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

DATABASE DESIGN I - 1DL300 Fall 2012
Sobhan badiozamany
Silvia Stefanova
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 relation schemas

What are the criteria for "good" tables?

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

<table>
<thead>
<tr>
<th>EMP_PROJ</th>
<th>Redundancy</th>
<th>Redundancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ssn</td>
<td>Pnumber</td>
<td>Hours</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>123456789</td>
<td>1</td>
<td>32.5</td>
</tr>
<tr>
<td>123456789</td>
<td>2</td>
<td>7.5</td>
</tr>
<tr>
<td>666884444</td>
<td>3</td>
<td>40.0</td>
</tr>
<tr>
<td>453453453</td>
<td>1</td>
<td>20.0</td>
</tr>
<tr>
<td>453453453</td>
<td>2</td>
<td>20.0</td>
</tr>
<tr>
<td>333445555</td>
<td>2</td>
<td>10.0</td>
</tr>
<tr>
<td>333445555</td>
<td>3</td>
<td>10.0</td>
</tr>
<tr>
<td>333445555</td>
<td>10</td>
<td>10.0</td>
</tr>
<tr>
<td>333445555</td>
<td>20</td>
<td>10.0</td>
</tr>
<tr>
<td>999887777</td>
<td>30</td>
<td>30.0</td>
</tr>
<tr>
<td>999887777</td>
<td>10</td>
<td>10.0</td>
</tr>
<tr>
<td>987987987</td>
<td>10</td>
<td>35.0</td>
</tr>
<tr>
<td>987987987</td>
<td>30</td>
<td>5.0</td>
</tr>
<tr>
<td>987654321</td>
<td>30</td>
<td>20.0</td>
</tr>
<tr>
<td>987654321</td>
<td>20</td>
<td>15.0</td>
</tr>
<tr>
<td>888665555</td>
<td>20</td>
<td>Null</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 a project specified by "Pnumber" to "NewBenefits" to "software.pjy".

This causes updating all records whose employees work on the project.

Insert Anomaly:

- Inserting an employee that is not assigned to a project is not possible.
- Defining a project to which no employee is assigned yet is not possible.

Delete Anomaly:

- If you want to delete a project, all employees working in that project are at risk of being deleted from the database.
- If only one employee works on a given project, deleting her would delete the project from the database.
**Update Anomaly:**
Assume we want to change the name of a project specified by "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 a project is not possible.
- Defining a project to which no employee is assigned yet is not possible.
Here is the Delete Anomaly:

- If you want to delete a project, all employees working in that project are at risk of being deleted from the database.
- If only one employee works on a given project, deleting her would delete the project from the database.

Note the compound key: \{SSN, Pnumber\}
In addition, redundant data causes logical design problems:

- **Modification anomalies**
- **Insertion anomalies**
- **Deletion anomalies**

Update Anomaly:
Assume we want to change the name of a project specified by "Pnumber" = 30 from "Newbenefits" to "software.py". 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.

Delete Anomaly:
- If you want to delete a project, all employees working in that project are at risk of being deleted from the database.
- If only one employee works on a given project, deleting her would delete the project from the database.
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.

**Figure 14.4**

<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>

What is a good design here?
Functional Dependencies

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

Prime attribute

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

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

Prime attributes:
- SSN
- Surname
- Firstname
- Lastname
- Number
- Score

Non-prime attributes:
- Roles
- Phone

Full Functional Dependency (FFD)

- A functional dependency X → Y is trivial if either (a) Y is a subset of X, or (b) Y = Y′.
- A functional dependency X → Y is non-trivial if X covers Y, i.e., X = X′.

Formally:
- X → Y
- X ∩ Y = ∅
- X ∪ Y = Y′

Examples:
- EMAIL → (NAME) (4/1/17) FID
- EMAIL → (NAME) (4/1/17) FID
- EMAIL → (NAME) (4/1/17) FID
- EMAIL → (NAME) (4/1/17) FID
- EMAIL → (NAME) (4/1/17) FID
- EMAIL → (NAME) (4/1/17) FID

Prime attribute

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

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

Prime attributes:
- SSN
- Surname
- Firstname
- Lastname
- Number
- Score

Non-prime attributes:
- Roles
- Phone

Full Functional Dependency (FFD)

- A functional dependency X → Y is trivial if either (a) Y is a subset of X, or (b) Y = Y′.
- A functional dependency X → Y is non-trivial if X covers Y, i.e., X = X′.

Formally:
- X → Y
- X ∩ Y = ∅
- X ∪ Y = Y′

Examples:
- EMAIL → (NAME) (4/1/17) FID
- EMAIL → (NAME) (4/1/17) FID
- EMAIL → (NAME) (4/1/17) FID
- EMAIL → (NAME) (4/1/17) FID
- EMAIL → (NAME) (4/1/17) FID
- EMAIL → (NAME) (4/1/17) FID

Prime attribute

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

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

Prime attributes:
- SSN
- Surname
- Firstname
- Lastname
- Number
- Score

Non-prime attributes:
- Roles
- Phone

Full Functional Dependency (FFD)

- A functional dependency X → Y is trivial if either (a) Y is a subset of X, or (b) Y = Y′.
- A functional dependency X → Y is non-trivial if X covers Y, i.e., X = X′.

Formally:
- X → Y
- X ∩ Y = ∅
- X ∪ Y = Y′

Examples:
- EMAIL → (NAME) (4/1/17) FID
- EMAIL → (NAME) (4/1/17) FID
- EMAIL → (NAME) (4/1/17) FID
- EMAIL → (NAME) (4/1/17) FID
- EMAIL → (NAME) (4/1/17) FID
- EMAIL → (NAME) (4/1/17) FID
Functional dependency

- formally specify how good a relational design is.
- Fds are constraints deduced from the miniworld 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 --> Y
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

![Figure 14.3](b)

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

Any key attribute by definition functionally determines all other attributes of the relation.
Armstrong’s axioms

X, Y, Z and W are sets of attributes

1. If Y ⊆ X, then X → Y (reflexive rule)
2. If X → Y, then XZ → YZ (augmentation rule)
3. If X → Y and Y → Z, then X → Z (transitive rule)

Additional rules:
4. If X → YZ, then X → Y (decomposition, or projection, rule)
5. If X → Y and X → Z, then X → YZ (union, or additive, rule)
6. If X → Y and WY → Z, then WX → Z (pseudo-transitive rule)
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)**

\[
\begin{align*}
\{\text{SNAME,INAME}\} & \rightarrow \{\text{PRICE}\} \quad \text{(FD/FFD?)} \quad \text{FFD} \\
\{\text{SNAME,INAME}\} & \rightarrow \{\text{SADDR}\} \quad \text{(FD/FFD?)} \quad \text{FD} \\
\{\text{SNAME}\} & \rightarrow \{\text{SADDR}\} \quad \text{(FD/FFD?)} \quad \text{FFD}
\end{align*}
\]
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:
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.

Second normal form - 2NF
Following are not allowed:
• composite attributes
• multivalued attributes
• nested relations; attributes whose values for an individual tuple are non-atomic

Conditions get more strict as the level of normalization increases:
• 1NF << 2NF << 3NF << BCNF
First normal form - 1NF

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

Second normal form - 2NF

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

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

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.
How should we normalize this?
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 key shall not determine any attribute in \( R \).

Here SSN\( \rightarrow \)Ename reveals a partial dependency, which violates 2NF.
Other explanations? why this table has to be broken like this?

Here SSN-->Ename reveals a partial dependency, which violates 2NF.
Third normal form - 3NF

A relation schema R is in 3NF if:
- It is in 2NF
- No non-key attribute A in R is allowed to be FFD of any other non-key attribute.
- That is, Transitive (indirect) functional dependencies shall not exist:
  - if X → Y and Y → Z hold, X → Z also holds (transitivity)
  - BUT, Z is indirectly determined by X

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

- if $X \rightarrow Y$ and $Y \rightarrow Z$ hold, $X \rightarrow Z$ also holds (transitivity)
- **BUT**, $Z$ is indirectly determined by $X$

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 cannot 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.
Attribute can not be FFD of a non-key (or non-prime) attribute (i.e. a partial key). Therefore BCNF is a more stringent 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.

Second normal form - 2NF
Following are not allowed:
• composite attributes
• multivalued attributes
• nested relations; attributes whose values for an individual tuple are non-atomic

Conditions get more strict as the level of normalization increases:
• 1 NF << 2NF << 3NF << BCNF
Summary

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 - 1DL300 Fall 2012
Sobhan Badiozamany
Silvia Stefanova
Department of Information Technology, Uppsala University

"Good" Relational database design

"Bad" Relational database design