Normalization
Elmasri/Navathe ch 14
Padron-McCarthy/Risch ch 11

DATABASE DESIGN I - IDL300 Spring 2013
Sobhan badizamany
Department of Information Technology, Uppsala University

"Good" Relational database design

"Bad" Relational database design

Summary
By normalizing a database, we can ensure that the data is accurate and consistent.

- Relational database
- Normalization
- Normal forms of a database: 1NF, 2NF, 3NF, and 4NF

Information on the database's structure and relationships.
Normalization
Elmasri/Navathe ch 14
Padron-McCarthy/Risch ch 11

DATABASE DESIGN I - IDL300 Spring 2013
Sobhan badiozamany
Department of Information Technology, Uppsala University

"Good" Relational database design

"Bad" Relational database design
What is a bad relational database design?

What is relational database design?
- The grouping of attributes to form the relation schema.

How "grouping of attributes" affects the quality of the design?

In addition, redundant data causes logical design problems:
- Modification anomalies
- Insertion anomalies
- Deletion anomalies
A bad design causes data redundancy.

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Pnumber</th>
<th>Hours</th>
<th>Ename</th>
<th>Pname</th>
<th>Plocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>1</td>
<td>32.5</td>
<td>Smith, John B.</td>
<td>ProductX</td>
<td>Bellaire</td>
</tr>
<tr>
<td>123456789</td>
<td>2</td>
<td>7.5</td>
<td>Smith, John B.</td>
<td>ProductY</td>
<td>Sugarland</td>
</tr>
<tr>
<td>666884444</td>
<td>3</td>
<td>40.0</td>
<td>Narayan, Ramesh K.</td>
<td>ProductZ</td>
<td>Houston</td>
</tr>
<tr>
<td>453453453</td>
<td>1</td>
<td>20.0</td>
<td>English, Joyce A.</td>
<td>ProductX</td>
<td>Bellaire</td>
</tr>
<tr>
<td>453453453</td>
<td>2</td>
<td>20.0</td>
<td>English, Joyce A.</td>
<td>ProductY</td>
<td>Sugarland</td>
</tr>
<tr>
<td>333445555</td>
<td>2</td>
<td>10.0</td>
<td>Wong, Franklin T.</td>
<td>ProductY</td>
<td>Sugarland</td>
</tr>
<tr>
<td>333445555</td>
<td>3</td>
<td>10.0</td>
<td>Wong, Franklin T.</td>
<td>ProductZ</td>
<td>Houston</td>
</tr>
<tr>
<td>333445555</td>
<td>10</td>
<td>10.0</td>
<td>Wong, Franklin T.</td>
<td>Computerization</td>
<td>Stafford</td>
</tr>
<tr>
<td>333445555</td>
<td>20</td>
<td>10.0</td>
<td>Wong, Franklin T.</td>
<td>Reorganization</td>
<td>Houston</td>
</tr>
<tr>
<td>999887777</td>
<td>30</td>
<td>30.0</td>
<td>Zelaya, Alicia J.</td>
<td>Newbenefits</td>
<td>Stafford</td>
</tr>
<tr>
<td>999887777</td>
<td>10</td>
<td>10.0</td>
<td>Zelaya, Alicia J.</td>
<td>Computerization</td>
<td>Stafford</td>
</tr>
<tr>
<td>987987987</td>
<td>10</td>
<td>35.0</td>
<td>Jabbar, Ahmad V.</td>
<td>Computerization</td>
<td>Stafford</td>
</tr>
<tr>
<td>987987987</td>
<td>30</td>
<td>5.0</td>
<td>Jabbar, Ahmad V.</td>
<td>Newbenefits</td>
<td>Stafford</td>
</tr>
<tr>
<td>987654321</td>
<td>30</td>
<td>20.0</td>
<td>Wallace, Jennifer S.</td>
<td>Newbenefits</td>
<td>Stafford</td>
</tr>
<tr>
<td>987654321</td>
<td>20</td>
<td>15.0</td>
<td>Wallace, Jennifer S.</td>
<td>Reorganization</td>
<td>Houston</td>
</tr>
<tr>
<td>888665555</td>
<td>20</td>
<td>Null</td>
<td>Borg, James E.</td>
<td>Reorganization</td>
<td>Houston</td>
</tr>
</tbody>
</table>
In addition, redundant data causes logical design problems:

- **Modification anomalies**
- **Insertion anomalies**
- **Deletion anomalies**
Update Anomaly:
Assume we want to change the name of project number 30 [Pnumber=30] from "Newbenefits" to "software_prj".

This causes updating all records whose employees work on the project.
**Insert Anomaly:**

- Inserting an employee that is not assigned to project is not possible.
- Defining a project to which no employee is assigned yet is not possible.
Here is the Delete Anomaly:

- If you delete a project, all employees working in that project, who don't work in any other project, will be deleted from the database.
- If only one employee works on a given project, deleting her would delete the project from the database.
In addition, redundant data causes logical design problems:

- **Modification anomalies**
- **Insertion anomalies**
- **Deletion anomalies**
Informal design guidelines for relational databases

- Attributes of different entities (EMPLOYEES, DEPARTMENTs, PROJECTs) should not be mixed in the same relation.

- Foreign keys should be used to refer to other entities.

- Design a schema that can be explained easily relation by relation.

- Design a schema that does not suffer from the insertion, deletion and update anomalies.

- Relations should be designed such that their tuples will have as few NULL values as possible.
Let's apply the informal guidelines!

1-Separate attributes of different entities.

2-Use foreign keys to relate entities.

3-Eliminate insertion, update and deletion anomalies.

4-Eliminate the need for NULL values.

What is a good design here? and why?
Functional Dependencies

The first step towards formal specification of a “good” relational database design

Definition: an attribute that is a member in any of the candidate keys is called a prime attribute of it.

Attributes that are not part of any candidate key is called non-prime or non-key attributes.

Prime attributes:
- SSN
- Phone

Non-prime attributes:
- Name
- Address
- Phone
- Age

Full Functional Dependency (FFD)

- A functional dependency X → Y holds if there is no additional attribute E, Z ∈ R such that X → Y | E.
- Intuitively, a FFD is a dependency that does not contain any unnecessary attribute.

Example:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>E</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>Phone</td>
<td>Name</td>
<td>Address</td>
</tr>
<tr>
<td>SSN</td>
<td>Phone</td>
<td>Phone</td>
<td>Age</td>
</tr>
<tr>
<td>SSN</td>
<td>Phone</td>
<td>Name</td>
<td>Phone</td>
</tr>
</tbody>
</table>

Formal notation:
X → Y (X, E, Z)
Functional dependency

Functional dependencies (FDs):

- formally specify how good a relational design is.
- are constraints deduced from the mini-world concepts that describe the relationships between attributes.

A set of attributes X functionally determines a set of attributes Y if the value of X determines a unique value for Y.

It is written as $X \rightarrow Y$
A set of attributes X functionally determines a set of attributes Y if the value of X determines a unique value for Y. It is written as X → Y.

### Table 1: EMP_PROJ

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Phnumber</th>
<th>Hours</th>
<th>Ename</th>
<th>Phone</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>88866555</td>
<td>15.0</td>
<td>35.0</td>
<td>Zelaya, Alicia J.</td>
<td>123456789</td>
<td></td>
</tr>
<tr>
<td>987654321</td>
<td>20.0</td>
<td>10.0</td>
<td>Jabbar, Ahmad V.</td>
<td>203754468</td>
<td></td>
</tr>
<tr>
<td>987987987</td>
<td>30.0</td>
<td>10.0</td>
<td>Zelaya, Alicia J.</td>
<td>203754468</td>
<td></td>
</tr>
<tr>
<td>998877777</td>
<td>30.0</td>
<td>10.0</td>
<td>Wong, Franklin T.</td>
<td>203754468</td>
<td></td>
</tr>
<tr>
<td>334455555</td>
<td>10.0</td>
<td>10.0</td>
<td>Reorganization</td>
<td>203754468</td>
<td></td>
</tr>
<tr>
<td>334455555</td>
<td>10.0</td>
<td>10.0</td>
<td>ProductZ</td>
<td>203754468</td>
<td></td>
</tr>
<tr>
<td>453434534</td>
<td>10.0</td>
<td>10.0</td>
<td>Sugarland</td>
<td>203754468</td>
<td></td>
</tr>
</tbody>
</table>

### Figure 1.4

- **Redundancy**
  - EMP_PROJ
    - Ssn, Phnumber, Hours, Ename
    - Location: Houston, Stafford, Sugarland
  - Redundancy Reduction

### FDs (Functional Dependencies)

- FD1: Ssn → Phnumber
- FD2: Phnumber → Hours
- FD3: Hours → Ename

**Figure 14.3 (b)**

- EMP_PROJ
  - Ssn, Phnumber, Hours, Ename, Phone
  - Location: Houston, Stafford, Sugarland
  - Redundancy Reduction

**Figure 14.4**

- EMP_PROJ
  - Ssn, Phnumber, Hours, Ename, Phone
  - ProductX, ProductY, ProductZ
  - Location: Houston, Stafford, Sugarland
  - Redundancy Reduction
More formally

X -> Y holds if whenever two tuples have the same value for X, they must have the same value for Y, in other words:

- For any two tuples t1 and t2 in any relation instance r(R): If t1[X] = t2[X], then t1[Y] = t2[Y]

X -> Y in R specifies a constraint on all relation instances r(R)

Written as X -> Y; can be displayed graphically on a relation schema as in Figures. (denoted by the arrow: ).

FDs are derived from the real-world constraints on the attributes

Examples in EMP_PROJ relation:
SSN --> Ename
Pnumber --> Pname
{SSN,Pnumber} --> Hours

Candidate keys, by definition, functionally determine all other attributes of the relation.
Armstrong’s axioms

X, Y, Z and W are sets of attributes

1. If $Y \subseteq X$, then $X \rightarrow Y$ (reflexive rule)
2. If $X \rightarrow Y$, then $XZ \rightarrow YZ$ (augmentation rule)
3. If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$ (transitive rule)

Additional rules:
4. If $X \rightarrow YZ$, then $X \rightarrow Y$ (decomposition, or projection, rule)
5. If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$ (union, or additive, rule)
6. If $X \rightarrow Y$ and $WY \rightarrow Z$, then $WX \rightarrow Z$ (pseudotransitive rule)
Full Functional Dependency (FFD)

- A functional dependency \( X \Rightarrow Y \) is termed FFD if there is no attribute \( A \in X \) such that \( (X - \{A\}) \Rightarrow Y \) holds.
- In other words, a FFD is a dependency that do not contain any unnecessary attributes in its determinant (i.e. the left-hand side of the dependency).

\[
SUPP\_INFO(SNAME, INAME, SADDR, PRICE)
\]

\[
\begin{align*}
\{SNAME,INAME\} & \rightarrow \{PRICE\} \quad \ldots \quad (FD/FFD?) \quad FFD \\
\{SNAME,INAME\} & \rightarrow \{SADDR\} \quad \ldots \quad (FD/FFD?) \quad FD \\
\{SNAME\} & \rightarrow \{SADDR\} \quad \ldots \quad (FD/FFD?) \quad FFD
\end{align*}
\]
Prime attribute

Definition: an attribute that is a member in any of the candidate keys is called a prime attribute of $R$.

Attributes that is not part of any candidate key is called non-prime or non-key attributes.

Prime attributes:
- SSN
- Pnumber

non-prime attributes:
- Hours
- Ename
- Pname
Normalization

Breaking up tables to smaller ones to improve relational database design

Normal forms:

- Set of Criteria expressed in terms of FFDs/keys

First normal form - 1NF

Only atomic values are allowed as attribute values in the relational model.

Second normal form - 2NF

Following are not allowed:
- composite attributes
- multivalued attributes
- nested relations; attributes whose values for an individual tuple are non-atomic

Ideally, no FDs should be lost during normalization

Conditions get more strict as the level of normalization increases

Most strict  
Least strict

BCNF <- 3NF <- 2NF <- 1NF
**First normal form - 1NF**

Only atomic values are allowed as attribute values in the relational model.

**Second normal form - 2NF**

A relation schema $R$ is in 2NF if:
- $R$ is in 1NF.
- No partial dependencies are allowed, that is, a part of a candidate key shall not determine any non-prime attribute in $R$. 

Here SSN-Name reveals a partial dependency, which violates 2NF. What is the other partial dependency?

**Ideally, no FDs should be lost during normalization**

Following are not allowed:
- composite attributes
- multivalued attributes
- nested relations; attributes whose values for an individual tuple are non-atomic
### Figure 14.9
Normalization into 1NF.
(a) A relation schema that is not in 1NF. (b) Example state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

#### (a)
- **DEPARTMENT**
  - | Dname | Dnumber | Dmgr_ssn | Dlocations |
  - |-------|---------|----------|------------|
  - | Research | 5 | 333445555 | {Bellaire, Sugarland, Houston} |
  - | Administration | 4 | 987654321 | {Stafford} |
  - | Headquarters | 1 | 888665555 | {Houston} |

#### (b)
- **DEPARTMENT**
  - | Dname | Dnumber | Dmgr_ssn | Dlocations |
  - |-------|---------|----------|------------|
  - | Research | 5 | 333445555 | {Bellaire, Sugarland, Houston} |
  - | Administration | 4 | 987654321 | {Stafford} |
  - | Headquarters | 1 | 888665555 | {Houston} |

#### (c)
- **DEPARTMENT**
  - | Dname | Dnumber | Dmgr_ssn | Dlocation |
  - |-------|---------|----------|-----------|
  - | Research | 5 | 333445555 | Bellaire |
  - | Research | 5 | 333445555 | Sugarland |
  - | Research | 5 | 333445555 | Houston |
  - | Administration | 4 | 987654321 | Stafford |
  - | Headquarters | 1 | 888665555 | Houston |
<table>
<thead>
<tr>
<th>Ssn</th>
<th>Ename</th>
<th>Pnumber</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>Smith, John B.</td>
<td>1</td>
<td>32.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>7.5</td>
</tr>
<tr>
<td>666884444</td>
<td>Narayan, Ramesh K.</td>
<td>3</td>
<td>40.0</td>
</tr>
<tr>
<td>453453453</td>
<td>English, Joyce A.</td>
<td>1</td>
<td>20.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>20.0</td>
</tr>
<tr>
<td>333445555</td>
<td>Wong, Franklin T.</td>
<td>2</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>10.0</td>
</tr>
<tr>
<td>999887777</td>
<td>Zelaya, AliciaJ.</td>
<td>30</td>
<td>30.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>10.0</td>
</tr>
<tr>
<td>987987987</td>
<td>Jabbar, Ahmad V.</td>
<td>10</td>
<td>35.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>5.0</td>
</tr>
<tr>
<td>987654321</td>
<td>Wallace, Jennifer S.</td>
<td>30</td>
<td>20.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>15.0</td>
</tr>
<tr>
<td>888665555</td>
<td>Borg, James E.</td>
<td>20</td>
<td>NULL</td>
</tr>
</tbody>
</table>

Figure 14.10

Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a nested relation attribute PROJS. (b) Example extension of the EMP_PROJ relation showing nested relations within each tuple.
Second normal form - 2NF

A relation schema \( R \) is in 2NF if:

- It is in 1NF
- No partial dependencies are allowed, that is, a part of a candidate key shall not determine any none-prime attribute in \( R \).

Here SSN\(\rightarrow\)Ename reveals a partial dependency, which violates 2NF.
What is the other partial dependency?
Figure 14.11
Normalizing into 2NF and 3NF.
(a) Normalizing EMP_PROJ into 2NF relations.

Other explanations? why this table has to be broken like this?

Here SSN--->Ename reveals a partial dependency, which violates 2NF.

What is the other partial dependency?
Third normal form - 3NF

A relation schema \( R \) is in 3NF if:

- It is in 2NF
- No non-prime attribute \( A \) in \( R \) is allowed to be FFD of any other non-prime attribute.
- That is, Transitive (indirect) functional dependencies shall not exist:
  - if \( X \rightarrow Y \) and \( Y \rightarrow Z \) hold, \( X \rightarrow Z \) also holds (transitivity)
  - \( Z \) is indirectly determined by \( X \)
- \([\text{here } Y \text{ and } Z \text{ are none-prime attributes, and } X \text{ is a candidate key}]

(b) Normalizing EMP DEPT into 3NF relations.

(a)

EMP_PROJ

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Pn</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FD1

FD2

FD3

2NF Normalization

Here we have \( \text{SSN} \rightarrow \text{Dnumber} \) and \( \text{Dnumber} \rightarrow \{ \text{Dname,DMGR_ssn} \} \)
- So there is a transitional functional dependency:
  - \( \text{SSN} \rightarrow \{ \text{Dname,DMGR_ssn} \} \)
  - which violates 3NF criteria.
not exist:

- if $X \rightarrow Y$ and $Y \rightarrow Z$ hold, $X \rightarrow Z$ also holds (transitivity)
  - $Z$ is indirectly determined by $X$
- [here $Y$ and $Z$ are non-prime attributes, and $X$ is a candidate key]

**Figure 14.11**
(b) Normalizing EMP_DEPT into 3NF relations.

Here we have SSN $\rightarrow$ Dnumber and Dnumber $\rightarrow$ {Dname, DMGR_ssn}

- So there is a transitional functional dependency:
  - SSN $\rightarrow$ {Dname, DMGR_ssn}
- which violates 3NF criteria.
Boyce-Codd normal form - BCNF

A relation schema $R$ is in BCNF if:

- It is in 1NF
- Every determinant $X$ is a candidate key.
- The difference between BCNF and 3NF is that in BCNF, a prime attribute can not be FFD of a non-key (or non-prime) attribute or of a prime attribute (i.e. a partial key). Therefore BCNF is a more strict condition.

- BCNF is a strong condition. It is not always possible to transform (through decomposition) a schema to BCNF and keep the dependencies.
- 3NF has the most of BCNF's advantages and can still be fulfilled without giving up dependencies.
- In most practical cases, when a relation is in 3NF it is also in BCNF.
A tuple can not be FFD of a non-key (or non-prime) attribute (i.e. a partial key). Therefore BCNF is a more strict condition.

- BCNF is a strong condition. It is not always possible to transform (through decomposition) a schema to BCNF and keep the dependencies.
- 3NF has the most of BCNF’s advantages and can still be fulfilled without giving up dependencies.
- In most practical cases, when a relation is in 3NF it is also in BCNF.
Normalization
Breaking up tables to smaller ones to improve relational database design

Normal forms:
Are quality metrics:
- Set of Criteria expressed in terms of FFDs/keys

First normal form - 1NF
- Only atomic values are allowed as attribute values in the relational model.
- Following are not allowed:
  - composite attributes
  - multivalued attributes
  - nested relations; attributes whose values for an individual tuple are non-atomic

Ideally, no FDs should be lost during normalization

Conditions get more strict as the level of normalization increases

Most strict: BCNF
Least strict: 1NF
Summary

By normalizing relations, we avoid modification, insertion and deletion anomalies.

1 NF ................. Only atomic attribute values
2 NF ................. No partial dependencies allowed
3 NF ................. No transitive FDs allowed
BCNF ................. Left hand sides of FFDs should be candidate keys
Normalization

Elmasri/Navathe ch 14
Padron-McCarthy/Risch ch 11

DATABASE DESIGN I - IDL300 Spring 2013
Sobhan badiozamany
Department of Information Technology, Uppsala University

"Good" Relational database design

"Bad" Relational database design