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
The Training Company Example
Normal Forms and Normalization of Relational Schemas

Advantages

- Good for data integration; allow for more flexible formulas (not just records)

Disadvantages

- High cost; inapplicable in a centralized facility

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 procedures); 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
- Deductive databases - database contains rules that can be fired automatically
- Hypertext databases - store and access efficiently HTML/XML documents; provide navigational facilities through a database, so that a user can retrieve and/or browse

* - not available commercially

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...

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
  ⇒ Need query processing facility, which generates automatically an access plan given a query
- 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 procedures defined in terms of atomic operations called transactions

Database Concepts

- Data model - defines a set of data structures along with associated operations, for building and accessing a database
  ✓ e.g., the relational model offers relations (tables) as data structure for building 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
  ✓ E.g., Employee (emp#, name, addr, sal, dept#, mng)
- 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.
  ✓ E.g., TransferFunds (fromAcct#, toAcct#, amount, date)
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

<table>
<thead>
<tr>
<th>Book</th>
<th>Novels</th>
<th>24.50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Novels</td>
<td>War &amp; Peace</td>
<td>15.99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Borrower</th>
<th>Status</th>
<th>38 Elm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Toronto</td>
<td>Jan 28, 1994</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Borrowing</th>
<th>Status</th>
<th>1:1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Toronto</td>
<td>Jan 28, 1994 to Feb 24, 1994</td>
</tr>
</tbody>
</table>

The Network Data Model

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

Customer

Order

<table>
<thead>
<tr>
<th>Order</th>
<th>Date</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>11/3/93</td>
<td>Status</td>
</tr>
<tr>
<td>Status</td>
<td>25/8/93</td>
<td>Status</td>
</tr>
</tbody>
</table>

Part

<table>
<thead>
<tr>
<th>Part</th>
<th>Name</th>
<th>Description</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>widget</td>
<td>12,980</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>nut</td>
<td>16,000doz.</td>
<td></td>
</tr>
<tr>
<td>Quantity</td>
<td>bolt</td>
<td>14,340doz</td>
<td></td>
</tr>
</tbody>
</table>

Comparing Data Models

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

- The network data model came next (early '70s). At the time of its proposal, it was viewed as a breakthrough. It emphasized the role of the database programmer as "navigator," chasing links (pointers, actually) around a database.

- But, the network model was found to be in many respects too implementation-oriented, not insulating sufficiently the programmer from implementation features of network DBMSs.

- 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 implementation-dependent).

Designing a Database Schema

- The aim of database design is to construct a relational schema that correctly and efficiently represents all of the information described by a class or Entity-Relationship diagram (or 'schema') produced during requirements analysis.

- From now, we'll only talk about transforming an E-R schema into a relational schema. Most of the transformation process applies for class diagrams as well.

- 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.
Database Design Process

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 can be 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 on the database);
  - 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 will 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 the record of an employee, including the department where she works, and the projects she works for.
- Operation 3: Find the records of all employees for a given 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 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

<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 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>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

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

Table of volumes

<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 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>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

<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 Relationship</td>
<td>1</td>
<td>R</td>
<td></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>

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 can contain various forms of redundancy.

Examples of Redundancies
Deciding About Redundancies

- The presence of a redundancy in a database may
  - 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 delete 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>
<thead>
<tr>
<th>Table of volumes</th>
<th>Table of operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept Type</td>
<td>Volume</td>
</tr>
<tr>
<td>Town</td>
<td>200</td>
</tr>
<tr>
<td>Person</td>
<td>1,000,000</td>
</tr>
<tr>
<td>Residence</td>
<td>100,000</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:**
  - Concept Type: Person
  - Accesses: Type W
  - Person: Entity
  - Residence: Relationship
  - Town: Entity

- **Operation 2:**
  - Concept Type: Person
  - Accesses: Type W
  - Person: Entity
  - Residence: Relationship
  - Town: Entity

Table of Accesses, without Redundancy

- **Operation 1:**
  - Concept Type: Person
  - Accesses: Type W
  - Person: Entity
  - Residence: Relationship

- **Operation 2:**
  - Concept Type: Person
  - Accesses: Type R
  - Person: Entity
  - Residence: Relationship

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 can improve performance, sometimes!
Removing Generalizations

The relational model does not allow the direct representation of generalizations that may be present in an E-R diagram. We need, therefore, to transform these constructs into other constructs that are easier to translate: entities and relationships.

For example, here is an E-R schema with generalizations:

![ER schema with generalizations]

Option 1
Option 2
Option 3
Option 4 (combination)

Possible Restructurings

General Rules For Removing Generalization

- Option 1 is convenient when the operations involve the occurrences and the attributes of E₀, E₁, and E₂ more or less in the same way.
- Option 2 is possible only if the generalization is total (i.e., every instance of E₀ is either an instance of E₁ or E₂) and is useful when there are operations that apply only to occurrences of E₁ or of E₂.
- Option 3 is useful when the generalization is not total and the operations refer to either occurrences and attributes of E₁ or E₂ 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 of Entities

Partitioning of a Relationship

Selection of Primary Identifiers

The criteria for this decision are as follows.

- Attributes with null values cannot form primary identifiers;
- One or few attribute identifiers are preferable to ones with many attributes;
- An internal identifier with few attributes is preferable to an external one, possibly involving many entities;
- 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** (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)

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)

Optional One-to-One Relationships

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

Result of the Translation
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 times, dates 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).
Access Tables

The attribute NumberOfParticipants in CourseEdition can be derived from the relationships CurrentAttendance and PastAttendance.

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>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>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>20 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

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 do not distinguish 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.

Analyze 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

- 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 StartDateTime 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.

Partitioning and Merging of Concepts

- The relationships PastTeaching and PresentTeaching can be merged since they describe similar concepts between which the relationships 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.
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*)
PartEmployment(Trainee, Employer, StartDate, EndDate)
Professional(Trainee, Expertise, ProfessionalTitle*)
Employee(Trainee, Level, Position, Employer, 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 Like?

- 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

Examples of Functional Dependencies

- 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.

Example: For the relation

\[
\text{Course}(\text{name}, \text{title}, \text{Instructor}, \text{room}, \text{address}),
\]

E.g., \( \text{name} \), \( \text{title} \), \( \text{Instructor} \), \( \text{room} \). The \( \text{title} \) attribute depends on the \( \text{name} \) attribute, Likewise, the \( \text{address} \) attribute depends on the \( \text{room} \) attribute.

\[
\text{name} \rightarrow \text{title}, \text{Instructor} \rightarrow \text{room}
\]
What’s Wrong with Un-Normalized Relations?

- Normalization helps produce a schema that is not redundant and does not suffer from anomalies.
- Consider Emp(Dept, Name, Address, Dept, Mngr#) with Emp# -> {Name, Address, Dept, Mngr#}, Dept -> Mngr#.
- Insertion anomaly: We can’t add information about a new department and its manager until we have an employee in that department.
- Deletion anomaly: If we delete the only employee in a department, we lose information about the department (e.g., its manager).
- Update anomaly: If we update the Mngr# attribute 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!

How Do We Identify Functional Dependencies?

- Think about the meaning of different attributes and try to think of situations where the value of a is not determined by the values of a₁, a₂, ..., aₙ.
- Alternatively, if you are given sample values for the attributes of the relation (see below), check to ensure that every combination of values for a₁, a₂, ..., aₙ has the same associated value for a.

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. Multi-valued 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 students 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 streetNm, streetNo, city and postalCode.

⇒ Course(name, title)
CourseStudt(name, studentNo*)
CourseInstr(name, instrName)
CourseAddr(name, streetNm, streetNo, city, postalCode)

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(street, number, bldgNm, room#, capacity, AVEquip)
- Room is not in 2NF because its addresses attributes functionally depend on its bldgNm key element, rather than the combination (room#, bldgNm)
- To place a 1NF relation into 2NF, move each non-key attribute that does not depend on the full key and move it to a relation identified by the partial key.

NORMALIZING RELATIONAL SCHEMAS: 3NF

- A relation is in Third Normal Form (3NF) if it is in 2NF and none of its non-key attributes depends on any other 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.
- To place a 2NF relation into 3NF, move each non-key attribute that depends on other non-key attributes to another relation.

NORMALIZING RELATIONAL SCHEMAS: 3NF

- A relation is in Third Normal Form (3NF) if it is in 2NF and none of its non-key attributes depends on any other 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.
- To place a 2NF relation into 3NF, move each non-key attribute that depends on other non-key attributes to another relation.