Chapter 4
Normalization

Data Normalization

• Formal process of decomposing relations with anomalies to produce smaller, *well-structured* and *stable* relations

• Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that *avoid unnecessary duplication of data*
Well-Structured Relations

- A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies
- Goal is to avoid *(minimize)* anomalies
  - **Insertion Anomaly** – adding new rows forces user to create duplicate data
  - **Deletion Anomaly** – deleting a row may cause loss of other data representing completely different facts
  - **Modification Anomaly** – changing data in a row forces changes to other rows because of duplication

**General rule of thumb: a table should not pertain to more than one entity type**

Example – Figure 4.2b

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Name</th>
<th>DeptName</th>
<th>Salary</th>
<th>CourseTitle</th>
<th>DateCompleted</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Margaret Simpson</td>
<td>Marketing</td>
<td>48,000</td>
<td>SPSS</td>
<td>6/19/201X</td>
</tr>
<tr>
<td>100</td>
<td>Margaret Simpson</td>
<td>Marketing</td>
<td>48,000</td>
<td>Surveys</td>
<td>10/7/201X</td>
</tr>
<tr>
<td>140</td>
<td>Alan Beeton</td>
<td>Accounting</td>
<td>52,000</td>
<td>Tax Acc</td>
<td>12/0/201X</td>
</tr>
<tr>
<td>110</td>
<td>Chris Lucero</td>
<td>Info Systems</td>
<td>43,000</td>
<td>Visual Basic</td>
<td>1/12/201X</td>
</tr>
<tr>
<td>110</td>
<td>Chris Lucero</td>
<td>Info Systems</td>
<td>43,000</td>
<td>C++</td>
<td>4/22/201X</td>
</tr>
<tr>
<td>190</td>
<td>Lorenzo Davis</td>
<td>Finance</td>
<td>55,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>Susan Martin</td>
<td>Marketing</td>
<td>42,000</td>
<td>SPSS</td>
<td>6/19/201X</td>
</tr>
<tr>
<td>150</td>
<td>Susan Martin</td>
<td>Marketing</td>
<td>42,000</td>
<td>Java</td>
<td>8/12/201X</td>
</tr>
</tbody>
</table>

**Question** – Is this a relation?

**Question** – What’s the primary key?
Anomalies in this Table

- **Insertion** – can’t enter a new employee without having the employee take a class
- **Deletion** – if we remove employee 140, we lose information about the existence of a Tax Acc class
- **Modification** – giving a salary increase to employee 100 forces us to update multiple records

**Why do these anomalies exist?**

Because there are two themes (entity types – what are they?) in this one relation (two themes, entity types, were combined). This results in duplication, and an unnecessary dependency between the entities

Functional Dependencies

- **Functional Dependency**: The value of one attribute (the *determinant*) determines the value of another attribute.
  - A \( \rightarrow \) B reads “Attribute B is functionally dependent on A”
  - A \( \rightarrow \) B means if two rows have same value of A they necessarily have same value of B
  - FDs are determined by **semantics**: You *can’t* say that a FD exists just by looking at data. But can say whether it does not exist by looking at data.
Quick Check

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Orlando</td>
<td>Male</td>
<td>35</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>Male</td>
<td>35</td>
</tr>
<tr>
<td>3</td>
<td>Jane</td>
<td>Female</td>
<td>31</td>
</tr>
<tr>
<td>4</td>
<td>Jane</td>
<td>Female</td>
<td>30</td>
</tr>
</tbody>
</table>

- Id → Name?
- Age → Gender?
- Name → Id?
- Name, Age → Id?

Functional Dependencies and Keys

- **Functional Dependency**: The value of one attribute (the *determinant*) determines the value of another attribute.
- **Candidate Key**
  - Attribute that uniquely identifies a row in a relation
  - Could be a combination of (*non-redundant*) attributes
  - Each non-key field is functionally dependent on every candidate key

![Diagram of EMPLOYEE1 table]
Figure 4-23: Representing Functional Dependencies (cont.)

EmpID \rightarrow ___________________________
EmpID, CourseTitle \rightarrow ___________________________

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Name</th>
<th>DeptName</th>
<th>Salary</th>
<th>CourseTitle</th>
<th>DateCompleted</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Margaret Simpson</td>
<td>Marketing</td>
<td>48,000</td>
<td>SPSS</td>
<td>6/19/201X</td>
</tr>
<tr>
<td>100</td>
<td>Margaret Simpson</td>
<td>Marketing</td>
<td>48,000</td>
<td>Surveys</td>
<td>10/7/201X</td>
</tr>
</tbody>
</table>

First Normal Form (1NF)

- Only atomic attributes (simple, single-value)
- A primary key has been identified
- *Every relation is in 1NF* by definition

- Fig. 4-2a *is not* in 1\textsuperscript{st} Normal Form (multivalued attributes) \(\Rightarrow\) it is not a relation
- Fig. 4-2b *is* in 1\textsuperscript{st} Normal form (but not in a well-structured relation)
**1NF Example**

<table>
<thead>
<tr>
<th>StudentId</th>
<th>StuName</th>
<th>CourseId</th>
<th>CourseName</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Mike</td>
<td>112</td>
<td>C++</td>
<td>A</td>
</tr>
<tr>
<td>100</td>
<td>Mike</td>
<td>111</td>
<td>Java</td>
<td>B</td>
</tr>
<tr>
<td>101</td>
<td>Susan</td>
<td>222</td>
<td>Database</td>
<td>A</td>
</tr>
<tr>
<td>140</td>
<td>Lorenzo</td>
<td>224</td>
<td>Graphics</td>
<td>B</td>
</tr>
</tbody>
</table>

### Practice Exercise #7, page #196

**TABLE 4-3** Sample Data for Parts and Vendors

<table>
<thead>
<tr>
<th>Part No</th>
<th>Description</th>
<th>Vendor Name</th>
<th>Address</th>
<th>Unit Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Logic chip</td>
<td>Fast Chips</td>
<td>Cupertino</td>
<td>10.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Smart Chips</td>
<td>Phoenix</td>
<td>8.00</td>
</tr>
<tr>
<td>5678</td>
<td>Memory chip</td>
<td>Fast Chips</td>
<td>Cupertino</td>
<td>3.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quality Chips</td>
<td>Austin</td>
<td>2.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Smart Chips</td>
<td>Phoenix</td>
<td>5.00</td>
</tr>
</tbody>
</table>

1. Convert this table to a relation (named PART SUPPLIER) in 1NF
2. Draw a relational schema for PART SUPPLIER and show the functional dependencies. Identify a candidate key.
3. Identify each of the following: an insert anomaly, a delete anomaly, and a modification anomaly.
Figure: 4-22 Steps in normalization

- Table with Multivalued attributes
  - First normal form (1NF)
  - Second normal form (2NF)
  - Third normal form (3NF)
  - Boyce-Codd normal form (BC-NF)
  - Fourth normal Form (4NF)
  - Fifth normal form (5NF)
  - Remove Multivalued Attributes
  - Remove Partial Dependencies
  - Remove remaining anomalies resulting from multiple candidate keys
  - Remove Multivalued Dependencies
  - Remove Remaining Anomalies
Second Normal Form (2NF)

- 1NF and no *partial* (functional) dependencies
- A partial dependency is when a non-key attribute depends on a *part of the primary key*
- Of course, you need a composite PK
- Fig. 4-2b is NOT in 2NF

**Functional Dependencies in Student**

Can represent FDs with arrows as above, or
- StudentId → StuName,
- CourseId → CourseName
- StudentId,CourseId → Grade (and StuName, CourseName)

Any partial FDs?
Functional Dependencies in Student

- StudentId → StuName,
- CourseId → CourseName
- StudentId, CourseId → Grade (and StuName, CourseName)

Therefore, NOT in 2nd Normal Form!!

2NF: Normalizing

- How do we convert the partial dependencies into normal ones? By breaking into more tables.

- Becomes … (notice above arrows mean functional dependency, below they mean FK constraints)
You Try It Now …

<table>
<thead>
<tr>
<th>SeriesId</th>
<th>EpisodeId</th>
<th>SeriesTitle</th>
<th>EpisodeTitle</th>
<th>AiringDate</th>
</tr>
</thead>
</table>

- List all FDs
- Eliminate partial FDs, if any
Third Normal Form

• 2NF and no transitive dependencies
• A transitive dependency is when a non-key attribute depends on another non-key attribute

3NF Example

<table>
<thead>
<tr>
<th>Course</th>
<th>SectNum</th>
<th>Classroom</th>
<th>Capacity</th>
</tr>
</thead>
</table>

• Classroom $\rightarrow$ Capacity
• Any partial FDs?
• Any transitive FDs?
### 3NF Example

<table>
<thead>
<tr>
<th>Course</th>
<th>SectNum</th>
<th>Classroom</th>
<th>Capacity</th>
</tr>
</thead>
</table>

- Classroom $\rightarrow$ Capacity **TRANSITIVE**
- Any partial FDs? **NO**
- Any transitive FDs? **YES**!
  - How do we eliminate it?
  - By breaking into its own table

### 3NF Normalization

<table>
<thead>
<tr>
<th>Course</th>
<th>SectNum</th>
<th>Classroom</th>
</tr>
</thead>
</table>

```
Classroom    Capacity
```
You Try It …

<table>
<thead>
<tr>
<th>StudentId</th>
<th>ProgramId</th>
<th>StudentName</th>
<th>ProgramName</th>
</tr>
</thead>
</table>

- Partial FDs? Eliminate, if any.
- Transitive FDs? Eliminate, if any.

Practice Exercise #15, page #198

<table>
<thead>
<tr>
<th>Insertion anomaly?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deletion anomaly?</td>
</tr>
<tr>
<td>Modification anomaly?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE 4-8</th>
<th>Shipment Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shipment#</td>
<td>Origin</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
</tr>
<tr>
<td>409</td>
<td>Seattle</td>
</tr>
<tr>
<td>618</td>
<td>Chicago</td>
</tr>
<tr>
<td>723</td>
<td>Boston</td>
</tr>
<tr>
<td>824</td>
<td>Denver</td>
</tr>
<tr>
<td>629</td>
<td>Seattle</td>
</tr>
</tbody>
</table>

1. Develop a diagram that shows the functional dependencies in the SHIPMENT relation.
2. In what normal form is SHIPMENT? Why?
3. Convert SHIPMENT to 3NF if necessary. Show the resulting table(s) with the sample data presented in SHIPMENT.
Figure: 4-22 Steps in normalization

1. Table with Multivalued attributes
2. First normal form (1NF)
3. Second normal form (2NF)
4. Third normal form (3NF)
5. Boyce-Codd normal form (BC-NF)
6. Fourth normal form (4NF)
7. Fifth normal form (5NF)

- Remove Multivalued Attributes
- Remove Partial Dependencies
- Remove Transitive Dependencies
- Remove remaining anomalies resulting from multiple candidate keys
- Remove Multivalued Dependencies
- Remove Remaining Anomalies
Further Normalization

• Boyce-Codd Normal form (BCNF)
  – Slight difference with 3NF
  – To be in 3NF but not in BNF, needs two composite candidate keys, with one attribute of one key depending on one attribute of the other
  – Not very common 😊
  – If a table contains only one candidate key, the 3NF and the BCNF are equivalent.

• Fourth Normal Form (4NF)
  – To break it, need to have multivalued dependencies, a generalization of functional dependencies

• Usually, if you’re in 3NF you’re in BCNF, 4NF, …

Logical Database Design

You have just learned and completed one of the most important concepts and theories, **integrity constraints** and **normalization**, for developing a quality of database.
After learning one of most important database concepts and theories... WHAT’S NEXT?

Steps of Database Development

User view-1  User view-2  User view-3  ...  User view-N

User interview & Integrated Model → Conceptual Schema (Model)

Logical Model (ERD or E/ERD)

(Six) Relations Transformation

NORMALIZATION (up to 3NF)

IMPLEMENTATION
Merging Relations

Merging Relations
(View Integration)

• In a project development process, there may be a number of separate E-R diagrams and user views created and some of them may be redundant.

• Therefore, some relations should be merged to remove the redundancy.
Merging Relations (View Integration - An example)

EMPLOYEE1( EmployeeID, Name, Address, Phone)
EMPLOYEE2(EmployeeID, Name, Address, Jobcode, No_Years)

EMPLOYEE(EmployeeID, Name, Address, Phone, Jobcode, No_Years)

Merging Relations (Problems on View Integration)

Issues to watch out for when merging entities from different ER models:

- Synonyms: Different names, same meaning.
- Homonyms: Same name, different meanings.
- Transitive Dependencies:
  ✓–even if relations are in 3NF prior to merging, they may not be after merging
Problems on View Integration

• Synonyms: Different names, same meaning.
  STUDENT1(StudentID, Name)
  STUDENT2(MatriculationNo, Name, Address)
  STUDENT(StudentNo, Name, Address)

• Homonyms: Same name, different meanings.
  STUDENT1(StudentID, Name, Address)
  STUDENT2(StudentID, Name, PhoneNo, Address)
  STUDENT(StudentID, Name, PhoneNo, Campus_Address, Permanent_Address)

• Transitive Dependencies
  STUDENT1(StudentID, Major)
  STUDENT2(StudentID, Advisor)

• The result is ...
  STUDENT(StudentID, Major, Advisor) ??NF

• Suppose that each major has exactly one advisor. After removing transitive dependencies:

  STUDENT_Major(StudentID, Major)
  Major_Advisor(Major, Advisor)