XX. Database Design

Databases and DBMS
Data Models, Hierarchical, Network, Relational
Database Design
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

Files
Advantages -- many already exist; good for simple applications; very efficient
Disadvantages -- data duplication; hard to evolve; hard to build for complex applications

Databases
Advantages -- Good for data integration; allow for more flexible formats (not just records)
Disadvantages -- high cost; drawbacks in a centralized facility

The future is with databases!

Database Concepts

- **Data model** -- defines a set of data structures along with associated operations, for building and accessing a database
- **Database management system (DBMS)** -- generic tool for building, accessing, updating and managing a database
  - E.g., Oracle, DB2, Access,… are all relational DBMSs
- **Database schema** -- describes the types and structure of the data stored in the database; consists of one or more relation schemas
- **Transaction** -- an atomic operation on a database; looks like a procedure but has different semantics: when called, it either completes its execution, or aborts and undoes all changes it made to the database.
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

Hierarchical

Part

(1,N) supplies

(1,N)

Customer

Supplier

Part (Name, Description, Part#)

Supplier (Name, Addr)

Customer (Name, Addr)

Supplies (Name, Part#, Date)

Orders (Name, Part#)

Relational

Network

Part

(1,N) orders

Customer

Date

(1,1)

Supplier

Part

name

Supplies

Customer

Supplier

(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>1900</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.

Examples of Redundancies

![Diagram showing examples of redundancies in an Entity-Relationship schema.](image-url)
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>I</td>
<td>500 per day</td>
</tr>
<tr>
<td>Operation 2</td>
<td>I</td>
<td>2 per day</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

**Operation 1**

<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>
<tr>
<td>Town</td>
<td>Entity</td>
<td>1</td>
<td>W</td>
</tr>
</tbody>
</table>

**Operation 2**

<table>
<thead>
<tr>
<th>Concept</th>
<th>Type</th>
<th>Accesses</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Town</td>
<td>Entity</td>
<td>1</td>
<td>R</td>
</tr>
</tbody>
</table>
### Table of Accesses, without Redundancy

#### Operation 1

<table>
<thead>
<tr>
<th>Concept</th>
<th>Type</th>
<th>Accesses</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person Residence</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

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

### 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 diagram with generalizations]

Option 1

Option 2

Note!
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.

Example of Partitioning
Deletion of Multi-Valued Attribute

Merging Entities
Partitioning of a Relationship

Suppose that Composition represents current and past compositions of a team.

Selecting a Primary Identifier

- 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.
Translation into a Logical Schema

- The second step of logical design corresponds to 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(Number, Surname, salary)
Project(Code, Name, Budget)
Participation(Number, Code, StartDate)
Many-to-Many Recursive Relationships

Product\(\text{Code, Name, Cost}\)

Composition\(\text{Part, SubPart, Quantity}\)

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
1-1 and Optional 1-1 Relationships

Entities with External Identifiers
Many-to-Many Relationships

R2(A21, A11, A51, A31, A41, AR21, AR22)

Result of the Translation

E1(A11, A51, A12)
E2(A21, A11, A51, A22)
E3(A31, A32)
E4(A41, A42)
E6(A61, A62, A63)
R2(A21, A11, A51, A31, A41, AR21, AR22)
### Summary of Transformation Rules

<table>
<thead>
<tr>
<th>Type</th>
<th>Initial schema</th>
<th>Possible translation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Binary many-to-many relation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$E_1 \rightarrow A_{E11}$</td>
<td>$E_1(A_{E11}, A_{E12})$</td>
</tr>
<tr>
<td></td>
<td>$R \rightarrow A_R$</td>
<td>$E_2(A_{E21}, A_{E22})$</td>
</tr>
<tr>
<td></td>
<td>$E_3 \rightarrow A_{E32}$</td>
<td>$R(A_{E11}, A_{E21}, A_R)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Ternary many-to-many relation</strong></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$E_1 \rightarrow A_{E11}$</td>
<td>$E_1(A_{E11}, A_{E12})$</td>
</tr>
<tr>
<td></td>
<td>$E_3 \rightarrow A_{E32}$</td>
<td>$E_2(A_{E21}, A_{E22})$</td>
</tr>
<tr>
<td></td>
<td>$E_3 \rightarrow A_{E32}$</td>
<td>$E_3(A_{E31}, A_{E32})$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>One-to-many relation with mandatory participation</strong></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$E_1 \rightarrow A_{E11}$</td>
<td>$E_1(A_{E11}, A_{E12})$</td>
</tr>
<tr>
<td></td>
<td>$R \rightarrow A_R$</td>
<td>$E_2(A_{E21}, A_{E22})$</td>
</tr>
<tr>
<td></td>
<td>$E_3 \rightarrow A_{E32}$</td>
<td>$E_3(A_{E31}, A_{E32})$</td>
</tr>
</tbody>
</table>

---

### More Rules...

<table>
<thead>
<tr>
<th>Type</th>
<th>Initial schema</th>
<th>Possible translation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>One-to-many relationship with optional participation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$E_1 \rightarrow A_{E11}$</td>
<td>$E_1(A_{E11}, A_{E12})$</td>
</tr>
<tr>
<td></td>
<td>$R \rightarrow A_R$</td>
<td>$E_2(A_{E21}, A_{E22})$</td>
</tr>
<tr>
<td></td>
<td>$E_3 \rightarrow A_{E32}$</td>
<td>$R(A_{E11}, A_{E21}, A_R)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Alternatively:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$E_1(A_{E11}, A_{E12}, A_{E21}, A_{E31})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$E_2(A_{E21}, A_{E22})$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Relationship with external identifiers</strong></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$E_1 \rightarrow A_{E11}$</td>
<td>$E_1(A_{E11}, A_{E12})$</td>
</tr>
<tr>
<td></td>
<td>$R \rightarrow A_R$</td>
<td>$E_2(A_{E21}, A_{E22})$</td>
</tr>
<tr>
<td></td>
<td>$E_3 \rightarrow A_{E32}$</td>
<td>$E_3(A_{E31}, A_{E32})$</td>
</tr>
</tbody>
</table>
### Even More Rules

<table>
<thead>
<tr>
<th>Type</th>
<th>Initial schema</th>
<th>Possible translation</th>
</tr>
</thead>
</table>
| One-to-one relationship with mandatory participation for both entities | \[\text{E}_1 \rightarrow \text{A}_{E1} \leftarrow \text{A}_{E12}, \text{E}_2 \rightarrow \text{A}_{E2} \leftarrow \text{A}_{E22}\] | \[E_1(\text{A}_{E1}, \text{A}_{E12}, \text{A}_{E12}, \text{A}_{A})
E_2(\text{A}_{E2}, \text{A}_{E22})
\text{Alternatively:} E_2(\text{A}_{E2}, \text{A}_{E22}, \text{A}_{E1}, \text{A}_{A})
E_1(\text{A}_{E1}, \text{A}_{E12})\] |

### and the Last One

<table>
<thead>
<tr>
<th>Type</th>
<th>Initial schema</th>
<th>Possible translation</th>
</tr>
</thead>
</table>
| One-to-one relationship with optional participation for both entities | \[\text{E}_1 \rightarrow \text{A}_{E1} \leftarrow \text{A}_{E12}, \text{E}_2 \rightarrow \text{A}_{E2} \leftarrow \text{A}_{E22}\] | \[E_1(\text{A}_{E1}, \text{A}_{E12}, \text{A}_{E2}, \text{A}_{A})
E_2(\text{A}_{E2}, \text{A}_{E22})
\text{Alternatively:} E_1(\text{A}_{E1}, \text{A}_{E12}, \text{A}_{E2}, \text{A}_{A})
E_2(\text{A}_{E2}, \text{A}_{E22})\] |
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).
**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>

**Table of operations**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Type</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation 1</td>
<td>I</td>
<td>40 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>2 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>

**Access Tables**

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>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trainee</td>
<td>E</td>
<td>1</td>
<td>R</td>
</tr>
<tr>
<td>CurrentAtt'nce</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>CourseEdition</td>
<td>E</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>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CourseEdition</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>Course</td>
<td>E</td>
<td>1</td>
<td>R</td>
</tr>
<tr>
<td>Composition</td>
<td>E</td>
<td>40</td>
<td>R</td>
</tr>
<tr>
<td>Class</td>
<td>E</td>
<td>40</td>
<td>R</td>
</tr>
</tbody>
</table>

**Operation 2 without redundancy**

<table>
<thead>
<tr>
<th>Concept</th>
<th>Type</th>
<th>Acc</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trainee</td>
<td>E</td>
<td>1</td>
<td>R</td>
</tr>
<tr>
<td>CurrentAtt'nce</td>
<td>R</td>
<td>1</td>
<td>W</td>
</tr>
</tbody>
</table>

**Operation 5 without redundancy**

<table>
<thead>
<tr>
<th>Concept</th>
<th>Type</th>
<th>Acc</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CourseEdition</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>Course</td>
<td>E</td>
<td>1</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>PastAtt'nce</td>
<td>E</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 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(\text{Code}, \text{StartDate}, \text{EndDate}, \text{Course}, \text{Instructor})

Class(\text{Time}, \text{Room}, \text{Date}, \text{Edition})

Instructor(\text{SSN}, \text{Surname}, \text{Age}, \text{TownOfBirth}, \text{Type})

Telephone(\text{Number}, \text{Instructor})

Course(\text{Code}, \text{Name})

Qualification(\text{Course}, \text{Instructor})

Trainee(\text{Code}, \text{SSN}, \text{Surname}, \text{Age}, \text{TownOfBirth}, \text{Sex})

Attendance(\text{Trainee}, \text{Edition}, \text{Marks}^*)

Employer(\text{Name}, \text{Address}, \text{Telephone})

PastEmployment(\text{Trainee}, \text{Employer}, \text{StartDate}, \text{EndDate})

Professional(\text{Trainee}, \text{Expertise}, \text{ProfessionalTitle}^*)

Employee(\text{Trainee}, \text{Level}, \text{Position}, \text{Employer}, \text{StartDate})
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

\[ \text{Course(name,title,instrName,rmName,address)}, \]

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,

name \( \rightarrow \) title, also rmName \( \rightarrow \) 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, Mgr#)
  with
  Emp1# --> EName, Address, Dept, Mgr#, Dept --> Mgr#
- 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 Mgr# 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 \( a \)

<table>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
<th>Instructor</th>
<th>Office</th>
<th>Tutors</th>
<th>Enrolment</th>
</tr>
</thead>
<tbody>
<tr>
<td>csc148</td>
<td>&quot;Intro.&quot;</td>
<td>Reiter</td>
<td>LP290G</td>
<td>4</td>
<td>133</td>
</tr>
<tr>
<td>csc228</td>
<td>&quot;DP&quot;</td>
<td>Clarke</td>
<td>SF285</td>
<td>3</td>
<td>124</td>
</tr>
<tr>
<td>csc328</td>
<td>&quot;Logic&quot;</td>
<td>Fich</td>
<td>SF254</td>
<td>3</td>
<td>85</td>
</tr>
<tr>
<td>csc324</td>
<td>&quot;PLs&quot;</td>
<td>Bonner</td>
<td>LP354</td>
<td>2</td>
<td>72</td>
</tr>
<tr>
<td>csc340</td>
<td>&quot;SA&quot;</td>
<td>Reiter</td>
<td>LP290G</td>
<td>2</td>
<td>121</td>
</tr>
<tr>
<td>csc408</td>
<td>&quot;SE&quot;</td>
<td>Clarke</td>
<td>SF254</td>
<td>3</td>
<td>88</td>
</tr>
<tr>
<td>csc434</td>
<td>&quot;DM&quot;</td>
<td>Fich</td>
<td>SF254</td>
<td>3</td>
<td>107</td>
</tr>
</tbody>
</table>

What functional dependencies are appropriate here?

---

**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 `Course(name, title, instrName*, studentNo*, addr)`
  - `Course` is not in 1NF because of two attribute groups that repeat (instructor and student groups)
- To place a relation in 1NF, take each multi-valued attribute or composite attribute and promote it into an relation in its own right.
An Example

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

=> Course(name, title)
CourseStudt(name, studentNo)
CourseInstr(name, instrName)
CourseAddr(name, strNm, strNo, city, 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: 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, bldNm, room#, cp, AVEquip)
Room is not in 2NF because its address attributes functionally depend on its bldgNm key element, rather than the combination (room#, bldgNm)

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

=> Room(bldgNm, room#, cp, AVEquip), Building(bldgNm, str, no)
Normalizing Relational Schemas: 3NF

- 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:

\[ \text{Course}(\text{name}, \text{year}, \text{sem}, \text{instrNm}, \text{instrDept}, \text{enrol#}) \]

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

\[ \Rightarrow \text{Course}(\text{name}, \text{year}, \text{sem}, \text{instrNm}, \text{enrol#}) \]

\[ \text{Instructor}(\text{name}, \text{dept}) \]