Chapter 7

Functional Dependencies
Outline

● Informal Design Guidelines for Relational Databases
  – Semantics of the Relation Attributes
  – Redundant Information in Tuples and Update Anomalies
  – Null Values in Tuples
  – Spurious Tuples

● Functional Dependencies (FDs)
  – Definition of FD
  – Inference Rules for FDs
  – Equivalence of Sets of FDs
  – Minimal Sets of FDs
Informal Design Guidelines for Relational Databases (1)

- What is relational database design?
  The grouping of attributes to form "good" relation schemas

- Two levels of relation schemas
  - The logical "user view" level
  - The storage "base relation" level

- Design is concerned mainly with base relations

- What are the criteria for "good" base relations?
Informal Design Guidelines for Relational Databases (2)

- We first discuss informal guidelines for good relational design
- Then we discuss formal concepts of functional dependencies and normal forms
  - 1NF (First Normal Form)
  - 2NF (Second Normal Form)
  - 3NF (Third Normal Form)
  - BCNF (Boyce-Codd Normal Form)
- Additional types of dependencies, further normal forms, relational design algorithms by synthesis are discussed in Chapter 11
Semantics of the Relation Attributes

GUIDELINE 1: Informally, each tuple in a relation should represent one entity or relationship instance. (Applies to individual relations and their attributes).

- Attributes of different entities (EMPLOYEES, DEPARTMENTs, PROJECTs) should not be mixed in the same relation
- Only foreign keys should be used to refer to other entities
- Entity and relationship attributes should be kept apart as much as possible.

Bottom Line: Design a schema that can be explained easily relation by relation. The semantics of attributes should be easy to interpret.
A simplified COMPANY relational database schema

- **EMPLOYEE**
  - ENAME
  - SSN
  - BDATE
  - ADDRESS
  - DNUMBER
  - p.k.
  - f.k.

- **DEPARTMENT**
  - DNAME
  - DNUMBER
  - DMGRSSN
  - p.k.
  - f.k.

- **DEPT_LOCATIONS**
  - DNUMBER
  - DLOCATION
  - p.k.
  - f.k.

- **PROJECT**
  - PNAME
  - PNUMBER
  - PLOCATION
  - DNUM
  - p.k.
  - f.k.

- **WORKS_ON**
  - SSN
  - PNUMBER
  - HOURS
  - p.k.
  - f.k.
Redundant Information in Tuples and Update Anomalies

- Mixing attributes of multiple entities may cause problems
- Information is stored redundantly wasting storage
- Problems with update anomalies
  - Insertion anomalies
  - Deletion anomalies
  - Modification anomalies
EXAMPLE OF AN UPDATE ANOMALY (1)

Consider the relation:
EMP_PROJ (Emp#, Proj#, Ename, Pname, No_hours)

- **Update Anomaly:** Changing the name of project number P1 from “Billing” to “Customer-Accounting” may cause this update to be made for all 100 employees working on project P1.
EXAMPLE OF AN UPDATE ANOMALY (2)

- **Insert Anomaly:** Cannot insert a project unless an employee is assigned to.

  *Inversely* - Cannot insert an employee unless an he/she is assigned to a project.

- **Delete Anomaly:** When a project is deleted, it will result in deleting all the employees who work on that project. Alternately, if an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.
Two relation schemas and their functional dependencies. Both suffer from update anomalies. (a) The EMP_DEPT relation schema. (b) The EMP_PROJ relation schema.
### EMP_DEPT

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>Bdate</th>
<th>Address</th>
<th>Dnumber</th>
<th>Department</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>999987777</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>

### EMP_PROJ

<table>
<thead>
<tr>
<th>SSN</th>
<th>Pnumber</th>
<th>Hours</th>
<th>Name</th>
<th>Project</th>
<th>Location</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>
Guideline to Redundant Information in Tuples and Update Anomalies

- **GUIDELINE 2:** Design a schema that does not suffer from the insertion, deletion and update anomalies. If there are any present, then note them so that applications can be made to take them into account.
GUIDELINE 3: Relations should be designed such that their tuples will have as few NULL values as possible

- Attributes that are NULL frequently could be placed in separate relations (with the primary key)
- Reasons for nulls:
  - attribute not applicable or invalid
  - attribute value unknown (may exist)
  - value known to exist, but unavailable
Spurious Tuples

- Bad designs for a relational database may result in erroneous results for certain JOIN operations.
- The "lossless join" property is used to guarantee meaningful results for join operations.

GUIDELINE 4: The relations should be designed to satisfy the lossless join condition. No spurious tuples should be generated by doing a natural-join of any relations.
Spurious Tuples (2)

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 [1]).
Functional Dependencies (FDs)

- Definition of FD
- Direct, indirect, partial dependencies
- Inference Rules for FDs
- Equivalence of Sets of FDs
- Minimal Sets of FDs
Functional Dependencies (1)

- 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.
Functional Dependencies (2)

- 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):
  \[ \text{If } t1[X]=t2[X], \text{ then } t1[Y]=t2[Y] \]
- X -> Y in R specifies a constraint on all relation instances r(R).
- Written as X -> Y; can be displayed graphically on a relation schema as in Figures. (denoted by the arrow: ).
- FDs are derived from the real-world constraints on the attributes.
Examples of FD constraints (1)

- social security number determines employee name
  
  SSN \rightarrow \text{ENAME}

- project number determines project name and location
  
  PNUMBER \rightarrow \{\text{PNAME, PLOCATION}\}

- employee ssn and project number determines the hours per week that the employee works on the project
  
  \{\text{SSN, PNUMBER}\} \rightarrow \text{HOURS}
Examples of FD constraints (2)

- An FD is a property of the attributes in the schema R
- The constraint must hold on *every relation instance* $r(R)$
- If K is a key of R, then K functionally determines all attributes in R (since we never have two distinct tuples with $t1[K]=t2[K]$)
Functional Dependencies (3)

- 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)

```
TicketID ———> TicketName
            |        
            v        
TicketType ———> TicketLocation
```

Functional Dependencies (4)

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

Diagram:
- TicketID
- TicketName
- TicketType
- TicketLocation
- Price
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

  $$\text{SSN} \rightarrow \text{ENAME}$$

  $$\text{PNUMBER} \rightarrow \{\text{PNAME, PLOCATION}\}$$
Partial dependency
Inference Rules for FDs (1)

- 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 $Y$ is a subset of $X$, 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 $X \rightarrow Z$.

IR1, IR2, IR3 form a sound and complete set of inference rules.
Inference Rules for FDs (2)

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 (3)

- **Closure** of a set $F$ of FDs is the set $F^+$ of all FDs (include $F$) 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 $F$
Determining $X^+$

**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 ($X^+ = \text{old}X^+$);

- Example: Emp_Proj($\text{Ssn, Ename, Pnumber, Pname, Plocation, Hours}$)

$$F = \{\text{Ssn} \rightarrow \text{Ename}, \quad \text{Pnumber} \rightarrow \{\text{Pname, Plocation}\}, \quad \{\text{Ssn, Pnumber}\} \rightarrow \text{Hours}\}$$

$$\{\text{Ssn}\}^+ = \{\text{Ssn, Ename}\}$$
$$\{\text{Pnumber}\}^+ = \{\text{Pnumber, Pname, Plocation}\}$$
$$\{\text{Ssn, Pnumber}\}^+ = \{\text{Ssn, Pnumber, Ename, Pname, Plocation, Hours}\}$$
Equivalence of Sets of FDs

- Two sets of FDs F and G are equivalent if:
  - every FD in F can be inferred from G, and
  - every FD in G can be inferred from F
- Hence, F and G are equivalent if $F^+ = G^+$

**Definition**: F covers G if every FD in G can be inferred from F (i.e., if $G^+$ is a subset of $F^+$)

- F and G are equivalent if F covers G and G covers F
- There is an algorithm for checking equivalence of sets of FDs
Minimal Sets of FDs (1)

- A set of FDs is **minimal** if it satisfies the following conditions:

  1. Every dependency in F has a single attribute for its RHS.
  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.
Minimal Sets of FDs (2)

- 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 (e.g., see algorithms 11.2 and 11.4)
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 $X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow A_n$.
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 $F$. 
Algorithm for Finding a Key

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 $K := K - \{A\}$;}

Note: the algorithm determines only one key out of the possible candidate keys for $R$;
Chapter 8

Normalization for Relational Databases
Outline

- Normal Forms Based on Primary Keys
  - Normalization of Relations
  - Practical Use of Normal Forms
  - Definitions of Keys and Attributes Participating in Keys
  - First Normal Form
  - Second Normal Form
  - Third Normal Form

- General Normal Form Definitions (For Multiple Keys)

- BCNF (Boyce-Codd Normal Form)
Normalization of Relations (1)

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

- **Normal form**: Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form.
Normalization of Relations (2)

- 2NF, 3NF, BCNF based on keys and FDs of a relation schema
- 4NF based on keys, multi-valued dependencies: MVDs; 5NF based on keys, join dependencies: JDs (Chapter 11)
- Additional properties may be needed to ensure a good relational design (lossless join, dependency preservation; Chapter 11)
Practical Use of Normal Forms

- **Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties.
- The practical utility of these normal forms becomes questionable when the constraints on which they are based are **hard to understand** or to detect.
- The database designers **need not** normalize to the highest possible normal form. (usually up to 3NF, BCNF or 4NF)
- **Denormalization**: the process of storing the join of higher normal form relations as a base relation—which is in a lower normal form.
Definitions of Keys and Attributes Participating in Keys (1)

- A superkey of a relation schema $R = \{A_1, A_2, \ldots, A_n\}$ is a set of attributes $S$ subset-of $R$ with the property that no two tuples $t_1$ and $t_2$ in any legal relation state $r$ of $R$ will have $t_1[S] = t_2[S]$.

- A key $K$ is a superkey with the additional property that removal of any attribute from $K$ will cause $K$ not to be a superkey any more.
Definitions of Keys and Attributes Participating in Keys (2)

- If a relation schema has more than one key, each is called a **candidate key**. One of the candidate keys is *arbitrarily* designated to be the **primary key**, and the others are called **secondary keys**.
- A **Prime attribute** must be a member of some candidate key.
- A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any candidate key.
First Normal Form

- Disallows composite attributes, multivalued attributes, and nested relations; attributes whose values for an individual tuple are non-atomic

- Considered to be part of the definition of relation
Normalization into 1NF

**Figure 14.8** Normalization into 1NF. (a) Relation schema that is not in 1NF. (b) Example relation instance. (c) 1NF relation with redundancy.

### (a) DEPARTMENT

<table>
<thead>
<tr>
<th>DNAME</th>
<th>DNUMBER</th>
<th>DMGRSSN</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>DMGRSSN</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>DMGRSSN</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>
<tr>
<td>Ssn</td>
<td>Ename</td>
<td>Pnumber</td>
<td>Hours</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>123456789</td>
<td>Smith, John B.</td>
<td>1</td>
<td>32.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>7.5</td>
</tr>
<tr>
<td>666884444</td>
<td>Narayan, Ramesh K.</td>
<td>3</td>
<td>40.0</td>
</tr>
<tr>
<td>453453453</td>
<td>English, Joyce A.</td>
<td>1</td>
<td>20.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>20.0</td>
</tr>
<tr>
<td>333445555</td>
<td>Wong, Franklin T.</td>
<td>2</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>10.0</td>
</tr>
<tr>
<td>999887777</td>
<td>Zelaya, Alicia J.</td>
<td>30</td>
<td>30.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>10.0</td>
</tr>
<tr>
<td>987987987</td>
<td>Jabbar, Ahmad V.</td>
<td>10</td>
<td>35.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>5.0</td>
</tr>
<tr>
<td>987654321</td>
<td>Wallace, Jennifer S.</td>
<td>30</td>
<td>20.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>15.0</td>
</tr>
<tr>
<td>888665555</td>
<td>Borg, James E.</td>
<td>20</td>
<td>NULL</td>
</tr>
</tbody>
</table>

**EMP_PROJ1**

| Projs
| Ssn | Ename |

**EMP_PROJ2**

| Ssn | Pnumber | Hours |

Second Normal Form (1)

- Uses the concepts of FDs, primary key

**Definitions:**

- **Prime attribute** - attribute that is member of the primary key K

- **Full functional dependency** - a FD $Y \rightarrow Z$ where removal of any attribute from $Y$ means the FD does not hold any more

**Examples:**
- $\{\text{SSN, PNUMBER}\} \rightarrow \text{HOURS}$ is a full FD since neither $\text{SSN} \rightarrow \text{HOURS}$ nor $\text{PNUMBER} \rightarrow \text{HOURS}$ hold
- $\{\text{SSN, PNUMBER}\} \rightarrow \text{ENAME}$ is *not* a full FD (it is called a *partial dependency*) since $\text{SSN} \rightarrow \text{ENAME}$ also holds
Second Normal Form (2)

- A relation schema \( R \) is in second normal form (2NF) if every non-prime attribute \( A \) in \( R \) is fully functionally dependent on the primary key.

- \( R \) can be decomposed into 2NF relations via the process of 2NF normalization.
Normalizing into 2NF

(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>
Third Normal Form (1)

Definition:

- **Transitive functional dependency** - a FD $X \rightarrow Z$ that can be derived from two FDs $X \rightarrow Y$ and $Y \rightarrow Z$

Examples:

- SSN $\rightarrow$ DMGRSSN is a *transitive* FD since SSN $\rightarrow$ DNUMBER and DNUMBER $\rightarrow$ DMGRSSN hold
- SSN $\rightarrow$ ENAME is *non-transitive* since there is no set of attributes $X$ where SSN $\rightarrow$ $X$ and $X$ $\rightarrow$ ENAME
Third Normal Form (2)

- 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.
- $R$ can be decomposed into 3NF relations via the process of 3NF normalization.

**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# $\rightarrow$ Salary and Emp# is a candidate key.
Normalizing into 3NF
General Normal Form Definitions (For Multiple Keys) (1)

● 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 fully functionally dependent on every key of R
General Normal Form Definitions (2)

Definition:

- **Superkey** of relation schema R - a set of attributes S of R that contains a key of R
- A relation schema R is in **third normal form** (3NF) if whenever a FD X -> A holds in R, then either:
  - (a) X is a superkey of R, or
  - (b) A is a prime attribute of R

**NOTE:** Boyce-Codd normal form disallows condition (b) above
(a) LOTS

<table>
<thead>
<tr>
<th>Property_id#</th>
<th>County_name</th>
<th>Lot#</th>
<th>Area</th>
<th>Price</th>
<th>Tax_rate</th>
</tr>
</thead>
</table>

FD1
FD2
FD3
FD4

Candidate Key

(b) LOTS1

<table>
<thead>
<tr>
<th>Property_id#</th>
<th>County_name</th>
<th>Lot#</th>
<th>Area</th>
<th>Price</th>
</tr>
</thead>
</table>

FD1
FD2
FD4

Lots2

<table>
<thead>
<tr>
<th>County_name</th>
<th>Tax_rate</th>
</tr>
</thead>
</table>

FD3

(c) LOTS1A

<table>
<thead>
<tr>
<th>Property_id#</th>
<th>County_name</th>
<th>Lot#</th>
<th>Area</th>
</tr>
</thead>
</table>

FD1
FD2

Lots1B

<table>
<thead>
<tr>
<th>Area</th>
<th>Price</th>
</tr>
</thead>
</table>

FD4
BCNF (Boyce-Codd Normal Form)

- 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 \( R \).
- Each normal form is strictly stronger than the previous one:
  - Every 2NF relation is in 1NF
  - Every 3NF relation is in 2NF
  - Every BCNF relation is in 3NF
- There exist relations that are in 3NF but not in BCNF.
- The goal is to have each relation in BCNF (or 3NF).
Boyce-Codd normal form

Figure 15.13
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF.
Case Study

- Consider the relation R, which has attributes that hold schedules of courses and sections at a university.

  \[ R = \{ \text{CourseNo, SecNo, OfferingDept, CreditHours, CourseLevel, InstructorSSN, Semester, Year, Days\_Hours, RoomNo, NoOfStudents} \}. \]
Suppose that the following FDs hold on R

- `{CourseNo} -> {OfferingDept, CreditHours, CourseLevel}
- `{CourseNo, SecNo, Semester, Year} -> {Days_Hours, RoomNo, NoOfStudents, InstructorSSN}
- `{RoomNo, Days_Hours, Semester, Year} -> {InstructorSSN, CourseNo, SecNo}

Find the candidate keys of R

How would you normalize this relation?
Case study (2)

- Homework: 15.19 [1]
Chapter 10

Database Security: An Introduction
Outline

- Introduction to Database Security Issues
- Discretionary Access Control
- Mandatory Access Control
Introduction

DB security is a broad area, addressing:

- Legal and ethical issues
- Policy issues
- System-related issues
- The need to identify multiple security levels
Introduction

- Threats to databases
  - Loss of integrity
  - Loss of confidentiality
  - Loss of availability
  - Repudiation
Introduction

- Fundamental data security requirements

- Confidentiality
- Non-repudiation
- Integrity
- Availability
Introduction

- Fundamental data security requirements

- Confidentiality: Protection of data from unauthorized disclosure
- Integrity
- Availability
- Non-repudiation
Introduction

Fundamental data security requirements

- Confidentiality
- Integrity
- Availability
- Non-repudiation

Only authorized users should be allowed to modify data.
Introduction

- Fundamental data security requirements

Making data available to the authorized users & application programs
Introduction

- Fundamental data security requirements

Confidentiality

Non-repudiation

Integrity

Availability

The ability to prevent the effective denial of an act.
Countermeasures

To protect databases against these types of threats four kinds of countermeasures can be implemented:

- Access control
- Inference control
- Flow control
- Encryption
Access control

- The security mechanism of a DBMS for restricting access to the database as a whole
  - Handled by creating user accounts and passwords to control login process by the DBMS.

- Two types of database security mechanisms:
  - **Discretionary** security mechanisms (DAC)
  - **Mandatory** security mechanisms (MAC)
Inference control

- The security problem associated with databases is that of controlling the access to a statistical database, which is used to provide statistical information or summaries of values based on various criteria.

- The countermeasures to statistical database security problem is called inference control measures.
Flow control

- Flow control prevents information from flowing in such a way that it reaches unauthorized users.

- Channels that are pathways for information to flow implicitly in ways that violate the security policy of an organization are called covert channels.
Data encryption

- Data encryption is used to protect sensitive data (such as credit card numbers) that is being transmitted via some type communication network.

- The data is **encoded** using some **encoding algorithm**.
  - An unauthorized user who access encoded data will have difficulty deciphering it, but authorized users are given decoding or decrypting algorithms (or keys) to decipher data.
Database Security and the DBA

- The database administrator (DBA) is the central authority for managing a database system.
  - The DBA’s responsibilities include
    - granting privileges to users who need to use the system
    - classifying users and data in accordance with the policy of the organization
- The DBA is responsible for the overall security of the database system.
Database Security and the DBA

- The DBA has a DBA account in the DBMS
  - Sometimes these are called a system or super user account
  - These accounts provide powerful capabilities such as:
    - Account creation
    - Privilege granting
    - Privilege revocation
    - Security level assignment
  - Action 1 is access control, whereas 2 and 3 are discretionary and 4 is used to control mandatory authorization
Access Protection, User Accounts, and Database Audits

- DB security process can be summarized by the following three steps:

  1. **Identification**
     - A user presents an identity to the database

  2. **Authentication**
     - The user proves that the identity is valid

  3. **Authorization**
     - What privileges and authorizations the user has
Access Protection, User Accounts, and Database Audits

- The database system must also keep track of all operations on the database that are applied by a certain user throughout each login session.
  - To keep a record of all updates applied to the database and of the particular user who applied each update, we can modify system log, which includes an entry for each operation applied to the database that may be required for recovery from a transaction failure or system crash.
Access Protection, User Accounts, and Database Audits

- If any tampering with the database is suspected, a database audit is performed.
  - A database audit consists of reviewing the log to examine all accesses and operations applied to the database during a certain time period.

- A database log that is used mainly for security purposes is sometimes called an audit trail.
Outline

- Introduction to Database Security Issues
- Discretionary Access Control
- Mandatory Access Control
Discretionary Access Control

- The typical method of enforcing discretionary access control in a database system is based on the granting and revoking privileges.
Types of Discretionary Privileges

• The account level:
  – At this level, the DBA specifies the particular privileges that each account holds independently of the relations in the database.

• The relation level (or table level):
  – At this level, the DBA can control the privilege to access each individual relation or view in the database.
Types of Discretionary Privileges

- SQL standard supports DAC through the GRANT and REVOKE commands:
  - The GRANT command gives privileges to users
  - The REVOKE command takes away privileges
Types of Discretionary Privileges

- The privileges at the account level apply can include:
  - the CREATE SCHEMA or CREATE TABLE privilege, to create a schema or base relation;
  - the CREATE VIEW privilege;
  - the ALTER privilege, to apply schema changes such as adding or removing attributes from relations;
  - the DROP privilege, to delete relations or views;
  - the MODIFY privilege, to insert, delete, or update tuples;
  - the SELECT privilege, to retrieve information from the database by using a SELECT query.
Types of Discretionary Privileges

- The relation level of privileges applies to base relations and virtual (view) relations.
- Notice that to create a view, the account must have SELECT privilege on all relations involved in the view definition.
Types of Discretionary Privileges

- To control the granting and revoking of relation privileges, for each relation R in a database:
  - The owner of a relation is given **all** privileges on that relation.
  - The owner account holder can **pass** privileges on any of the owned relation to other users by **granting** privileges to their accounts.
  - The owner account holder can also **take back** the privileges by **revoking** privileges from their accounts.
Types of Discretionary Privileges

- In SQL the following types of privileges can be granted on each individual relation R:
  - **SELECT** (retrieval or read) privilege on R
  - **MODIFY** privileges on R
    - **UPDATE**, **DELETE**, and **INSERT** privileges
    - **INSERT** and **UPDATE** privileges can specify that only certain attributes can be updated by the account.
  - **REFERENCES** privilege on R
    - This gives the account the capability to *reference* relation R when specifying integrity constraints.
    - The privilege can also be *restricted* to specific attributes of R.
Specifying Privileges Using Views

- **The mechanism of views** is an important discretionary authorization mechanism in its own right.
  - **Column level security**
    - Owner A (of R) can create a view V of R that includes several attributes and then grant SELECT on V to B.
  - **Row level security**
    - Owner A (of R) can create a view V’ which selects several tuples from R and then grant SELECT on V’ to B.
Propagation of Privileges using the GRANT OPTION

- Whenever the owner A of a relation R grants a privilege on R to another account B, privilege can be given to B with or without the **GRANT OPTION**.
- B can also grant that privilege on R to other accounts.
- If B grants the privilege on R to C with **GRANT OPTION**
- Privileges on R can **propagate** to other accounts without the knowledge of the owner of R
Propagation of Privileges using the GRANT OPTION

- If the owner account A now revokes the privilege granted to B, all the privileges that B propagated based on that privilege should automatically be revoked by the system.
An Example

- Suppose that the DBA creates four accounts
  - A1, A2, A3, A4
- A1: Create table privilege
  
  \[
  \text{GRANT CREATE TABLE TO A1;}
  \]
- Suppose that A1 creates the two base relations \text{EMPLOYEE} and \text{DEPARTMENT}
- A1 is then owner of these two relations and hence all the relation privileges on each of them.
An Example

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dnumber</td>
</tr>
</tbody>
</table>

**Figure 23.1**
Schemas for the two relations EMPLOYEE and DEPARTMENT.
An Example

- A1 wants to grant A2 the privilege to insert and delete tuples in both of these relations, but A1 does not want A2 to be able to propagate these privileges to additional accounts:

```
GRANT INSERT, DELETE ON EMPLOYEE, DEPARTMENT TO A2;
```
An Example

- A1 wants to allow A3 to retrieve information from either of the two tables and also to be able to propagate the SELECT privilege to other accounts.

```
GRANT SELECT ON EMPLOYEE, DEPARTMENT TO A3 WITH GRANT OPTION;
```

- A3 can grant SELECT privilege to A4 to retrieve information from the Employee relation

```
GRANT SELECT ON EMPLOYEE TO A4;
```

Notice that A4 can’t propagate the SELECT privilege because GRANT OPTION was not given to A4
An Example

- A1 decides to revoke the SELECT privilege on the EMPLOYEE relation from A3

```
REVOKE SELECT ON EMPLOYEE FROM A3;
```

- The DBMS must now automatically revoke the SELECT privilege on EMPLOYEE from A4, too, because A3 granted that privilege to A4 and A3 does not have the privilege any more.
An Example

- A1 wants to give back to A3 a limited capability to SELECT from the EMPLOYEE relation and wants to allow A3 to be able to propagate the privilege.
  - The limitation is to retrieve only the NAME, BDATE, and ADDRESS attributes and only for the tuples with DNO=5.

- A1 then create the view:

  ```sql
  CREATE VIEW A3EMPLOYEE AS
  SELECT NAME, BDATE, ADDRESS
  FROM EMPLOYEE
  WHERE DNO = 5;
  ```

- And then,

  ```sql
  GRANT SELECT ON A3EMPLOYEE TO A3
  WITH GRANT OPTION;
  ```
An Example

- A1 wants to allow A4 to update only the SALARY attribute of EMPLOYEE;
- A1 can issue:

```
GRANT UPDATE ON EMPLOYEE (SALARY) TO A4;
```
DAC: Weakness

Example of a Trojan Horse

Table f1
Owner X
X: SELECT, INSERT ...
Y: NOT SELECT ON

Table f2
Owner Y
Y: SELECT, INSERT, ...
X: INSERT ON

Program P
select * from f1;
commit;
...

Read f1
Write f2
Outline

- Introduction to Database Security Issues
- Discretionary Access Control
- Mandatory Access Control
Mandatory Access Control

- Mandatory Access Control (MAC):
  - MAC applies to large amounts of information requiring strong protection in environments where both the system data and users can be classified clearly.
  - MAC is a mechanism for enforcing multiple levels of security.

Propose Model: Bell-LaPadula
Security Classes

● Classifies subjects and objects based on security classes.

● Security class:
  – Classification level
  – Category

● A subject classification reflects the degree of trust and the application area.

● A object classification reflects the sensitivity of the information.
Security Classes

• Typical classification levels are:
  – Top secret (TS)
  – Secret (S)
  – Confidential (C)
  – Unclassified (U)

Where TS is the highest level and U is the lowest: \( TS \geq S \geq C \geq U \)

• Categories tend to reflect the system areas or departments of the organization
Security Classes

● A security class is defined as follow:
  \[ SC = (A, C) \]
  
  A: classification level
  C: category

● A relation of partial order on the security classes: \( SC \leq SC' \) is verified, only if:
  \[ A \leq A' \text{ and } C' \supseteq C \]
MAC Properties

- **Simple security property:** A subject $S$ is not allowed to read or access to an object $O$ unless $\text{class}(S) \geq \text{class}(O)$.
  $\Rightarrow$ No read-up

- **Star property (or * property):** A subject $S$ is not allowed to write an object $O$ unless $\text{class}(S) \leq \text{class}(O)$
  $\Rightarrow$ No write-down
Why star property?

User Alice

Pgm X

User Bob

TH

File A

r: Alice; w:Alice

File B

r: Bob; w:Bob, Alice
Why star property?

User Alice

Pgm X

User Bob

TH

File A

File B

Read

Write

r: Alice; w: Alice

r: Bob; w: Bob, Alice

User Bob can read contents of file A copied to file B
Why star property?
Multilevel Relation

- **Multilevel relation**: MAC + relational database model
- **Data objects**: attributes and tuples
- Each attribute $A$ is associated with a **classification** attribute $C$
- A **tuple classification** attribute $TC$ is to provide a classification for each tuple as a whole, the highest of all attribute classification values.

$$R(A_1,C_1,A_2,C_2, \ldots, A_n,C_n,TC)$$
Multilevel Relation

SELECT * FROM EMPLOYEE

A user with security level S

<table>
<thead>
<tr>
<th>Name</th>
<th>Salary</th>
<th>JobPerformance</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>U</td>
<td>40000</td>
<td>C</td>
</tr>
<tr>
<td>Brown</td>
<td>C</td>
<td>80000</td>
<td>S</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Salary</th>
<th>JobPerformance</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>U</td>
<td>40000</td>
<td>C</td>
</tr>
<tr>
<td>Brown</td>
<td>C</td>
<td>80000</td>
<td>S</td>
</tr>
</tbody>
</table>
Multilevel Relation

```sql
SELECT * FROM EMPLOYEE
```

- A user with security level U

<table>
<thead>
<tr>
<th>Name</th>
<th>Salary</th>
<th>JobPerformance</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>U</td>
<td>40000</td>
<td>C</td>
</tr>
<tr>
<td>Brown</td>
<td>C</td>
<td>80000</td>
<td>S</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Salary</th>
<th>JobPerformance</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>U</td>
<td>null</td>
<td>U</td>
</tr>
<tr>
<td>Brown</td>
<td>C</td>
<td>null</td>
<td>U</td>
</tr>
</tbody>
</table>
A user with *security level C* tries to update the value of JobPerformance of Smith to ‘Excellent’:

```sql
UPDATE EMPLOYEE
SET JobPerformance = 'Excellent'
WHERE Name = 'Smith';
```
# Multilevel Relation

<table>
<thead>
<tr>
<th>Name</th>
<th>Salary</th>
<th>Job Performance</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith U</td>
<td>40000</td>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>Smith U</td>
<td>40000</td>
<td>Excellent</td>
<td>C</td>
</tr>
<tr>
<td>Brown  C</td>
<td>80000</td>
<td>Good</td>
<td>C</td>
</tr>
</tbody>
</table>

Pros and Cons of MAC

● **Pros:**
  – Provide a high degree of protection – in a way of preventing any illegal flow of information.
  – Suitable for military types of applications.

● **Cons:**
  – Not easy to apply: require a strict classification of subjects and objects into security levels.
  – Applicable for very few environments.