Database Normalization

Todd Bacastow
IST 210
Overview

- Introduction
- The Normal Forms
- Relationships and Referential Integrity
- Real World Exercise
Keys in the relational model

- **Superkey**
  - A set of one or more attributes, which, taken collectively, allow us to identify uniquely a tuple in a relation.

- **Candidate key (or Key)**
  - A superkey for which no proper subset is a superkey.

- **Primary key**
  - The candidate key that is chosen by the database designer as the principle key.
Terminology

- Use **capital letters** to denote attributes
  - A, B, C, D, ...

- Use **strings** to denote sets of attributes
  - ABCD represents \{A, B, C, D\}

- Use **underlined capital letters** to represent sets of attributes
  - A, B, C, ...

- A string of set symbols represent the union of the attributes in each set:
  - ABC = A \cup B \cup C
Two important goals of decomposition

- **Lossless-join property**
  - enables us to find any instance of the original relation from corresponding instances in the smaller relations.

- **Dependency preservation property**
  - enables us to enforce a constraint on the original relation by enforcing some constraint on each of the smaller relations.
More on Lossless Join

- This example is **not** a lossless decomposition
Anomalies

- A bad database design may suffer from anomalies that make the database difficult to use:

COMPANIES(company_name, company_address, date_founded, owner_id, owner_name, owner_title, #shares)

Suppose Primary Key (company_name, owner_id)

- Anomalies:
  - **update anomaly** occurs if changing the value of an attribute leads to an inconsistent database state.
  - **insertion anomaly** occurs if we cannot insert a tuple due to some design flaw.
  - **deletion anomaly** occurs if deleting a tuple results in unexpected loss of information.

- Normalization is the systematic process for removing all such anomalies in database design.
Update Anomaly

- If a company has three owners, there are three tuples in the COMPANIES relation for this company.

- If this company moves to a new location, the company’s address must be updated consistently in all three tuples.
  - updating the company address in just one or two of the tuples creates an inconsistent database state.

- It would be better if the company name and address were in a separate relation so that the address of each company appears in only one tuple.

COMPANIES(company_name, company_address, date_founded, owner_id, owner_name, owner_title, #shares)
Insert Anomaly

- Suppose that three people have just created a new company:
  - the three founders have no titles yet
  - stock distributions have yet to be defined
- The new company cannot be added to the COMPANIES relation because there is not enough information to fill in all the attributes of a tuple
  - at best, null values can be used to complete a tuple
- It would be better if owner and stock information was stored in a different relation

COMPANIES(company_name, company_address, date_founded, owner_id, owner_name, owner_title, #shares)
Delete Anomaly

- Suppose that an owner of a company retires so is no longer an owner but retains stock in the company.
- If this person’s tuple is deleted from the COMPANIES relation, then we lose the information about how much stock the person still owns.
- If the stock information was stored in a different relation, then we can retain this information after the person is deleted as an owner of the company.

COMPANIES (company_name, company_address, date_founded, owner_id, owner_name, owner_title, #shares)
A functional dependency is a constraint between two sets of attributes in a relational database.

If X and Y are two sets of attributes in the same relation R, then $X \rightarrow Y$ means that X functionally determines Y so that

- the values of the attributes in X uniquely determine the values of the attributes in Y
Functional Dependencies

What are the functional dependencies in:

COMPANIES(company_name, company_address, date_founded, owner_id, owner_name, owner_title, #shares)

company_name $\rightarrow$ company_address
company_name $\rightarrow$ date_founded
company_name, owner_id $\rightarrow$ owner_title
company_name, owner_id $\rightarrow$ #shares
company_name, owner_title $\rightarrow$ owner_id
owner_id $\rightarrow$ owner_name
Functional Dependency

- Main concept associated with normalization.
- Diagrammatic representation.

Diagram:

A → B

B is functionally dependent on A.
Armstrong’s Axioms

Armstrong’s Axioms: Let $X, Y$ be sets of attributes from a relation $T$.

- **Inclusion rule:** If $Y \subseteq X$, then $X \rightarrow Y$
- **Transitivity rule:** If $X \rightarrow Y$, and $Y \rightarrow Z$, then $X \rightarrow Z$.
- **Augmentation rule:** If $X \rightarrow Y$, then $XZ \rightarrow YZ$.

Other derived rules:

- **Union rule:** If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
- **Decomposition rule:** If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
- **Pseudotransitivity:** If $X \rightarrow Y$ and $WY \rightarrow Z$, then $XW \rightarrow Z$
- **Accumulation rule:** If $X \rightarrow YZ$ and $Z \rightarrow BW$, then $X \rightarrow YZB$

$\subseteq = \text{subset, } \rightarrow = \text{functionally dependent}$
Referential Integrity

- Every piece of “foreign” key data has a primary key on the one side of the relationship
  - No “orphan” records. *Every child has a parent*
  - Can’t delete records from primary table if in related table

- Benefits - Data Integrity and Propagation
  - If update fields in main table, reflected in all queries
  - Can’t add a record in related table without adding it to main
  - **Cascade Delete**: If delete record from primary table, all children deleted
  - **Cascade Update**: If change the primary key field, will change foreign key
Normalization Defined

- In relational database design,
  - the process of organizing data to minimize duplication.
  - **Normalization** usually involves dividing a database into two or more tables and defining relationships between the tables.
    - The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database via the defined relationships.
"Normalization" refers to the process of creating an efficient, reliable, flexible, and appropriate "relational" structure for storing information.
Why Normalize? In Summary ...

- **Flexibility**
  - Structure supports many ways to look at the data

- **Data Integrity – Prevent anomalies**
  - Deletion
  - Insertion
  - Update

- **Efficiency**
  - Eliminate redundant data and save space
The Normal Forms

- Two means
  - Inspection
  - Closure

- A series of logical steps to take to normalize data tables
  - First NF
  - Second NF
  - Third NF
- There’s more…but this is enough for now
Normal forms

- **Unnormalized Form (UNF)**: A table that contains one or more repeating groups.

- **First Normal Form (1NF)**: A relation in which the intersection of each row and column contains one and only one value.
First Normal Form (1NF)

- “Flattening” the table
- All columns (fields) must have no repeating items in columns

<table>
<thead>
<tr>
<th>OrderDate</th>
<th>Customer</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/30/1998</td>
<td>Joe Smith</td>
<td>Hammer, Saw, Nails</td>
</tr>
</tbody>
</table>

Solution: make a separate table for each set of attributes with a primary key (parser, append query)
Second Normal Form (2NF)

- In 2NF and every non-key column is fully dependent on the (entire) primary key
  - Means: Does the key field imply the rest of the fields?
  - Do we need to know both **OrderID** and **Item** to know the Customer and Date?

<table>
<thead>
<tr>
<th>OrderID</th>
<th>Item</th>
<th>CustomerID</th>
<th>OrderDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hammer</td>
<td>1</td>
<td>11/30/1998</td>
</tr>
<tr>
<td>1</td>
<td>Saw</td>
<td>1</td>
<td>11/30/1998</td>
</tr>
<tr>
<td>1</td>
<td>Nails</td>
<td>1</td>
<td>11/30/1998</td>
</tr>
</tbody>
</table>

Solution: Remove to a separate table (Make Table)
Third Normal Form (3NF)

- In 3NF, every non-key column is mutually independent
  - means: no transitive dependency like calculations

<table>
<thead>
<tr>
<th>Item</th>
<th>Quantity</th>
<th>Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hammer</td>
<td>2</td>
<td>$10</td>
<td>$20</td>
</tr>
<tr>
<td>Saw</td>
<td>5</td>
<td>$40</td>
<td>$200</td>
</tr>
<tr>
<td>Nails</td>
<td>8</td>
<td>$1</td>
<td>$8</td>
</tr>
</tbody>
</table>

- Solution: Put calculations in queries and forms
Transitive Dependency

Transitive Dependency is a condition where

A, B and C are attributes of a relation such that if A → B and B → C,

then C is transitively dependent on A through B. (Provided that A is not functionally dependent on B or C).
# DreamHome Example

<table>
<thead>
<tr>
<th>Property Number</th>
<th>Property Address</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG4</td>
<td>6 Lawrence St, Glasgow</td>
<td>1-Oct-98</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inspection Date</th>
<th>Inspection Time</th>
<th>Comments</th>
<th>Staff Number</th>
<th>Staff Name</th>
<th>Car Reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>18-Oct-96</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-97</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-98</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>
Example - Normalization
UNF to 1NF Relation

Property_Inspection Table

<table>
<thead>
<tr>
<th>Property_No</th>
<th>PAddress</th>
<th>IDate</th>
<th>ITime</th>
<th>Comments</th>
<th>Staff_No</th>
<th>SName</th>
<th>Car_Reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG4</td>
<td>6 Lawrence St, Glasgow</td>
<td>18-Oct-96</td>
<td>10.00</td>
<td>need to replace crockery in good order</td>
<td>SG37</td>
<td>Ann Beech</td>
<td>M231 JGR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22-Apr-97</td>
<td>09.00</td>
<td></td>
<td>SG14</td>
<td>David Ford</td>
<td>M533 HDR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-Oct-98</td>
<td>12.00</td>
<td>damp rot in bathroom</td>
<td>SG14</td>
<td>David Ford</td>
<td>N721 HFR</td>
</tr>
<tr>
<td>PG16</td>
<td>5 Novar Dr, Glasgow</td>
<td>22-Apr-96</td>
<td>13.00</td>
<td>replace living room carpet</td>
<td>SG14</td>
<td>David Ford</td>
<td>M533 HDR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24-Oct-97</td>
<td>14.00</td>
<td>good condition</td>
<td>SG37</td>
<td>Ann Beech</td>
<td>N721 HFR</td>
</tr>
</tbody>
</table>

Property_Inspection Relation

<table>
<thead>
<tr>
<th>Property_No</th>
<th>IDate</th>
<th>ITime</th>
<th>PAddress</th>
<th>Comments</th>
<th>Staff_No</th>
<th>SName</th>
<th>Car_Reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG4</td>
<td>18-Oct-96</td>
<td>10.00</td>
<td>6 Lawrence St, Glasgow</td>
<td>need to replace crockery in good order</td>
<td>SG37</td>
<td>Ann Beech</td>
<td>M231 JGR</td>
</tr>
<tr>
<td>PG4</td>
<td>22-Apr-97</td>
<td>09.00</td>
<td>6 Lawrence St, Glasgow</td>
<td></td>
<td>SG14</td>
<td>David Ford</td>
<td>M533 HDR</td>
</tr>
<tr>
<td>PG4</td>
<td>1-Oct-98</td>
<td>12.00</td>
<td>6 Lawrence St, Glasgow</td>
<td>damp rot in bathroom</td>
<td>SG14</td>
<td>David Ford</td>
<td>N721 HFR</td>
</tr>
<tr>
<td>PG16</td>
<td>22-Apr-96</td>
<td>13.00</td>
<td>5 Novar Dr, Glasgow</td>
<td>replace living room carpet</td>
<td>SG14</td>
<td>David Ford</td>
<td>M533 HDR</td>
</tr>
<tr>
<td>PG16</td>
<td>24-Oct-97</td>
<td>14.00</td>
<td>5 Novar Dr, Glasgow</td>
<td>good condition</td>
<td>SG37</td>
<td>Ann Beech</td>
<td>N721 HFR</td>
</tr>
</tbody>
</table>
Second Normal Form (2NF)

- A relation that is in 1NF, and
- Every non-primary-key attribute is functionally dependent only on the primary key, but not any subset of the primary key.
1NF to 2NF

- Identify the primary key for the 1NF relation.
- Identify the functional dependencies in the relation.
- If partial dependencies exist on the primary key remove them by placing them in a new relation.
# FDs for Customer_Rental Relation

## Primary Key

<table>
<thead>
<tr>
<th>Customer_No</th>
<th>Property_No</th>
<th>CName</th>
<th>PAddress</th>
<th>RentStart</th>
<th>RentFinish</th>
<th>Rent</th>
<th>Owner_No</th>
<th>OName</th>
</tr>
</thead>
</table>


FDs for Customer_Rental Relation

Primary Key

<table>
<thead>
<tr>
<th>Customer_No</th>
<th>Property_No</th>
<th>CName</th>
<th>PAddress</th>
<th>RentStart</th>
<th>RentFinish</th>
<th>Rent</th>
<th>Owner_No</th>
<th>OName</th>
</tr>
</thead>
</table>

fd1

(Primary key)
### FDs for Customer_Rental Relation

**Primary Key**

<table>
<thead>
<tr>
<th>Customer_No</th>
<th>Property_No</th>
<th>CName</th>
<th>PAddress</th>
<th>RentStart</th>
<th>RentFinish</th>
<th>Rent</th>
<th>Owner_No</th>
<th>OName</th>
</tr>
</thead>
</table>

- **fd1**
  - (Primary key)

- **fd2**
  - (Partial dependency)
FDs for Customer_Rental Relation

Primary Key

<table>
<thead>
<tr>
<th>Customer_No</th>
<th>Property_No</th>
<th>CName</th>
<th>PAddress</th>
<th>RentStart</th>
<th>RentFinish</th>
<th>Rent</th>
<th>Owner_No</th>
<th>OName</th>
</tr>
</thead>
</table>

- fd1: (Primary key)
- fd2: (Partial dependency)
- fd3: (Partial dependency)
FDs for Customer_Rental Relation

Primary Key

<table>
<thead>
<tr>
<th>Customer_No</th>
<th>Property_No</th>
<th>CName</th>
<th>PAddress</th>
<th>RentStart</th>
<th>RentFinish</th>
<th>Rent</th>
<th>Owner_No</th>
<th>OName</th>
</tr>
</thead>
</table>

- fd1: (Primary key)
- fd2: (Partial dependency)
- fd3: (Partial dependency)

Rental: (Customer_No, Property_No, RentStart, RentFinish)
Customer: (Customer_No, Cname)
Property_Owner: (Property_No, Paddress, Rent, Owner_No, Oname)
### Example - Normalization

**Customer_Rental to 2NF Relations**

<table>
<thead>
<tr>
<th>Customer_No</th>
<th>CName</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR76</td>
<td>John Kay</td>
</tr>
<tr>
<td>CR56</td>
<td>Aline Stewart</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rental Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer_No</td>
</tr>
<tr>
<td>CR76</td>
</tr>
<tr>
<td>CR76</td>
</tr>
<tr>
<td>CR56</td>
</tr>
<tr>
<td>CR56</td>
</tr>
<tr>
<td>CR56</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property_Owner Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property_No</td>
</tr>
<tr>
<td>PG4</td>
</tr>
<tr>
<td>PG16</td>
</tr>
<tr>
<td>PG36</td>
</tr>
</tbody>
</table>
Third Normal Form (3NF)

- Remove transitive dependency.

**E.g.:**

```
Property_Owner (Property_No, PAddress, Rent, Owner_No, OName)
```

Therefore, the 3 NF is a relation that is in 1NF and 2NF and in which no non-primary-key attribute is transitively dependent on the primary key.
2NF to 3NF

- Identify the primary key in the 2NF relation.
- Identify functional dependencies in the relation.
- If transitive dependencies exist on the primary key remove them by placing them in a new relation along with a copy of their dominant.
Example - Normalization
FDs for Customer_Rental Relation

Primary Key

<table>
<thead>
<tr>
<th>Customer_No</th>
<th>Property_No</th>
<th>CName</th>
<th>PAddress</th>
<th>RentStart</th>
<th>RentFinish</th>
<th>Rent</th>
<th>Owner_No</th>
<th>OName</th>
</tr>
</thead>
</table>

fd1

fd2

fd3

fd4

(fd1) (Primary key)

(fd2) (Partial dependency)

(fd3) (Partial dependency)

(fd4) (Transitive dependency)
## Example - Normalization

### Property Owner to 3NF Relations

<table>
<thead>
<tr>
<th>Property_No</th>
<th>PAddress</th>
<th>Rent</th>
<th>Owner_No</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG4</td>
<td>6 Lawrence St, Glasgow</td>
<td>350</td>
<td>CO40</td>
</tr>
<tr>
<td>PG16</td>
<td>5 Novar Dr, Glasgow</td>
<td>450</td>
<td>CO93</td>
</tr>
<tr>
<td>PG36</td>
<td>2 Manor Rd, Glasgow</td>
<td>375</td>
<td>CO93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Owner_No</th>
<th>OName</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO40</td>
<td>Tina Murphy</td>
</tr>
<tr>
<td>CO93</td>
<td>Tony Shaw</td>
</tr>
</tbody>
</table>
Example - Normalization
Process of Decomposition

Customer_Rental

Customer
Rental
Property_for_Rent
Owner

Property_Owner

1NF

2NF

3NF
## Summary of 3NF Relations

### Customer Relation

<table>
<thead>
<tr>
<th>Customer_No</th>
<th>CName</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR76</td>
<td>John Kay</td>
</tr>
<tr>
<td>CR56</td>
<td>Aline Stewart</td>
</tr>
</tbody>
</table>

### Rental Relation

<table>
<thead>
<tr>
<th>Customer_No</th>
<th>Property_No</th>
<th>RentStart</th>
<th>RentFinish</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR76</td>
<td>PG4</td>
<td>1-Jul-94</td>
<td>31-Aug-96</td>
</tr>
<tr>
<td>CR76</td>
<td>PG16</td>
<td>1-Sep-96</td>
<td>1-Sep-98</td>
</tr>
<tr>
<td>CR56</td>
<td>PG4</td>
<td>1-Sep-92</td>
<td>10-Jun-94</td>
</tr>
<tr>
<td>CR56</td>
<td>PG36</td>
<td>10-Oct-94</td>
<td>1-Dec-95</td>
</tr>
<tr>
<td>CR56</td>
<td>PG16</td>
<td>1-Jan-96</td>
<td>10-Aug-96</td>
</tr>
</tbody>
</table>

### Property_for_Rent Relation

<table>
<thead>
<tr>
<th>Property_No</th>
<th>PAddress</th>
<th>Rent</th>
<th>Owner_No</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG4</td>
<td>6 Lawrence St, Glasgow</td>
<td>350</td>
<td>CO40</td>
</tr>
<tr>
<td>PG16</td>
<td>5 Novar Dr, Glasgow</td>
<td>450</td>
<td>CO93</td>
</tr>
<tr>
<td>PG36</td>
<td>2 Manor Rd, Glasgow</td>
<td>375</td>
<td>CO93</td>
</tr>
</tbody>
</table>

### Owner Relation

<table>
<thead>
<tr>
<th>Owner_No</th>
<th>OName</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO40</td>
<td>Tina Murphy</td>
</tr>
<tr>
<td>CO93</td>
<td>Tony Shaw</td>
</tr>
</tbody>
</table>