Normalization in Database Design

Marek Rychly
mrychly@strathmore.edu

Strathmore University,
@iLabAfrica

&

Brno University of Technology,
Faculty of Information Technology

Advanced Databases and Enterprise Systems
14 October 2015
Outline

1. Normalization
   - Redundancy Anomalies
   - Functional Dependency
   - Normal Forms and Process of Normalization

2. An Example on Normalization
   - Normalization into 1NF
   - Normalization into 2NF and 3NF
   - Normalization into BCNF
Normalization is a technique for producing a set of relations with desirable properties, given the data requirements of an enterprise.

- Db. design should create an accurate representation of the data, their relationships and constraints, pertinent to the enterprise.
- Normalization begins by examining the relationships (called functional dependencies, FD) between attributes.
- A series of tests (known as normal forms, NF) help to identify the optimal grouping for these attributes into a set of suitable relations.
Suitable Set of Relations

- The suitable set of relations that support the data should contain:
  - the minimal number of (necessary) attributes
  - attributes with a close logical relationship (FD) in the same relation
  - minimal redundancy with each attribute represented only once with the important exception of attributes that form all or part of foreign keys, which are essential for the joining of related relations

- The such relations are easier to access and maintain because:
  - updates are achieved with a minimal number of operations (reducing opportunities for data inconsistencies occurring in db.)
  - data take up a minimal storage space (minimizing costs on the data storage)

- The suitable set of relations can be found by two approaches:
  - bottom-up by creating the set of relations from a list of attrs. by normalization
  - top-down by verifying an ER model and resulting relations on normal forms
Normalization and Redundancy Anomalies

Normalization prevents destructive anomalies caused by data redundancy.
(by minimizing a set of attributes and enforcing the proper use of foreign keys)

- **Insertion anomaly**
  is the inability to insert data into db. due to
  - absence of other data
    (the data are missing but (redundant) attributes need to be filled)
  - necessity to insert also redundant data that must be consistent with their other occurrences in the other relations
    (the data for attributes that should not be there because the same data are already stored somewhere else)

- **Deletion anomaly**
  is the accidental loss of data due to deletion of other data
  (the data were stored in wrong relations together with (the right) data to delete)

- **Update anomaly**
  is the necessity to update the redundant data in all their occurrences
  (otherwise, the update will not be done properly and there will be old and new data)
An Example of Insertion Anomaly

Faculty and Their Courses

<table>
<thead>
<tr>
<th>Faculty ID</th>
<th>Faculty Name</th>
<th>Faculty Hire Date</th>
<th>Course Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>389</td>
<td>Dr. Giddens</td>
<td>10-Feb-1985</td>
<td>ENG-206</td>
</tr>
<tr>
<td>407</td>
<td>Dr. Saperstein</td>
<td>19-Apr-1999</td>
<td>CMP-101</td>
</tr>
<tr>
<td>407</td>
<td>Dr. Saperstein</td>
<td>19-Apr-1999</td>
<td>CMP-201</td>
</tr>
<tr>
<td>424</td>
<td>Dr. Newsome</td>
<td>29-Mar-2007</td>
<td></td>
</tr>
</tbody>
</table>

(adopted from “Database normalization. Wikipedia.”)

- The new data can not be inserted as the Course Code is missing.
  (e.g., the new faculty member has not yet been assigned any courses)
- In this insert, the Course Code has to be set to Null value.
- **Solution:** Move the Course Code into a separate table and link it with the original table by Faculty ID (the FK–PK ref. integrity).
An Example of Deletion Anomaly

Faculty and Their Courses

<table>
<thead>
<tr>
<th>Faculty ID</th>
<th>Faculty Name</th>
<th>Faculty Hire Date</th>
<th>Course Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>389</td>
<td>Dr. Giddens</td>
<td>10-Feb-1985</td>
<td>ENG-206</td>
</tr>
<tr>
<td>407</td>
<td>Dr. Saperstein</td>
<td>19-Apr-1999</td>
<td>CMP-101</td>
</tr>
<tr>
<td>407</td>
<td>Dr. Saperstein</td>
<td>19-Apr-1999</td>
<td>CMP-201</td>
</tr>
</tbody>
</table>

(adopted from “Database normalization. Wikipedia.”)

- The fact that the faculty member is teaching the course can not be deleted as the action will result into the deletion of the member.
- In this delete the Course Code has to be set to Null value.
- **Solution:** Move the Course Code into a separate table and link it with the original table by Faculty ID (the FK–PK ref. integrity).
Normalization

An Example on Normalization

Redundancy Anomalies
Functional Dependency
Normal Forms and Process of Normalization

An Example of Update Anomaly

Employees' Skills

<table>
<thead>
<tr>
<th>Employee ID</th>
<th>Employee Address</th>
<th>Skill</th>
</tr>
</thead>
<tbody>
<tr>
<td>426</td>
<td>87 Sycamore Grove</td>
<td>Typing</td>
</tr>
<tr>
<td>426</td>
<td>87 Sycamore Grove</td>
<td>Shorthand</td>
</tr>
<tr>
<td>519</td>
<td>94 Chestnut Street</td>
<td>Public Speaking</td>
</tr>
<tr>
<td>519</td>
<td>96 Walnut Avenue</td>
<td>Carpentry</td>
</tr>
</tbody>
</table>

(adopted from “Database normalization. Wikipedia.”)

- Due to the redundancy, a (partial) update of the employee’s address resulted into inconsistent data. (i.e., not all occurrences of the redundant data have been updated)
- All occurrences of the redundant data have to be updated.
- **Solution:** Move the Employee Address into a separate table and link it with the original table by Empl. ID (the FK–PK ref. integrity).
Removing the Redundancy

- The redundancy has to be removed to eliminate the anomalies.

- It can be removed by splitting relations into smaller ones.
  (the smaller relations linked together by the referential integrity)

- However, the redundancy removal must not result into lost data.
  (it must not weaken the informational value of the database)

- More specifically, the following properties must hold:
  
  (lossless-join) any tuple from the original relation can be found also in a natural join
  of the smaller relations (so no tuples were lost by the splitting)

  (dependency preservation) any constraint on the original relation can be maintained by simply
  enforcing some constraint on each of the smaller relations (no need
  for joins on the smaller relations to check the constraint)

- The normalization keeps the informational value of the database.
  (both, the lossless-join and the dependency preservation)
Functional Dependency

Definition (Functional dependency, FD)

FD describes a permanent relationship between attributes in a relation. If $A$ and $B$ are attributes (or sets of attributes) of relation $R$, $B$ is FD on $A$ (also $A$ functionally determines $B$; both denoted $A \rightarrow B$), if each value (or tuple) of $A$ is associated with exactly one value (or tuple) of $B$.

Definition (Determinant)

Determinant refers to the attribute, or a set of attributes, on the left-hand side of the arrow of a FD.

- It is the “functional” dependency, so one value of determinant cannot result into two different values in determined attribute (if so, there is no such FD between given attributes in the given direction).

- If $A \rightarrow B$ then there is 1 : 1 relationship between $A$ and $B$. (in this direction; however, it can be 1 : 1 or 1 : $N$ in the opposite direction).
An Example of Functional Dependencies

Faculty and Their Courses

<table>
<thead>
<tr>
<th>Faculty ID</th>
<th>Faculty Name</th>
<th>Faculty Hire Date</th>
<th>Course Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>389</td>
<td>Dr. Giddens</td>
<td>10-Feb-1985</td>
<td>ENG-206</td>
</tr>
<tr>
<td>407</td>
<td>Dr. Saperstein</td>
<td>19-Apr-1999</td>
<td>CMP-101</td>
</tr>
<tr>
<td>407</td>
<td>Dr. Saperstein</td>
<td>19-Apr-1999</td>
<td>CMP-201</td>
</tr>
</tbody>
</table>

(adopted from “Database normalization. Wikipedia.”)

Functional dependencies:

1. FacultyID → FacultyName
2. FacultyID → FacultyHireDate
3. CourseCode → FacultyID

(1) and (2) result into FacultyID → \{FacultyName, FacultyHireDate\}

(FacultyID is a candidate key, and also a primary key, because there is just one CK)
Full/Partial Functional Dependency

Definition (Full/Partial functional dependency)

Let $A$ and $B$ be attributes of a relation. $B$ is **fully functionally dependent** on $A$ if $B$ is functionally dependent on $A$, but not on any proper subset of $A$ (i.e., determinant $A$ is the minimal possible); otherwise it is **partially functionally dependent**.

- By identification of FDs, integrity constraints can be identified.
- For example, if $A$ is a such attr. (or set of attrs.) that each attr. of the same relation is fully func. dep. on $A$, then $A$ is a CK. (and if it is the only one CK, it is also a primary key)
- Identifying all FDs between attrs. is simple if the meaning of each attr. and the relationships between the attrs. are well understood.
  1. identify the deps. determined from the semantics of the attrs.
  2. apply the inference rules to infer additional FDs for the relations.
Definition (Closure of set of functional dependencies)

The set of all functional dependencies that are implied by a given set of functional dependencies $X$ is called the closure of $X$, written $X^+$. All FDs implied by $X$ ($X^+$) are derived from $X$ by the inference rules:

- if $B$ is a subset of $A$, then $A \rightarrow B$ (reflexivity)
- if $A \rightarrow B$, then $A, C \rightarrow B, C$ for any $C$ (augmentation)
- if $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$ (transitivity)

Other rules can be derived from the inference rules:

- $A \rightarrow A$ (self-determination)
- $A \rightarrow B, C$, then $A \rightarrow B$ and $A \rightarrow C$ (decomposition)
- $A \rightarrow B$ and $A \rightarrow C$, then $A \rightarrow B, C$ (union)
- $A \rightarrow B$ and $C \rightarrow D$ then $A, C \rightarrow B, D$ (composition)
Equivalent and Minimal Set of FDs

Definition (Equivalence of two set of FDs)

Two sets of FDs $F$ and $G$ are equivalent, written $F \equiv G$, if $F^+ = G^+$.  

Definition (Minimal set of FDs)

A set of FDs $X$ is minimal if it satisfies the following conditions:

- Every FD has a single attribute on its right-hand side.
- Every FD is a full functional dependency.
- No FD can be removed and still have an equivalent set of FDs.

We are looking for the minimal sets of FDs, others are redundant.
Normalization and Normal Forms

Normalization is a formal technique for analysing relations.
(testing them by a series of rules known as normal forms, NFs; by Codd, 1972)
- based on their candidate keys (or a primary key)
- based on functional dependencies

Normalization is often executed as a series of steps.
(each step corresponds to a specific NF that has known properties)

As norm. proceeds, the relations become progressively more
restricted (stronger) and also less vulnerable to update anomalies.

There are several NFs; important are 1\textsuperscript{st} NF, 3\textsuperscript{rd} NF, and BCNF.
(un-normalized form/UNF, 1NF, 2NF, 3NF, BCNF/3.5NF, 4NF, 5NF, . . .)

1NF ensures valid relations according to the relational data model.
(only 1NF is critical in creating relations; all subsequent normal forms are optional)

However, it is recommended to proceed to 3NF or BCNF.
(to avoid the redundancy anomalies discussed at page 6)
Definition (Un-normalized normal form, UNF)
A relation/table that contains one or more repeating groups is in UNF.

Definition (The first normal form, 1NF)
A rel. where each attr. domain contains only atomic values is in 1NF.

- **1NF** has the same requirements as the relational model.
  (i.e., all attributes can have only atomic values)

- The atomicity of attribute values is determined by its usage.
  (e.g., an address attribute is typically composed from street, city, etc., however, it
  can be still atomic if always used as the whole, not by its individual components)

- Trans. UNF to 1NF by decomposing composed/repeating values:
  - duplicating rows in such way that each copy will have just one value
    (data from columns of atomic domains will be just duplicated for each row)
  - or moving the problematic values decomposed into a new relation
    (and adding a copy of PK into the target relation to link with the orig. data)
The Second Normal Form (2NF)

**Definition (The second normal form, 2NF)**

A relation is in 2NF if it is in 1NF and every non-candidate-key attribute is fully functionally dependent on every candidate key (i.e., also on PK).

- FD on a part of any candidate key is a violation of 2NF. (2NF requires the full functional dependency)
- Trans. 1NF to 2NF by moving the partially dependent attribute(s) into a new relation along with a copy of their determinant\(^1\). (determinant ensures keeping a link to corresponding rows of the original relation)

---

\(^1\) determinant is attribute(s) on which the other attribute is fully functionally dependent
The Third Normal Form (3NF)

Definition (The third normal form, 3NF)
A relation is in 3NF if it is in 2NF and no non-candidate-key attribute is transitive dependent on any candidate key (i.e., also on PK).

- All attrs. determined only by CKs/PK and not by any non-key attrs. (the non-key attr. deps. result into transitive deps. as all attrs. depended on CKs)
- Trans. 2NF to 3NF by moving the transitively dependent attribute(s) into a new relation along with a copy of determinant. (determinant ensures keeping a link to corresponding rows of the original relation)
Definition (The Boyce-Codd normal form, BCNF/3.5NF)

A relation is in BCNF if and only if every determinant is a candidate key.

- Relations in BCNF are also in all the previous NFs (mainly in 3NF).
- In BCNF relations, there is no redundancy based on FD.
  (other types of redundancy may still exist, they are subjects of higher normal forms)
- Unlike the previous NFs, BCNF is not always achievable.
  (e.g., a set of FDs \{A, B \rightarrow C, C \rightarrow B\} cannot be represented by a BCNF schema)
- Trans. to BCNF by moving attribute(s) that do not depend on CKs into a new relation along with a copy of their determinant.
  (determinant ensures keeping a link to corresponding rows of the original relation)
**An Example: Property Inspection Database (PIDb)**

**DreamHome**
Property Inspection Report

<table>
<thead>
<tr>
<th>Property Number</th>
<th>PG4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Address</td>
<td>6 Lawrence St, Glasgow</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inspection Date</th>
<th>Inspection Time</th>
<th>Comments</th>
<th>Staff no</th>
<th>Staff Name</th>
<th>Car Registration</th>
</tr>
</thead>
<tbody>
<tr>
<td>18-Oct-03</td>
<td>10.00</td>
<td>Need to replace crockery</td>
<td>SG37</td>
<td>Ann Beech</td>
<td>M231 JGR</td>
</tr>
<tr>
<td>22-Apr-04</td>
<td>09.00</td>
<td>In good order</td>
<td>SG14</td>
<td>David Ford</td>
<td>M533 HDR</td>
</tr>
<tr>
<td>1-Oct-04</td>
<td>12.00</td>
<td>Damp rot in bathroom</td>
<td>SG14</td>
<td>David Ford</td>
<td>N721 HFR</td>
</tr>
</tbody>
</table>

The table above stores Property Inspection reports PG4 & PG16. (i.e., two rows in the table, or two tuples in the relation represented by the table)

PG4 (or PG16) report contains descriptions of 3 (or 2) inspections. (each inspection records: date, time, comment, staff no. and name, and car reg.)
The inspection record attributes contained non-atomic values. (there were groups of several values in individual cells of the table)

Therefore, to remove groups of data in the cells, each property report has been duplicated and assigned to its inspection records. (then, domains of attrs. of the resulting relation are atomic and the rel. is in 1NF)

If we were accessing individual parts of “pAddress” attribute, we would need also to split this attr. into “street” and city attributes to get 1NF.
### PIDb – StaffPropertyInspection Relation Func. Deps.

(Identifying individual functional dependencies)

<table>
<thead>
<tr>
<th>propertyNo</th>
<th>iDate</th>
<th>iTime</th>
<th>pAddress</th>
<th>comments</th>
<th>staffNo</th>
<th>sName</th>
<th>carReg</th>
</tr>
</thead>
</table>

- **fd1**
  - (Primary key)

- **fd2**
  - (Partial dependency)

- **fd3**
  - (Transitive dependency)

- **fd4**

- **fd5**
  - (Candidate key)

- **fd6**
  - (Candidate key)

PI Db – StaffPropertyInspection Relation Func. Deps. (writing down a (minimal) set of the functional dependencies)

fd1/PK: \{propertyNo, iDate\} \rightarrow \{ iTime, comments, staffNo, carReg\}  
(there is a particular inspection (its details) for a given property and date)

fd2: \{propertyNo\} \rightarrow \{ pAddress\}  
(there is a particular address for a given property describing its address)

fd3: \{staffNo\} \rightarrow \{ sName\}  
(there is a particular name for a given staff number describing his/her full name)

fd4: \{iDate, staffNo\} \rightarrow \{ carReg\}  
(there is a particular car assigned to a given staff for a given day)

fd5/CK: \{iDate, iTime, carReg\} \rightarrow \{propertyNo, comments, staffNo\}  
(there is a particular inspection where a given car was at a given date and time)

fd6/CK: \{iDate, iTime, staffNo\} \rightarrow \{propertyNo, comments, carReg\}  
(there is a particular inspection where a given staff was at a given date and time)

Note: Several dependencies were hidden as they can be derived by the inference rules.
Definition (The second normal form, 2NF)

A relation is in 2NF if it is in 1NF and every non-candidate-key attribute is fully functionally dependent on every candidate key (i.e., also on PK).

- Candidate keys are:
  \{propertyNo, iDate\}, \{iDate, iTime, carReg\}, \{iDate, iTime, staffNo\}
  (moreover, the first set above is also a primary key)

- pAddress is partially functionally dependent of the first CK above.
  \(fd2: \{propertyNo\} \rightarrow \{pAddress\}\)

- pAddress, with a copy of propertyNo, moved to a new relation.

- The resulting relations, with primary keys underlined, are
  Property(propertyNo, pAddress)
  PropertyInspection(propertyNo, iDate, iTime, comments, staffNo, sName, carReg)
Definition (The third normal form, 3NF)

A relation is in 3NF if it is in 2NF and no non-candidate-key attribute is transitively dependent on any candidate key (i.e., also on PK).

- \( sName \) is transitively functionally dependent of the first CK due to
  - \( \text{fd1/PK: } \{\text{propertyNo, iDate}\} \rightarrow \{\text{iTime, comments, staffNo, carReg}\} \)
  - \( \text{fd3: } \{\text{staffNo}\} \rightarrow \{\text{sName}\} \)

- \( sName \), with a copy of \( \text{staffNo} \), moved to a new relation.

The resulting relations, with primary keys underlined, are

\[
\begin{align*}
\text{Property(} & \text{propertyNo, pAddress)} \\
\text{Staff(} & \text{staffNo, sName)} \\
\text{PropertyInspection(} & \text{propertyNo, iDate, iTIme, comments, staffNo, carReg)}
\end{align*}
\]
Definition (The Boyce-Codd normal form, BCNF/3.5NF)

A relation is in BCNF if and only if every determinant is a candidate key.

- Candidate keys are:
  - \{propertyNo\}; \{staffNo\};
  - \{propertyNo, iDate\}, \{iDate, iTime, carReg\}, \{iDate, iTime, staffNo\}
  (the first two sets are CKs&PKs of new relations Property and Staff, respectively)
- \{iDate, staffNo\} is a determinant for fd4 and it is not CK.
  - \text{fd4: } \{iDate, staffNo\} \rightarrow \{carReg\}
- carReg, with a copy of iDate and staffNo, moved to a new relation.
- The resulting relations, with primary keys underlined, are
  - \text{Property}(propertyNo, pAddress)
  - \text{Staff}(staffNo, sName)
  - \text{StaffCar}(iDate, staffNo, carReg)
  - \text{Inspection}(propertyNo, iDate, iTime, comments, staffNo)
Normalization in Database Design — ADES, 14 October 2015

Marek Rychly

Normalization
An Example on Normalization

Normalization into 1NF
Normalization into 2NF and 3NF
Normalization into BCNF

PIDb – Decomposition of StaffPropertyInspection Rel.

Normalization into 1NF
Normalization into 2NF and 3NF
Normalization into BCNF

StaffPropertyInspection

PropertyInspection

PropertyInspect

Staff
StaffCar
Inspection
Property


Marek Rychly

Normalization in Database Design — ADES, 14 October 2015

30 / 33
(5 points) For each relation from the previous assessments, identify functional dependencies of (sets of) their attributes.
(you need not to write down all dependencies, just those of them that can be used with the inference rules to derive the all dependencies)

(5 points) Check, or modify, the relations to meet 1NF, 2NF, 3NF, and BCNF.

(5 points) Describe resulting relations as an ER model in UML notation.
(can be drawn by hand or in a modelling tool, e.g., in Visual Paradigm or Modelio)

The result should be submitted by email to
Marek Rychly <mrychly@strathmore.edu>
by 28 October 2015 at the latest.
Data redundancy may cause destructive anomalies. (insert, delete, and update anomalies resulting into wrong or missing data)

Functional dependency (FD) describes a relationship between attributes. (between attrs. of the same relation where a determinant determines dependent)

1NF ensures valid relations according to the relational data model.

The normalization proceeds with 2NF, 3NF, and BCNF. (however, BCNF may not be achievable for some relations and FDs)

In BCNF relations, there is no redundancy based on FD. (it means also no destructive anomalies caused by the redundancy)

In the next lecture:

Database Performance
(query processing, indices, explain plan & SQL hints, optimizing client access, etc.)
Thank you for your attention!

Marek Rychly
<mrychly@strathmore.edu>

Note: Definitions and some of examples in this presentation have been adopted from “Connolly & Begg: Database Systems, 2005, p. 387–428. ISBN 0-321-21025-5”.