName}, \text{rmName}, \text{address})
  \]

  E.g. (csc340, "Analysis and Design", JM, BA1130, "40 St George Street")

  the title attribute depends on the name attribute.
  Likewise, the address attribute depends on the rmName attribute.

  \[ \text{name} \rightarrow \text{title} \text{, also } \text{rmName} \rightarrow \text{address} \]
More Examples

- Consider Supplier(S#, SName, Status, Address)
- Here SName, Status, Address functionally depend on S#
  because S# uniquely determines the values of the other
  attributes of the Supplier relation
- S# --> SName, Status, Address
- Likewise, assuming that Lastname, Firstname uniquely
  identify people, we have
  Lastname, Firstname --> Salary, Address
- In general, for any relation, non-key attributes should
  functionally depend on key ones.

Un-Normalized Relations?

- Normalization helps produce a schema that is not redundant
  and does not suffer from anomalies
- Consider Emp(Emp#, EName, Address, Dept, Mngr#)
  with Emp# --> EName, Address, Dept, Mngr#
- Insertion anomaly: We can't add a new department and its
  manager until we have an employee in that department.
- Deletion anomaly: If we delete only employee in a
  department, we lose information about the department.
- Update anomaly: If we update Mngr# of one tuple, we must
  do it for all, otherwise we have an inconsistent database.
  It's easy to end up with an inconsistent database when it's not normalized!

Identifying Functional Dependencies

- Think about the meaning of different attributes.
- Alternatively, if you are given sample values for the
  attributes of the relation (see below), check to ensure that
  every combination of values for \( a_1, a_2, \ldots, a_n \) has the
  same associated value for a multi-valued attributes or any composite attributes.

What functional dependencies are appropriate here?

An Example

- For the Course(name, title, instrName*, studentNo*, addr),
  example, assume that addr is a composite attribute
  consisting of strNm, strNo, city and postalCode:

  \[\text{addr} \rightarrow \text{name}, \text{title}, \text{instrName*}, \text{studentNo*}, \text{addr}\]
  \[\text{Course} \rightarrow \text{name}, \text{title}\]
  \[\text{CourseStudt} \rightarrow \text{name}, \text{studentNo}\]
  \[\text{CourseInstr} \rightarrow \text{name}, \text{instrName}\]
  \[\text{CourseAddr} \rightarrow \text{name}, \text{strNm}, \text{strNo}, \text{city}, \text{postalCode}\]

- Note: The process outlined earlier does ensure that there are
  no multi-valued attributes for the relational schema generated
  from a conceptual schema.

Normalizing Relational Schemas: 1NF

- A relation is in First Normal Form (1NF) if it does not include any
  multi-valued attributes or any composite attributes.
  e.g., consider the relation

  \[\text{Course} \rightarrow \text{name}, \text{title}, \text{instrName*}, \text{studentNo*}, \text{addr}\]

- Course is not in 1NF because of two attribute groups that
  functionally depend on its

- To place a relation in 1NF, take each multi-valued attribute or
  composite attribute and promote it into a relation in its own right.

Normalizing Relational Schemas: 2NF

- An relation is in Second Normal Form (2NF) if it is in 1NF
  and, moreover, all non-key attributes depend on all
  elements of its key, rather than a subset.

- Consider Room(str, no, bldNms, room#, cp, AVEquip)
  Room is not in 2NF because its address attributes
  functionally depend on its bldNm key element, rather
  than the combination (room#, bldNm)

- To place a 1NF relation into 2 NF, take each non-key
  attribute that does not depend on the full key and move it
to a relation identified by the partial key

  \[\rightarrow \text{Room} \rightarrow \text{bldNm, room#, cp, AVEquip}\]
  \[\text{Building} \rightarrow \text{bldNm, str, no}\]
A relation is in Third Normal Form (3NF) if it is in 2NF and no non-key attribute depends on another non-key attribute. Assuming that each course has only one instructor (why do we need this assumption?), Course is not in 3NF because instrDept depends on instrNm:

Course(name, year, sem, instrNm, instrDept, enrol#)

To place a 2NF relation into 3NF, move each non-key attribute that depends on other non-key attributes to another relation:

--- Course(name, year, sem, instrNm, enrol#)

Instructor(name, dept)