Chapter 7:
Functional Dependencies &
Normalization for Relational DBs

CuuDuongThanCong.com

https://fb.com/tailieudientucntt
Contents

1 Introduction
2 Functional dependencies (FDs)
3 Normalization
4 Relational database schema design algorithms
5 Key finding algorithms
Contents

1 Introduction

2 Functional dependencies (FDs)

3 Normalization

4 Relational database schema design algorithms

5 Key finding algorithms
Top-Down Database Design

Mini-world

Requirements

E1

R

Conceptual schema

E2

Relation schemas
Introduction

- Each relation schema consists of a number of attributes and the relational database schema consists of a number of relation schemas.
- Attributes are grouped to form a relation schema.
- Need some formal measure of why one grouping of attributes into a relation schema may be better than another.
Introduction

“Goodness” measures:

- Redundant information in tuples.
- Update anomalies: modification, deletion, insertion.
- Reducing the NULL values in tuples.
- Disallowing the possibility of generating spurious tuples.
Redundant information

- The attribute values pertaining to a particular department (DNUMBER, DNAME, DMGRSSN) are repeated for every employee who works for that department.
Update anomalies

di thuong

- Update anomalies: modification, deletion, insertion
  - Modification
    - As the manager of a dept. changes we have to update many values according to employees working for that dept.
    - Easy to make the DB inconsistent.

<table>
<thead>
<tr>
<th>Ename</th>
<th>Ssn</th>
<th>Bdate</th>
<th>Address</th>
<th>Dnumber</th>
<th>Dname</th>
<th>Dmgr_ssn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith, John B.</td>
<td>123456789</td>
<td>1965-01-09</td>
<td>731 Fondren, Houston, TX</td>
<td>5</td>
<td>Research</td>
<td>333445555</td>
</tr>
<tr>
<td>Wong, Franklin T.</td>
<td>333445555</td>
<td>1955-12-08</td>
<td>638 Voss, Houston, TX</td>
<td>5</td>
<td>Research</td>
<td>333445555</td>
</tr>
<tr>
<td>Zelaya, Alicia J.</td>
<td>999887777</td>
<td>1968-07-19</td>
<td>3321 Castle, Spring, TX</td>
<td>4</td>
<td>Administration</td>
<td>987654321</td>
</tr>
<tr>
<td>Wallace, Jennifer S.</td>
<td>987654321</td>
<td>1941-06-20</td>
<td>291 Berry, Bellaire, TX</td>
<td>4</td>
<td>Administration</td>
<td>987654321</td>
</tr>
<tr>
<td>Narayan, Ramesh K.</td>
<td>666884444</td>
<td>1962-09-15</td>
<td>975 FireOak, Humble, TX</td>
<td>5</td>
<td>Research</td>
<td>333445555</td>
</tr>
<tr>
<td>English, Joyce A.</td>
<td>453453453</td>
<td>1972-07-31</td>
<td>5631 Rice, Houston, TX</td>
<td>5</td>
<td>Research</td>
<td>333445555</td>
</tr>
<tr>
<td>Jabbar, Ahmad V.</td>
<td>987987987</td>
<td>1969-03-29</td>
<td>980 Dallas, Houston, TX</td>
<td>4</td>
<td>Administration</td>
<td>987654321</td>
</tr>
<tr>
<td>Borg, James E.</td>
<td>888665555</td>
<td>1937-11-10</td>
<td>450 Stone, Houston, TX</td>
<td>1</td>
<td>Headquarters</td>
<td>888665555</td>
</tr>
</tbody>
</table>
Update anomalies

- Update anomalies: modification, deletion, insertion
  - Deletion: if Borg James E. leaves, we delete his tuple and lose the existing of dept. 1, the name of dept. 1, and who is the manager of dept. 1.

<table>
<thead>
<tr>
<th>Ename</th>
<th>Ssn</th>
<th>Bdate</th>
<th>Address</th>
<th>Dnumber</th>
<th>Dname</th>
<th>Dmgr_ssn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith, John B.</td>
<td>123456789</td>
<td>1965-01-09</td>
<td>731 Fondren, Houston, TX</td>
<td>5</td>
<td>Research</td>
<td>333445555</td>
</tr>
<tr>
<td>Wong, Franklin T.</td>
<td>333445555</td>
<td>1955-12-08</td>
<td>638 Voss, Houston, TX</td>
<td>5</td>
<td>Research</td>
<td>333445555</td>
</tr>
<tr>
<td>Zelaya, Alicia J.</td>
<td>999887777</td>
<td>1968-07-19</td>
<td>3321 Castle, Spring, TX</td>
<td>4</td>
<td>Administration</td>
<td>987654321</td>
</tr>
<tr>
<td>Wallace, Jennifer S.</td>
<td>987654321</td>
<td>1941-06-20</td>
<td>291 Berry, Bellaire, TX</td>
<td>4</td>
<td>Administration</td>
<td>987654321</td>
</tr>
<tr>
<td>Narayan, Ramesh K.</td>
<td>666884444</td>
<td>1962-09-15</td>
<td>975 FireOak, Humble, TX</td>
<td>5</td>
<td>Research</td>
<td>333445555</td>
</tr>
<tr>
<td>English, Joyce A.</td>
<td>453453453</td>
<td>1972-07-31</td>
<td>5631 Rice, Houston, TX</td>
<td>5</td>
<td>Research</td>
<td>333445555</td>
</tr>
<tr>
<td>Jabbar, Ahmad V.</td>
<td>987987987</td>
<td>1969-03-29</td>
<td>980 Dallas, Houston, TX</td>
<td>4</td>
<td>Administration</td>
<td>987654321</td>
</tr>
<tr>
<td>Borg, James E.</td>
<td>888665555</td>
<td>1937-11-10</td>
<td>450 Stone, Houston, TX</td>
<td>1</td>
<td>Headquarters</td>
<td>888665555</td>
</tr>
</tbody>
</table>
Update anomalies

- Update anomalies: modification, deletion, insertion
  - Insertion:
    - How can we create a department before any employees are assigned to it?
Reducing NULL values

- Employees not assigned to any dept.: waste the storage space.
- Other difficulties: aggregation operations (e.g., COUNT, SUM) and joins.
Generation spurious tuples

- Disallowing the possibility of generating spurious tuples.

EMP_PROJ(SSN, PNUMBER, HOURS, ENAME, PNAME, PLOCATION)

EMP_LOCS(ENAME, PLOCATION)

EMP_PROJ1(SSN, PNUMBER, HOURS, PNAME, PLOCATION)

- Generation of invalid and spurious data during JOINS: PLOCATION is the attribute that relates EMP_LOCS and EMP_PROJ1, and PLOCATION is neither a primary key nor a foreign key in either EMP_LOCS or EMP_PROJ1.
### Generation spurious tuples

<table>
<thead>
<tr>
<th><strong>EMP_LOCS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ename</strong></td>
</tr>
<tr>
<td>Smith, John B.</td>
</tr>
<tr>
<td>Smith, John B.</td>
</tr>
<tr>
<td>Narayan, Ramesh K.</td>
</tr>
<tr>
<td>English, Joyce A.</td>
</tr>
<tr>
<td>English, Joyce A.</td>
</tr>
<tr>
<td>Wong, Franklin T.</td>
</tr>
<tr>
<td>Wong, Franklin T.</td>
</tr>
<tr>
<td>Wong, Franklin T.</td>
</tr>
<tr>
<td>Zelaya, Alicia J.</td>
</tr>
<tr>
<td>Jabbar, Ahmad V.</td>
</tr>
<tr>
<td>Wallace, Jennifer S.</td>
</tr>
<tr>
<td>Wallace, Jennifer S.</td>
</tr>
<tr>
<td>Borg, James E.</td>
</tr>
</tbody>
</table>
### Generation spurious tuples

#### EMP_PROJ1

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Pnumber</th>
<th>Hours</th>
<th>Pname</th>
<th>Plocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>1</td>
<td>32.5</td>
<td>ProductX</td>
<td>Bellaire</td>
</tr>
<tr>
<td>123456789</td>
<td>2</td>
<td>7.5</td>
<td>ProductY</td>
<td>Sugarland</td>
</tr>
<tr>
<td>666884444</td>
<td>3</td>
<td>40.0</td>
<td>ProductZ</td>
<td>Houston</td>
</tr>
<tr>
<td>453453453</td>
<td>1</td>
<td>20.0</td>
<td>ProductX</td>
<td>Bellaire</td>
</tr>
<tr>
<td>453453453</td>
<td>2</td>
<td>20.0</td>
<td>ProductY</td>
<td>Sugarland</td>
</tr>
<tr>
<td>333445555</td>
<td>2</td>
<td>10.0</td>
<td>ProductY</td>
<td>Sugarland</td>
</tr>
<tr>
<td>333445555</td>
<td>3</td>
<td>10.0</td>
<td>ProductZ</td>
<td>Houston</td>
</tr>
<tr>
<td>333445555</td>
<td>10</td>
<td>10.0</td>
<td>Computerization</td>
<td>Stafford</td>
</tr>
<tr>
<td>333445555</td>
<td>20</td>
<td>10.0</td>
<td>Reorganization</td>
<td>Houston</td>
</tr>
<tr>
<td>999887777</td>
<td>30</td>
<td>30.0</td>
<td>Newbenefits</td>
<td>Stafford</td>
</tr>
<tr>
<td>999887777</td>
<td>10</td>
<td>10.0</td>
<td>Computerization</td>
<td>Stafford</td>
</tr>
<tr>
<td>987987987</td>
<td>10</td>
<td>35.0</td>
<td>Computerization</td>
<td>Stafford</td>
</tr>
<tr>
<td>987987987</td>
<td>30</td>
<td>5.0</td>
<td>Newbenefits</td>
<td>Stafford</td>
</tr>
<tr>
<td>987654321</td>
<td>30</td>
<td>20.0</td>
<td>Newbenefits</td>
<td>Stafford</td>
</tr>
<tr>
<td>987654321</td>
<td>20</td>
<td>15.0</td>
<td>Reorganization</td>
<td>Houston</td>
</tr>
<tr>
<td>888665555</td>
<td>20</td>
<td>NULL</td>
<td>Reorganization</td>
<td>Houston</td>
</tr>
<tr>
<td>Ssn</td>
<td>Pnumber</td>
<td>Hours</td>
<td>Pname</td>
<td>Plocation</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>-------</td>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>123456789</td>
<td>1</td>
<td>32.5</td>
<td>ProductX</td>
<td>Bellaire</td>
</tr>
<tr>
<td>123456789</td>
<td>1</td>
<td>32.5</td>
<td>ProductX</td>
<td>Bellaire</td>
</tr>
<tr>
<td>123456789</td>
<td>2</td>
<td>7.5</td>
<td>ProductY</td>
<td>Sugarland</td>
</tr>
<tr>
<td>123456789</td>
<td>2</td>
<td>7.5</td>
<td>ProductY</td>
<td>Sugarland</td>
</tr>
<tr>
<td>123456789</td>
<td>2</td>
<td>7.5</td>
<td>ProductY</td>
<td>Sugarland</td>
</tr>
<tr>
<td>6666884444</td>
<td>3</td>
<td>40.0</td>
<td>ProductZ</td>
<td>Houston</td>
</tr>
<tr>
<td>6666884444</td>
<td>3</td>
<td>40.0</td>
<td>ProductZ</td>
<td>Houston</td>
</tr>
<tr>
<td>453453453</td>
<td>1</td>
<td>20.0</td>
<td>ProductX</td>
<td>Bellaire</td>
</tr>
<tr>
<td>453453453</td>
<td>1</td>
<td>20.0</td>
<td>ProductX</td>
<td>Bellaire</td>
</tr>
<tr>
<td>453453453</td>
<td>2</td>
<td>20.0</td>
<td>ProductY</td>
<td>Sugarland</td>
</tr>
<tr>
<td>453453453</td>
<td>2</td>
<td>20.0</td>
<td>ProductY</td>
<td>Sugarland</td>
</tr>
<tr>
<td>453453453</td>
<td>2</td>
<td>20.0</td>
<td>ProductY</td>
<td>Sugarland</td>
</tr>
<tr>
<td>333445555</td>
<td>2</td>
<td>10.0</td>
<td>ProductY</td>
<td>Sugarland</td>
</tr>
<tr>
<td>333445555</td>
<td>2</td>
<td>10.0</td>
<td>ProductY</td>
<td>Sugarland</td>
</tr>
<tr>
<td>333445555</td>
<td>2</td>
<td>10.0</td>
<td>ProductY</td>
<td>Sugarland</td>
</tr>
<tr>
<td>333445555</td>
<td>3</td>
<td>10.0</td>
<td>ProductZ</td>
<td>Houston</td>
</tr>
<tr>
<td>333445555</td>
<td>3</td>
<td>10.0</td>
<td>ProductZ</td>
<td>Houston</td>
</tr>
<tr>
<td>333445555</td>
<td>10</td>
<td>10.0</td>
<td>Computerization</td>
<td>Stafford</td>
</tr>
<tr>
<td>333445555</td>
<td>20</td>
<td>10.0</td>
<td>Reorganization</td>
<td>Houston</td>
</tr>
<tr>
<td>333445555</td>
<td>20</td>
<td>10.0</td>
<td>Reorganization</td>
<td>Houston</td>
</tr>
</tbody>
</table>
Summary of Design Guidelines

- “Goodness” measures:
  - Redundant information in tuples
  - Update anomalies: modification, deletion, insertion
  - Reducing the NULL values in tuples
  - Disallowing the possibility of generating spurious tuples

  Normalization

- It helps DB designers determine the best relation schemas.
  - A formal framework for analyzing relation schemas based on their keys and on the functional dependencies among their attributes.
  - A series of normal form tests that can be carried out on individual relation schemas so that the relational database can be normalized to any desired degree.

- It is based on the concept of normal form 1NF, 2NF, 3NF, BCNF, 4NF, 5 NF.
## Contents

1. Introduction  
2. Functional dependencies (FDs)  
3. Normalization  
4. Relational database schema design algorithms  
5. Key finding algorithms
Functional Dependencies (FDs)

- Definition of FD
- Direct, indirect, partial dependencies
- Inference Rules for FDs
- Equivalence of Sets of FDs
- Minimal Sets of FDs
Definition of Functional dependencies

- Functional dependencies (FDs) are used to specify *formal measures* of the "goodness" of relational designs.
- FDs and keys are used to define *normal forms* for relations.
- FDs are *constraints* that are derived from the *meaning* and *interrelationships* of the data attributes.
- A set of attributes $X$ *functionally determines* a set of attributes $Y$ if the value of $X$ determines a unique value for $Y$. 
Definition of Functional dependencies

- X -> Y holds if whenever two tuples have the same value for X, they must have the same value for Y.
- 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).
- Examples:
  - social security number determines employee name: SSN -> ENAME
  - project number determines project name and location: PNUMBER -> {PNAME, PLOCATION}
  - employee ssn and project number determines the hours per week that the employee works on the project: {SSN, PNUMBER} -> HOURS
Definition of Functional dependencies

- If K is a key of R, then K functionally determines all attributes in R (since we never have two distinct tuples with \( t_1[K] = t_2[K] \)).
Functional Dependencies (FDs)

- Definition of FD
- Direct, indirect, partial dependencies
- Inference Rules for FDs
- Equivalence of Sets of FDs
- Minimal Sets of FDs
Direct, indirect, partial dependencies

- **Direct dependency (fully functional dependency):** All attributes in a R must be fully functionally dependent on the primary key (or the PK is a determinant of all attributes in R).

```
Performer-id -> Performer-name
             |                   | Performer-type
             |                   | Performer-location
```

Jan - 2015
CuuDuongThanCong.com

https://fb.com/tailieudientucntt
Direct, indirect, partial dependencies

- **Indirect dependency (transitive dependency):** Value of an attribute is not determined directly by the primary key.

```plaintext
Performer-id  →  Performer-name
               ↓                ↓
        Performer-type  →  Performer-location
                           ↓
                                →  Fee
```

- Performer-id
- Performer-name
- Performer-type
- Performer-location
- Fee
Direct, indirect, partial dependencies

- Partial dependency
  - **Composite determinant**: more than one value is required to determine the value of another attribute, the combination of values is called a composite determinant.

    \[
    \text{EMP\_PROJ(}\text{SSN, PNUMBER, HOURS, ENAME, PNAME, PLOCATION)}
    \{
    \text{SSN, PNUMBER}\} \rightarrow \text{HOURS}
    \]

  - **Partial dependency**: if the value of an attribute does not depend on an entire composite determinant, but only part of it, the relationship is known as the partial dependency.

    \[
    \text{SSN} \rightarrow \text{ENAME}
    \text{PNUMBER} \rightarrow \{\text{PNAME, PLOCATION}\}
    \]
Direct, indirect, partial dependencies

- Partial dependency

```
Performer-id
  ▸ Performer-name
  ▸ Performer-type
  ▸ Performer-location

Agent-id
  ▸ Fee
  ▸ Agent-name
  ▸ Agent-location
```
Functional Dependencies (FDs)

- Definition of FD
- Direct, indirect, partial dependencies
- Inference Rules for FDs
- Equivalence of Sets of FDs
- Minimal Sets of FDs
Inference Rules for FDs

- Given a set of FDs F, we can infer additional FDs that hold whenever the FDs in F hold.

Armstrong's inference rules:

IR1. (Reflexive) If $Y \subseteq X$, then $X \rightarrow Y$.

IR2. (Augmentation) If $X \rightarrow Y$, then $XZ \rightarrow YZ$.
   
   (Notation: $XZ$ stands for $X \cup Z$)

IR3. (Transitive) If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$. 
Inference Rules for FDs

- Some **additional inference rules** that are useful:
  - **(Decomposition)** If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
  - **(Union)** If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
  - **(Pseudotransitivity)** If $X \rightarrow Y$ and $WY \rightarrow Z$, then $WX \rightarrow Z$

- The last three inference rules, as well as any other inference rules, can be deduced from IR1, IR2, and IR3 (completeness property).
Inference Rules for FDs

- **Closure** of a set $F$ of FDs is the set $F^+$ of all FDs that can be inferred from $F$.

- **Closure** of a set of attributes $X$ with respect to $F$ is the set $X^+$ of all attributes that are functionally determined by $X$.

- $X^+$ can be calculated by repeatedly applying IR1, IR2, IR3 using the FDs in $F$. 
Inference Rules for FDs

Algorithm 16.1. Determining $X^+$, the Closure of $X$ under $F$

**Input:** A set $F$ of FDs on a relation schema $R$, and a set of attributes $X$, which is a subset of $R$.

$X^+ := X$;

repeat

old$X^+ := X^+$;

for each functional dependency $Y \rightarrow Z$ in $F$ do

if $X^+ \supseteq Y$ then $X^+ := X^+ \cup Z$;

until ($X^+ = \text{old}X^+$);
Inference Rules for FDs

Consider a relation \( R(A, B, C, D, E) \) with the following dependencies F:

- (1) \( AB \rightarrow C \),
- (2) \( CD \rightarrow E \),
- (3) \( DE \rightarrow B \)

Find \( \{A,B\}^+ \) ?
Functional Dependencies (FDs)

- Definition of FD
- Direct, indirect, partial dependencies
- Inference Rules for FDs
- Equivalence of Sets of FDs
- Minimal Sets of FDs
Equivalence of Sets of FDs

- Two sets of FDs F and G are **equivalent** if $F^+ = G^+$. 

- **Definition:** F **covers** G if $G^+ \subseteq F^+$. F and G are equivalent if F covers G and G covers F.

- There is an algorithm for checking equivalence of sets of FDs.
Functional Dependencies (FDs)

- Definition of FD
- Direct, indirect, partial dependencies
- Inference Rules for FDs
- Equivalence of Sets of FDs
- Minimal Sets of FDs
Minimal Sets of FDs

- A set of FDs is **minimal** if it satisfies the following conditions:
  1. Every dependency in $F$ has a single attribute for its right-hand side.
  2. We cannot remove any dependency from $F$ and have a set of dependencies that is equivalent to $F$.
  3. We cannot replace any dependency $X \rightarrow A$ in $F$ with a dependency $Y \rightarrow A$, where $Y$ proper-subset-of $X$ ( $Y$ subset-of $X$) and still have a set of dependencies that is equivalent to $F$. 
Algorithm 16.2. Finding a Minimal Cover $F$ for a Set of Functional Dependencies $E$

**Input:** A set of functional dependencies $E$.


2. Replace each functional dependency $X \rightarrow \{A_1, A_2, \ldots, A_n\}$ in $F$ by the $n$ functional dependencies $X \rightarrow A_1$, $X \rightarrow A_2$, ..., $X \rightarrow A_n$.

3. For each functional dependency $X \rightarrow A$ in $F$ for each attribute $B$ that is an element of $X$

   if $\{ F - \{X \rightarrow A\} \} \cup \{ (X - \{B\}) \rightarrow A\}$ is equivalent to $F$

   then replace $X \rightarrow A$ with $(X - \{B\}) \rightarrow A$ in $F$.

4. For each remaining functional dependency $X \rightarrow A$ in $F$

   if $\{ F - \{X \rightarrow A\} \}$ is equivalent to $F$,

   then remove $X \rightarrow A$ from $F$. 
Minimal Sets of FDs

- Every set of FDs has an equivalent minimal set.
- There can be several equivalent minimal sets.
- There is no simple algorithm for computing a minimal set of FDs that is equivalent to a set \( F \) of FDs.
- To synthesize a set of relations, we assume that we start with a set of dependencies that is a minimal set.
Contents

1 Introduction
2 Functional dependencies (FDs)
3 Normalization
4 Relational database schema design algorithms
5 Key finding algorithms
Normalization

- **Normalization**: The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations.

- **Normal form**: Using keys and FDs of a relation to certify whether a relation schema is in a particular normal form.

- **Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties.

- The database designers *need not* normalize to the highest possible normal form (3NF, BCNF or 4NF).
Normalization

There are two important properties of decompositions:

(a) non-additive or losslessness of the corresponding join.

(b) preservation of the functional dependencies.

Note that property (a) is extremely important and cannot be sacrificed. Property (b) is less stringent and may be sacrificed (see chapter 16).
Normalization

- **Superkey** of R: A set of attributes $SK$ of $R$ such that no two tuples in any valid relation instance $r(R)$ will have the same value for $SK$. That is, for any distinct tuples $t_1$ and $t_2$ in $r(R)$, $t_1[SK] \neq t_2[SK]$.

- **Key** of R: A "minimal" superkey; that is, a superkey $K$ such that removal of any attribute from $K$ results in a set of attributes that is not a superkey.

- If $K$ is a **key** of R, then $K$ functionally determines all attributes in $R$.  

Jan - 2015  
CuuDuongThanCong.com  
https://fb.com/tailieudientucntt
Normalization

Two new concepts:

- A **Prime attribute** must be a member of some **candidate key**.
- A **Nonprime attribute** is not a prime attribute: it is not a member of any candidate key.
Normalization

- 1NF and dependency problems
- 2NF – solves partial dependency
- 3NF – solves indirect dependency
- BCNF – well-normalized relations
First normal form (1NF): there is only one value at the intersection of each row and column of a relation - no set valued attributes in 1 NF → Disallows composite attributes, multivalued attributes, and nested relations.

The only attribute values permitted by 1NF are single atomic (or indivisible) values.
### (a) DEPARTMENT

<table>
<thead>
<tr>
<th>Dname</th>
<th>Dnumber</th>
<th>Dmgr_ssn</th>
<th>Dlocations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### (b) DEPARTMENT

<table>
<thead>
<tr>
<th>Dname</th>
<th>Dnumber</th>
<th>Dmgr_ssn</th>
<th>Dlocations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research</td>
<td>5</td>
<td>333445555</td>
<td>{Bellaire, Sugarland, Houston}</td>
</tr>
<tr>
<td>Administration</td>
<td>4</td>
<td>987654321</td>
<td>{Stafford}</td>
</tr>
<tr>
<td>Headquarters</td>
<td>1</td>
<td>888665555</td>
<td>{Houston}</td>
</tr>
</tbody>
</table>

### (c) DEPARTMENT

<table>
<thead>
<tr>
<th>Dname</th>
<th>Dnumber</th>
<th>Dmgr_ssn</th>
<th>Dlocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research</td>
<td>5</td>
<td>333445555</td>
<td>Bellaire</td>
</tr>
<tr>
<td>Research</td>
<td>5</td>
<td>333445555</td>
<td>Sugarland</td>
</tr>
<tr>
<td>Research</td>
<td>5</td>
<td>333445555</td>
<td>Houston</td>
</tr>
<tr>
<td>Administration</td>
<td>4</td>
<td>987654321</td>
<td>Stafford</td>
</tr>
<tr>
<td>Headquarters</td>
<td>1</td>
<td>888665555</td>
<td>Houston</td>
</tr>
</tbody>
</table>
### 1NF

(a) EMP_PROJ

<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>
</tbody>
</table>

(b) EMP_PROJ

<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>
</tbody>
</table>
1NF

(c)

EMP_PROJ1

| Ssn | Ename |

EMP_PROJ2

| Ssn | Pnumber | Hours |

CuuDuongThanCong.com

https://fb.com/tailieudientucntt
Normalization

- 1NF and dependency problems
- 2NF – solves partial dependency
- 3NF – solves indirect dependency
- BCNF – well-normalized relations
Second normal form (2NF) - all attributes must be fully functionally dependent on the primary key.

2NF solves partial dependency problem in 1NF.

2NF normalized: Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
(a) EMP_PROJ

<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>FD1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FD2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FD3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2NF Normalization

EP1

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Pnumber</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

EP2

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Ename</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD2</td>
<td></td>
</tr>
</tbody>
</table>

EP3

<table>
<thead>
<tr>
<th>Pnumber</th>
<th>Pname</th>
<th>Plocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem with 2NF:
- Insertion
- Modification
- Deletion
Normalization

- 1NF and dependency problems
- 2NF – solves partial dependency
- 3NF – solves indirect dependency
- BCNF – well-normalized relations
3NF

- A relation schema \( R \) is in **third normal form (3NF)** if it is in 2NF *and* no non-prime attribute \( A \) in \( R \) is transitively dependent on the primary key.

**NOTE:**
- In \( X \rightarrow Y \) and \( Y \rightarrow Z \), with \( X \) as the primary key, we consider this a problem only if \( Y \) is *not* a candidate key. When \( Y \) is a candidate key, there is no problem with the transitive dependency.
- E.g., Consider EMP (SSN, Emp#, Salary).
- Here, SSN \( \rightarrow \) Emp#. Emp# \( \rightarrow \) Salary and Emp# is a candidate key.
3NF

- 3NF solves indirect (transitive) dependencies problem in 1NF and 2NF.

- **3NF normalized:** identify all transitive dependencies and each transitive dependency will form a new relation.
3NF

(b) EMP_DEPT

<table>
<thead>
<tr>
<th>Ename</th>
<th>Ssn</th>
<th>Bdate</th>
<th>Address</th>
<th>Dnumber</th>
<th>Dname</th>
<th>Dmgr_ssn</th>
</tr>
</thead>
</table>

3NF Normalization

ED1

<table>
<thead>
<tr>
<th>Ename</th>
<th>Ssn</th>
<th>Bdate</th>
<th>Address</th>
<th>Dnumber</th>
</tr>
</thead>
</table>

ED2

<table>
<thead>
<tr>
<th>Dnumber</th>
<th>Dname</th>
<th>Dmgr_ssn</th>
</tr>
</thead>
</table>

CuuDuongThanCong.com
3NF

- LOCATION (city, street, zip-code)
- \( F = \{ \text{city, street} \rightarrow \text{zip-code}, \text{zip-code} \rightarrow \text{city} \) \\
  \( \text{Key}_1 : \text{city, street} \) (primary key) \\
  \( \text{Key}_2 : \text{street, zip-code} \)

<table>
<thead>
<tr>
<th>city</th>
<th>street</th>
<th>zip-code</th>
</tr>
</thead>
<tbody>
<tr>
<td>NY</td>
<td>55th</td>
<td>484</td>
</tr>
<tr>
<td>NY</td>
<td>56th</td>
<td>484</td>
</tr>
<tr>
<td>LA</td>
<td>55th</td>
<td>473</td>
</tr>
<tr>
<td>LA</td>
<td>56th</td>
<td>473</td>
</tr>
<tr>
<td>LA</td>
<td>57th</td>
<td>474</td>
</tr>
</tbody>
</table>
## SUMMARY OF NORMAL FORMS

### based on Primary Keys

Summary of Normal Forms Based on Primary Keys and Corresponding Normalization

<table>
<thead>
<tr>
<th>Normal Form</th>
<th>Test</th>
<th>Remedy (Normalization)</th>
</tr>
</thead>
<tbody>
<tr>
<td>First (1NF)</td>
<td>Relation should have no multivalued attributes or nested relations.</td>
<td>Form new relations for each multi-valued attribute or nested relation.</td>
</tr>
<tr>
<td>Second (2NF)</td>
<td>For relations where primary key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the primary key.</td>
<td>Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.</td>
</tr>
<tr>
<td>Third (3NF)</td>
<td>Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.</td>
<td>Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).</td>
</tr>
</tbody>
</table>
The above definitions consider the primary key only.

The following more general definitions take into account relations with multiple candidate keys.
A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is *not partially functionally dependent on any key* of R.

A relation schema R is in **third normal form (3NF)** if whenever a FD $X \rightarrow A$ holds in R, then either:

(a) $X$ is a superkey of R, or
(b) $A$ is a prime attribute of R
General Normal Form Example

The LOTS relation with its functional dependencies.
General Normal Form Example

Decomposing into the 2NF relations
Decomposing LOTS1 into the 3NF relations
Normalization

- 1NF and dependency problems
- 2NF – solves partial dependency
- 3NF – solves indirect dependency
- BCNF – well-normalized relations
A relation schema $R$ is in Boyce-Codd Normal Form (BCNF) if whenever an FD $X \rightarrow A$ holds in $R$, then $X$ is a superkey of $R$. 
BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition.
**BCNF**

- **TEACH** (Student, Course, Instructor)
- **FD1**: \{Student, Course\} $\rightarrow$ Instructor
- **FD2**: Instructor $\rightarrow$ Course

<table>
<thead>
<tr>
<th>Student</th>
<th>Course</th>
<th>Instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Narayan</td>
<td>Database</td>
<td>Mark</td>
</tr>
<tr>
<td>Smith</td>
<td>Database</td>
<td>Navathe</td>
</tr>
<tr>
<td>Smith</td>
<td>Operating Systems</td>
<td>Ammar</td>
</tr>
<tr>
<td>Smith</td>
<td>Theory</td>
<td>Schulman</td>
</tr>
<tr>
<td>Wallace</td>
<td>Database</td>
<td>Mark</td>
</tr>
<tr>
<td>Wallace</td>
<td>Operating Systems</td>
<td>Ahammad</td>
</tr>
<tr>
<td>Wong</td>
<td>Database</td>
<td>Omiecinski</td>
</tr>
<tr>
<td>Zelaya</td>
<td>Database</td>
<td>Navathe</td>
</tr>
<tr>
<td>Narayan</td>
<td>Operating Systems</td>
<td>Ammar</td>
</tr>
</tbody>
</table>
BCNF

Three possible pairs:

1. \{Student, Instructor\} and \{Student, Course\}
2. \{Course, Instructor\} and \{Course, Student\}
3. \{Instructor, Course\} and \{Instructor, Student\}

All three decompositions lose the functional dependency FD1. The desirable decomposition of those just shown is 3 because it will not generate spurious tuples after a join.
Notes & Suggestions

[1], chapter 15:
- 4NF: based on \textit{multivalued dependency} (MVD)
- 5NF: based on join dependency
  - Such a dependency is very difficult to detect in practice and therefore, normalization into 5NF is considered very rarely in practice

- Other normal forms & algorithms
- ER modeling: top-down database design
  - Bottom-up database design ??

[1], chapter 16: Properties of Relational Decompositions
Contents

1 Introduction

2 Functional dependencies (FDs)

3 Normalization

4 Relational database schema design algorithms

5 Key finding algorithms
Dependency-Preserving Decomposition into 3NF Schemas

**Algorithm 16.4.** Relational Synthesis into 3NF with Dependency Preservation

**Input:** A universal relation $R$ and a set of functional dependencies $F$ on the attributes of $R$.

1. Find a minimal cover $G$ for $F$ (use Algorithm 16.2);
2. For each left-hand-side $X$ of a functional dependency that appears in $G$, create a relation schema in $D$ with attributes \{X ∪ \{A_1\} ∪ \{A_2\} ... ∪ \{A_k\}\}, where $X → A_1$, $X → A_2$, ..., $X → A_k$ are the only dependencies in $G$ with $X$ as the left-hand-side ($X$ is the key of this relation);
3. Place any remaining attributes (that have not been placed in any relation) in a single relation schema to ensure the attribute preservation property.
Nonadditive Join Decomposition into BCNF Schemas

Algorithm 16.5. Relational Decomposition into BCNF with Nonadditive Join Property

**Input:** A universal relation $R$ and a set of functional dependencies $F$ on the attributes of $R$.

1. Set $D := \{R\}$ ;
2. While there is a relation schema $Q$ in $D$ that is not in BCNF do
   
   choose a relation schema $Q$ in $D$ that is not in BCNF;
   find a functional dependency $X \rightarrow Y$ in $Q$ that violates BCNF;
   replace $Q$ in $D$ by two relation schemas $(Q - Y)$ and $(X \cup Y)$;


Dependency-Preserving and Nonadditive (Lossless) Join Decomposition into 3NF Schemas

**Algorithm 16.6.** Relational Synthesis into 3NF with Dependency Preservation and Nonadditive Join Property

**Input:** A universal relation $R$ and a set of functional dependencies $F$ on the attributes of $R$.

1. Find a minimal cover $G$ for $F$ (use Algorithm 16.2).
2. For each left-hand-side $X$ of a functional dependency that appears in $G$, create a relation schema in $D$ with attributes $\{X \cup \{A_1\} \cup \{A_2\} \ldots \cup \{A_k\}\}$, where $X \rightarrow A_1$, $X \rightarrow A_2$, ..., $X \rightarrow A_k$ are the only dependencies in $G$ with $X$ as left-hand-side ($X$ is the key of this relation).
3. If none of the relation schemas in $D$ contains a key of $R$, then create one more relation schema in $D$ that contains attributes that form a key of $R$.
4. Eliminate redundant relations from the resulting set of relations in the relational database schema. A relation $R$ is considered redundant if $R$ is a projection of another relation $S$ in the schema; alternately, $R$ is subsumed by $S$. 

Dependency-Preserving and Nonadditive (Lossless) Join Decomposition into 3NF Schemas

- **Algorithm 16.6:**
  - Preserves dependencies.
  - Has the nonadditive join property.
  - Is such that each resulting relation schema in the decomposition is in 3NF.

- **It is preferred over Algorithm 16.4.**
## Contents

1. Introduction
2. Functional dependencies (FDs)
3. Normalization
4. Relational database schema design algorithms
5. Key finding algorithms

Jan - 2015

CuuDuongThanCong.com

https://fb.com/tailieudientucntt
Key-finding algorithm (1)

By Elmasri and Navathe

Algorithm 16.2(a). Finding a Key $K$ for $R$ Given a set $F$ of Functional Dependencies

**Input:** A relation $R$ and a set of functional dependencies $F$ on the attributes of $R$.

1. Set $K := R$.
2. For each attribute $A$ in $K$
   
   
   \{ compute $(K - A)^+$ with respect to $F$;
   if $(K - A)^+$ contains all the attributes in $R$,
   then set $K := K - \{A\}$
   \};
Key-finding algorithm (1)

By Elmasri and Navathe

- In algorithm (1), we start by setting $K$ to all the attributes of $R$; we then remove one attribute at a time and check whether the remaining attributes still form a superkey.
- The algorithm (1) determines only one key out of the possible candidate keys for $R$; the key returned depends on the order in which attributes are removed from $R$ in step 2.
Key-finding algorithm (2)

By Hossein Saiedian and Thomas Spencer

**Input:** A relation R and a set of functional dependencies F on the attributes of R.

**Output:** all candidate keys of R

Let:

- $U$ contain **all** attributes of R.
- $U_L$ contain attributes of R that occur only on the left-hand side of FDs in F.
- $U_R$ contain attributes of R that occur only on the right-hand side of FDs in F.
- $U_B$ contain attributes of R that occur on both sides of FDs in F.
**Key-finding algorithm (2)**

*By Hossein Saiedian and Thomas Spencer*

**Note:**

- $U_L \cap U_R = \emptyset$, $U_L \cap U_B = \emptyset$, and $U_R \cap U_B = \emptyset$
- $U_L \cup U_R \cup U_B = U$
- For every attribute $A \in U$, if $A \in U_L$, then $A$ **must be** part of every candidate key of $R$.
- For every attribute $A \in U$, if $A \in U_R$, then $A$ **will not** be part of any candidate key of $R$. 
Key-finding algorithm (2)

By Hossein Saiedian and Thomas Spencer

Input: A relation $R$ and a set of functional dependencies $F$ on the attributes of $R$.
Output: all candidate keys of $R$

1. Determine $U_L$, $U_R$ and $U_B$
2. If $U_L^+ = U$ under $F$, then $U_L$ forms the only key of $R$ and the algorithm stops here.
   Else: move to step 3 // $U_L^+ \neq U$ under $F$
3. Consider every subsets $U_{Bi}$ of $U_B$: $U_{Bi} \subset U_B$
   For each $U_{Bi}$, if $(U_L \cup U_{Bi})^+ = U$ under $F$, then $K_i = (U_L \cup U_{Bi})$ is a candidate key of $R$ (*)

(*) If $K_i = (U_L \cup U_{Bi})$ is a candidate key of $R$, then we need not to check $U_{Bj} \subset U_B$ where $U_{Bi} \subset U_{Bj}$
Key-finding algorithm (2)

By Hossein Saiedian and Thomas Spencer

- A simple categorization of attributes into the sets $U_L$, $U_L$ and $U_L$ allows to distinguish between those attributes that will participate in the candidate keys of a relational database schema and those that do not.

- The algorithm (2) finds all candidate keys.
Contents

1 Introduction
2 Functional dependencies (FDs)
3 Normalization
4 Relational database schema design algorithms
5 Key finding algorithms
Q & A
Exercise 1

Consider the universal relation $R = \{A, B, C, D, E, F\}$ and the set of functional dependencies:

1) $A \rightarrow B$
2) $C, D \rightarrow A$
3) $B, C \rightarrow D$
4) $A, E \rightarrow F$
5) $C, E \rightarrow D$

What is the key for $R$?
Exercise 2

Consider the universal relation \( R = \{A, B, C, D, E, F\} \) and the set of functional dependencies:

1) \( A, D \rightarrow B \)
2) \( A, B \rightarrow E \)
3) \( C \rightarrow D \)
4) \( B \rightarrow C \)
5) \( A, C \rightarrow F \)

What is the key for \( R \)? Decompose \( R \) into 2NF, 3NF, and BCNF relations.
Exercise 3

Consider the universal relation \( R = \{A, B, C, D, E, F\} \) and the set of functional dependencies:

1) \( A \rightarrow B \)
2) \( C \rightarrow A, D \)
3) \( A, F \rightarrow C, E \)

What is the key for \( R \)? Decompose \( R \) into 2NF, 3NF, and BCNF relations.
Exercise 4

Consider the universal relation \( R = \{A, B, C, D, E, F, G, H, I, J\} \) and the set of functional dependencies:

1) \( A, B \rightarrow C \)
2) \( B, D \rightarrow E, F \)
3) \( A, D \rightarrow G, H \)
4) \( A \rightarrow I \)
5) \( H \rightarrow J \)

What is the key for \( R \)? Decompose \( R \) into 2NF, 3NF, and BCNF relations.
Review questions

1) Define first, second, and third normal forms when only primary keys are considered. How do the general definitions of 2NF and 3NF, which consider all keys of a relation, differ from those that consider only primary keys?

2) Define Boyce-Codd normal form. How does it differ from 3NF? Why is it considered a stronger form of 3NF?

3) What is a minimal set of functional dependencies? Does every set of dependencies have a minimal equivalent set? Is it always unique?