Theory of Relational Database Design and Normalization
(Based on Chapter 14 and some part of Chapter 15 in Fundamentals of Database Systems by Elmasri and Navathe)

1 Informal Design Guidelines for Relational Databases
   1.1 Semantics of the Relation Attributes
   1.2 Redundant Information in Tuples and Update Anomalies
   1.3 Null Values in Tuples
   1.4 Spurious Tuples
2 Functional Dependencies (FDs)
   2.1 Definition of FD
   2.2 Inference Rules for FDs
3 Normal Forms Based on Primary Keys
   3.1 Introduction to Normalization
   3.2 First Normal Form
   3.3 Second Normal Form
   3.4 Third Normal Form
4 General Normal Form Definitions (For Multiple Keys)
5 BCNF (Boyce-Codd Normal Form)
6 Designing a Set of Relations
1 Informal Design Guidelines for Relational Databases
- 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?

- 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)
1.1 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.
Figure 14.1  Simplified version of the COMPANY relational database schema.
1.2 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

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.

Insert Anomaly: Cannot insert a project unless an employee is assigned to it. 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.
Figure 14.3  Two relation schemas and their functional dependencies. Both suffer from update anomalies. (a) The EMP_DEPT relation schema. (b) The EMP_PROJ relation schema.
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.

1.3 Null Values in Tuples

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:
  a. attribute not applicable or invalid
  b. attribute value unknown (may exist)
  c. value known to exist, but unavailable

1.4 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.

- 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.
2.1 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 $Y$.

- $X \rightarrow Y$ holds if whenever two tuples have the same value for $X$, they must have the same value for $Y$.

- For any two tuples $t_1$ and $t_2$ in any relation instance $r(R)$:
  
  $If \ t_1[X]=t_2[X], then \ t_1[Y]=t_2[Y]$.

- $X \rightarrow Y$ in $R$ specifies a constraint on all relation instances $r(R)$.

- Written as $X \rightarrow Y$; can be displayed graphically on a relation schema as in Figures. (denoted by the arrow: $\rightarrow$).

- FDs are derived from the real-world constraints on the attributes.
Examples of FD constraints:
- social security number determines employee name
  \( SSN \rightarrow ENAME \)

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

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

- 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 \( t_1[K] = t_2[K] \))
2.2 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:
A1. (Reflexive) If $Y \subseteq X$, then $X \rightarrow Y$
A2. (Augmentation) If $X \rightarrow Y$, then $XZ \rightarrow YZ$
   (Notation: $XZ$ stands for $X \cup Z$)
A3. (Transitive) If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

- A1, A2, A3 form a sound and complete set of inference rules

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 A1, A2, and A3 (completeness property)
3 Normal Forms Based on Primary Keys

3.1 Introduction to Normalization

- **Normalization**: 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
- 2NF, 3NF, BCNF based on keys and FDs of a relation schema
- 4NF based on keys, multi-valued dependencies: MVDs
- Additional properties may be needed to ensure a good relational design (lossless join, dependency preservation)

3.2 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
Figure 14.8  Normalization into 1NF. (a) Relation schema that is not in 1NF. (b) Example relation instance. (c) 1NF relation with redundancy.

(a)  
```
<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)  
```
<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>Admin</td>
<td>4</td>
<td>987654321</td>
<td>[Stafford]</td>
</tr>
<tr>
<td>Head</td>
<td>1</td>
<td>888665555</td>
<td>[Houston]</td>
</tr>
</tbody>
</table>
```

(c)  
```
<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>Admin</td>
<td>4</td>
<td>987654321</td>
<td>Stafford</td>
</tr>
<tr>
<td>Head</td>
<td>1</td>
<td>888665555</td>
<td>Houston</td>
</tr>
</tbody>
</table>
```
Relational Design and Normalization

(a) $EMP_{-}PROJ$

<table>
<thead>
<tr>
<th>SSN</th>
<th>ENAME</th>
<th>PROJ1</th>
<th>PROJ2</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>Smith, John B.</td>
<td>1</td>
<td>32.5</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>333445555</td>
<td>Wong, Franklin T.</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>999887777</td>
<td>Zelaya, Alicia J.</td>
<td>30</td>
<td>30.0</td>
</tr>
<tr>
<td>987987987</td>
<td>Jabbar, Ahmad V.</td>
<td>10</td>
<td>35.0</td>
</tr>
<tr>
<td>987564321</td>
<td>Wallace, Jennifer S.</td>
<td>30</td>
<td>20.0</td>
</tr>
<tr>
<td>888665555</td>
<td>Borg, James E.</td>
<td>20</td>
<td>null</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>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>333445555</td>
<td>Wong, Franklin T.</td>
<td>2</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>987987987</td>
<td>Jabbar, Ahmad V.</td>
<td>10</td>
<td>35.0</td>
</tr>
<tr>
<td>987564321</td>
<td>Wallace, Jennifer S.</td>
<td>30</td>
<td>20.0</td>
</tr>
<tr>
<td>888665555</td>
<td>Borg, James E.</td>
<td>20</td>
<td>null</td>
</tr>
</tbody>
</table>

d) Example extension of the $EMP_{-}PROJ$ relation showing nested relations within each tuple. (e) Decomposing $EMP_{-}PROJ$ into INF

relations $EMP_{-}PROJ1$ and $EMP_{-}PROJ2$ by propagating the primary key.
3.3 Second Normal Form

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

- 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
Relational Design and Normalization

(a) EMP_PROJ

(b) EMP_DEPT

2NF NORMALIZATION

3NF NORMALIZATION

B.SRINIVAS, Associate Professor, VJIT, Hyderabad
3.4 Third Normal Form

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

- 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.
4 General Normal Form Definitions (For Multiple Keys)

- 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

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
5 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)
Figure 14.12  Boyce-Codd normal form. (a) BCNF normalization with the dependency of FD2 being “lost” in the decomposition. (b) A relation R in 3NF but not in BCNF.
Figure 14.13  A relation TEACH that is in 3NF but not in BCNF.

<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>Ahamad</td>
</tr>
<tr>
<td>Wong</td>
<td>Database</td>
<td>Omiecinski</td>
</tr>
<tr>
<td>Zelaya</td>
<td>Database</td>
<td>Navathe</td>
</tr>
</tbody>
</table>
6 DESIGNING A SET OF RELATIONS:

The approach of Relational Synthesis:

- Assumes that all possible functional dependencies are known.
- First constructs a minimal set of f.d.s
- Then applies algorithms that construct a target set of 3NF or BCNF relations.

- Additional criteria may be needed to ensure the the set of relations in a relational database are satisfactory

Goals:

- Lossless join property (a must) – tests for general losslessness.
- Dependency preservation property – decomposes a relation into BCNF components by sacrificing the dependency preservation.