Design Process - Where are we?

Conceptual Design

Conceptual Schema (ER Model)

Logical Design

Logical Schema (Relational Model)

Relational Design Principles

- Relations should have semantic unity
- Information repetition should be avoided
  - Anomalies: insertion, deletion, modification
- Avoid null values as much as possible
  - Difficulties with interpretation
    - don’t know, don’t care, known but unavailable, does not apply
  - Specification of joins
- Avoid spurious joins
### Bad Design

#### Employee Name
- **Title**:
- **Salary**:
- **Duration**:

### EMP-PROJ

<table>
<thead>
<tr>
<th>EMP-PROJ</th>
<th>ENAME</th>
<th>TITLE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>J. Doe</td>
<td>Elect. Eng.</td>
<td>40000</td>
</tr>
<tr>
<td>E2</td>
<td>M. Smith</td>
<td>Analyst</td>
<td>34000</td>
</tr>
<tr>
<td>E3</td>
<td>M. Smith</td>
<td>Mech. Eng.</td>
<td>27000</td>
</tr>
<tr>
<td>E4</td>
<td>J. Miller</td>
<td>Programmer</td>
<td>24000</td>
</tr>
<tr>
<td>E5</td>
<td>B. Casey</td>
<td>Syst. Anal.</td>
<td>34000</td>
</tr>
<tr>
<td>E6</td>
<td>L. Chu</td>
<td>Elect. Eng.</td>
<td>40000</td>
</tr>
<tr>
<td>E7</td>
<td>R. Davis</td>
<td>Mech. Eng.</td>
<td>27000</td>
</tr>
<tr>
<td>E8</td>
<td>J. Jones</td>
<td>Syst. Anal.</td>
<td>34000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PNO</th>
<th>PNAME</th>
<th>BUDGET</th>
<th>DURATION</th>
<th>RESP</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Instrumentation</td>
<td>150000</td>
<td>12</td>
<td>Manager</td>
</tr>
<tr>
<td>P2</td>
<td>Database Develop.</td>
<td>135000</td>
<td>6</td>
<td>Analyst</td>
</tr>
<tr>
<td>P3</td>
<td>CAD/CAM</td>
<td>250000</td>
<td>10</td>
<td>Consultant</td>
</tr>
<tr>
<td>P4</td>
<td>Maintenance</td>
<td>310000</td>
<td>48</td>
<td>Engineer</td>
</tr>
<tr>
<td>P5</td>
<td>Database Develop.</td>
<td>135000</td>
<td>18</td>
<td>Programmer</td>
</tr>
<tr>
<td>P6</td>
<td>Database Develop.</td>
<td>150000</td>
<td>24</td>
<td>Manager</td>
</tr>
<tr>
<td>P7</td>
<td>Database Develop.</td>
<td>310000</td>
<td>48</td>
<td>Manager</td>
</tr>
<tr>
<td>P8</td>
<td>CAD/CAM</td>
<td>250000</td>
<td>36</td>
<td>Engineer</td>
</tr>
</tbody>
</table>

#### Project Name
- **Budget**:

### Information Repetition

- The TITLE, SALARY, BUDGET attribute values are repeated for each project that the engineer is involved in.
  - Waste of space
  - Complicates updates

This example instance of EMP-PROJ relation violates one of the constraints in our earlier design. Which one?

### EMP-PROJ

<table>
<thead>
<tr>
<th>EMP-PROJ</th>
<th>ENAME</th>
<th>TITLE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>J. Doe</td>
<td>Elect. Eng.</td>
<td>40000</td>
</tr>
<tr>
<td>E2</td>
<td>M. Smith</td>
<td>Analyst</td>
<td>34000</td>
</tr>
<tr>
<td>E3</td>
<td>M. Smith</td>
<td>Mech. Eng.</td>
<td>27000</td>
</tr>
<tr>
<td>E4</td>
<td>J. Miller</td>
<td>Programmer</td>
<td>24000</td>
</tr>
<tr>
<td>E5</td>
<td>B. Casey</td>
<td>Syst. Anal.</td>
<td>34000</td>
</tr>
<tr>
<td>E6</td>
<td>L. Chu</td>
<td>Elect. Eng.</td>
<td>40000</td>
</tr>
<tr>
<td>E7</td>
<td>R. Davis</td>
<td>Mech. Eng.</td>
<td>27000</td>
</tr>
<tr>
<td>E8</td>
<td>J. Jones</td>
<td>Syst. Anal.</td>
<td>34000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PNO</th>
<th>PNAME</th>
<th>BUDGET</th>
<th>DURATION</th>
<th>RESP</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Instrumentation</td>
<td>150000</td>
<td>12</td>
<td>Manager</td>
</tr>
<tr>
<td>P2</td>
<td>Database Develop.</td>
<td>135000</td>
<td>6</td>
<td>Analyst</td>
</tr>
<tr>
<td>P3</td>
<td>CAD/CAM</td>
<td>250000</td>
<td>10</td>
<td>Consultant</td>
</tr>
<tr>
<td>P4</td>
<td>Maintenance</td>
<td>310000</td>
<td>48</td>
<td>Engineer</td>
</tr>
<tr>
<td>P5</td>
<td>Database Develop.</td>
<td>135000</td>
<td>18</td>
<td>Programmer</td>
</tr>
<tr>
<td>P6</td>
<td>Database Develop.</td>
<td>150000</td>
<td>24</td>
<td>Manager</td>
</tr>
<tr>
<td>P7</td>
<td>Database Develop.</td>
<td>310000</td>
<td>48</td>
<td>Manager</td>
</tr>
<tr>
<td>P8</td>
<td>CAD/CAM</td>
<td>250000</td>
<td>36</td>
<td>Engineer</td>
</tr>
</tbody>
</table>
Insertion Anomaly

- It is difficult (impossible?) to store information about a new project until an employee is assigned to it. Why?

Deletion Anomaly

- If an engineer, who is the only employee on a project, leaves the company, his personal information cannot be deleted, or the information about that project is lost.
- May have to delete many tuples.
Modification Anomaly

- If any attribute of project (say BUDGET of P1) is modified, all the tuples for all employees who work on that project need to be modified.

<table>
<thead>
<tr>
<th>EMP</th>
<th>PROJ</th>
<th>ENAME</th>
<th>TITLE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>P1</td>
<td>J. Doe</td>
<td>Elect. Eng.</td>
<td>40000</td>
</tr>
<tr>
<td>E2</td>
<td>P1</td>
<td>M. Smith</td>
<td>Analyst</td>
<td>34000</td>
</tr>
<tr>
<td>E3</td>
<td>P1</td>
<td>A. Lee</td>
<td>Mech. Eng.</td>
<td>27000</td>
</tr>
<tr>
<td>E4</td>
<td>P1</td>
<td>J. Miller</td>
<td>Programmer</td>
<td>24000</td>
</tr>
<tr>
<td>E5</td>
<td>P1</td>
<td>B. Casey</td>
<td>Syst. Anal.</td>
<td>34000</td>
</tr>
<tr>
<td>E6</td>
<td>P1</td>
<td>L. Chu</td>
<td>Elect. Eng.</td>
<td>40000</td>
</tr>
<tr>
<td>E7</td>
<td>P1</td>
<td>R. Davis</td>
<td>Mech. Eng.</td>
<td>27000</td>
</tr>
<tr>
<td>E8</td>
<td>P1</td>
<td>J. Jones</td>
<td>Syst. Anal.</td>
<td>34000</td>
</tr>
<tr>
<td>PNO</td>
<td>PNAME</td>
<td>BUDGET</td>
<td>DURATION</td>
<td>RESP</td>
</tr>
<tr>
<td>P1</td>
<td>Instrumentation</td>
<td>150000</td>
<td>12</td>
<td>Manager</td>
</tr>
<tr>
<td>P2</td>
<td>Database Develop.</td>
<td>135000</td>
<td>6</td>
<td>Analyst</td>
</tr>
<tr>
<td>P3</td>
<td>CAD/CAM</td>
<td>250000</td>
<td>10</td>
<td>Consultant</td>
</tr>
<tr>
<td>P4</td>
<td>Maintenance</td>
<td>310000</td>
<td>48</td>
<td>Engineer</td>
</tr>
<tr>
<td>P5</td>
<td>Database Develop.</td>
<td>135000</td>
<td>18</td>
<td>Programmer</td>
</tr>
<tr>
<td>P6</td>
<td>Database Develop.</td>
<td>135000</td>
<td>24</td>
<td>Manager</td>
</tr>
<tr>
<td>P7</td>
<td>Maintenance</td>
<td>310000</td>
<td>48</td>
<td>Engineer</td>
</tr>
<tr>
<td>P8</td>
<td>CAD/CAM</td>
<td>250000</td>
<td>40</td>
<td>Manager</td>
</tr>
</tbody>
</table>

What to do?

- Take each relation individually and “improve” it in terms of the desired characteristics
  - Normal forms
    - Atomic values (1NF)
    - Can be defined according to keys and dependencies.
    - Functional Dependencies (2NF, 3NF, BCNF)
    - Multivalued dependencies (4NF)
  - Normalization
    - Normalization is a process of concept separation which applies a top-down methodology for producing a schema by subsequent refinements and decompositions.
    - Do not combine unrelated sets of facts in one table; each relation should contain an independent set of facts.
    - Universal relation assumption
    - 1NF to 3NF; 1NF to BCNF
Normalization Issues

- How do we decompose a schema into a desirable normal form?

- What criteria should the decomposed schemas follow in order to preserve the semantics of the original schema?
  - Reconstructability: recover the original relation \(\Rightarrow\) no spurious joins
  - Lossless decomposition: no information loss
  - Dependency preservation: the constraints (i.e., dependencies) that hold on the original relation should be enforceable by means of the constraints (i.e., dependencies) defined on the decomposed relations.

- What happens to queries?
  - Processing time may increase due to joins
  - Denormalization

Normal Forms

All relations
1NF
2NF
3NF
BCNF
4NF
...
Functional Dependence

- Given relation $R$ defined over $U = \{A_1, A_2, ..., A_n\}$ where $X \subseteq U, Y \subseteq U$. If, for all pairs of tuples $t_1$ and $t_2$ in any legal instance of relation scheme $R$, $t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$, then the functional dependency $X \rightarrow Y$ holds in $R$.

- Example
  - In relation EMP-PROJ
    - $(ENO, PNO) \rightarrow (ENAME, TITLE, SALARY, DURATION, RESP)$
    - $ENO \rightarrow (ENAME, TITLE, SALARY)$
    - $PNO \rightarrow (PNAME, BUDGET)$
    - $TITLE \rightarrow SALARY$

Some Basics

- **Superkey**
  - A set of one or more attributes, which, taken collectively, allow us to identify uniquely a tuple in a relation.
  - Let $R$ be a relation scheme. A subset $K$ of $R$ is a superkey of $R$ if, in any legal relation (instance) $r$ of $R$, for all pairs $t_1$ and $t_2$ of tuples in $r$ such that $t_1[K] = t_2[K] \Rightarrow t_1 = t_2$.

- **Candidate key**
  - A superkey for which no proper subset is a superkey.

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

- Attributes
  - Prime attribute is a member of any key
  - Non-prime attribute is any attribute which is not prime

- Full functional dependency
  - A FD $X \rightarrow Y$ is a full functional dependency if $X$ is minimal, i.e., removal of any attribute $A$ from $X$ means the dependency does not hold anymore.
  - Formally - $X \rightarrow Y \text{ iff for all } A \in X, (X-\{A\}) \not\rightarrow Y$.

- Partial functional dependency
  - Formally - $X \rightarrow Y \text{ iff for some } A \in X, (X-\{A\}) \rightarrow Y$.

- Transitive dependency
  - Formally - $X \rightarrow Y$ and $Y \rightarrow Z$ and $X \rightarrow Z$ and $Y \not\rightarrow X$ and $Z \not\subseteq Y$

Normal Forms Based on FDs

1NF eliminates the relations within relations or relations as attributes of tuples.

- **First Normal Form (1NF)**
  - eliminate the partial functional dependencies of non-prime attributes to key attributes

- **Second Normal Form (2NF)**
  - eliminate the transitive functional dependencies of non-prime attributes to key attributes

- **Third Normal Form (3NF)**
  - eliminate the partial and transitive functional dependencies of prime (key) attributes to key.

- **Boyce-Codd Normal Form (BCNF)**
First Normal Form

- All attribute values are atomic
- 1NF relation cannot have an attribute value that is:
  - a set of values (set-value)
  - a tuple of values (nested relation)
- This is a standard assumption in relational DBMSs and in the rest of this section
- In object-oriented DBMSs this assumption is relaxed.

Second Normal Form

- Two possible definitions:
  - A relation $R \in 2NF$ iff all non-prime attributes in $R$ are fully functionally dependent on primary key.
  - A relation $R \in 2NF$ iff the attributes are either
    - a candidate key, or
    - fully dependent on every key.
- Partial functional dependencies cause problems.
- 2NF is only of historical importance, since it is subsumed by 3NF.
- In the example, EMP-PROJ is not 2NF, we turn it into 2NF by decomposing it:
  - EMP($ENO$, $ENAME$, TITLE, SALARY)
  - PROJ($PNO$, $PNAME$, BUDGET, MGR)
  - ASSIGN($ENO$, $PNO$, DURATION, RESP)
Third Normal Form

- Intuitively: A relation $R \in 3$NF iff
  - $R \in 2$NF (i.e., every non-prime attribute is fully functionally dependent on every key)
  - No non-prime attribute of $R$ is transitively dependent on the primary key.
- The issues is to remove the transitive dependencies

Third Normal Form

- Formally: A relation scheme $R$ defined over $U = \{A_1, A_2, \ldots, A_n\}$ is in 3NF if for all functional dependencies that hold on $R$ of the form $X \rightarrow Y$, where $X \subseteq U$ and $X \subseteq U$, at least one of the following holds:
  - $X \rightarrow Y$ is a trivial functional dependency (i.e., $Y \subseteq X$)
  - $X$ is a superkey for $R$
  - $Y$ is contained in a candidate key for $R$ (i.e., a set of prime attributes)
- The first two conditions deal with transitive dependencies.
3NF – Example

EMP

<table>
<thead>
<tr>
<th>ENO</th>
<th>ENAME</th>
<th>TITLE</th>
<th>SALARY</th>
</tr>
</thead>
</table>

fd₁

fd₂

EMP is not in 3NF because of fd₂

- TITLE → SALARY but TITLE is not a superkey and SALARY is not prime
- Problem is that ENO transitively determines SALARY (as well as directly determining it)

Solution:

PAY

| TITLE | SALARY |

fd₁

fd₂

Boyce-Codd Normal Form

- You can still have transitive dependencies in 3NF if the dependent attribute(s) are prime.
- A 1NF relation scheme R is in BCNF if for every non-trivial functional dependency X → Y, X is a superkey.
- Properties of BCNF
  - All non-prime attributes are fully dependent on every key.
  - All prime attributes are fully dependent on the keys that they do not belong to.
  - No attribute is non-trivially dependent on any set of non-prime attributes.
Boyce-Codd Normal Form

- Formally: A relation scheme $R$ defined over $U = \{A_1, A_2, \ldots, A_n\}$ is in BCNF if for all functional dependencies that hold on $R$ of the form $X \rightarrow A$, where $X \subseteq U$ and $A \subseteq U$, at least one of the following holds:
  - $X \rightarrow A$ is a trivial functional dependency
  - $X$ is a superkey for $R$
- No transitive dependencies.

BCNF – Example

- Assume the following definition of the PROJECT relation with:
  - Each employee on a project has a unique location and responsibility with respect to that project, and
  - Only one project can be found at each location
- FDs would be

\[
\begin{array}{c|c|c|c|}
\text{PROJECT} & \text{PJNO} & \text{ENO} & \text{LOCATION} & \text{RESP} \\
\hline
1 & 2 & 3 & 4 & 5
\end{array}
\]

which makes PROJECT in 3NF but not in BCNF
Inferencing over FDs

- We would like to be able to infer from a given set of FDs $F$ all implied FDs $F^+$, which is called the closure of $F$.
- Important because the 3NF and BCNF definitions refer to “all functional dependencies”.
- Example:

$$ ENO \rightarrow (ENAME, TITLE, SALARY, APT#, STREET, CITY) \Rightarrow (ENO \rightarrow ENAME) $$

- This requires a set of inference rules
  - Armstrong’s axioms
  - Additional rules

Inference Rules

- Let $X$, $Y$ and $Z$ be sets of attributes in relation scheme $R$
- Armstrong’s axioms:
  - Augmentation: $\{X \rightarrow Y\} \Rightarrow \{XZ \rightarrow YZ\}$
  - Transitivity: $\{X \rightarrow Y, Y \rightarrow Z\} \Rightarrow \{X \rightarrow Z\}$
  - Reflexivity: $W \subseteq X \Rightarrow \{X \rightarrow W\}$
- These rules are
  - Sound: do not generate any incorrect FDs – anything derived from $F$ is in $F^+$
  - Complete: given $F$ as a set of FDs, they permit us to find all of $F^+$
- Additional Rules:
  - Union: $\{X \rightarrow Y, X \rightarrow Z\} \Rightarrow \{X \rightarrow YZ\}$
  - Decomposition: $\{X \rightarrow YZ\} \Rightarrow \{X \rightarrow Y, X \rightarrow Z\}$
  - Pseudotransitivity: $\{X \rightarrow Y, WY \rightarrow Z\} \Rightarrow \{XW \rightarrow Z\}$
Why These Rules?

■ Lossless join decomposition:
  ● If \( R \) is decomposed into \( R_1, \ldots, R_n \), it should be possible to reconstruct \( R \) with no additional (spurious) tuples.
  ● If a relation scheme \( R \) is decomposed into \( R_1 \) and \( R_2 \), then at least one of the following FDs should be in \( F^+ \):
    \[
    R_1 \cap R_2 \rightarrow R_1 \\
    R_1 \cap R_2 \rightarrow R_2
    \]

■ Dependency preservation:
  ● If a relation scheme \( R \) is decomposed into \( R_1 \) and \( R_2 \), then every FD in \( F \) that holds on relation \( R \) (even the implied ones) should be guaranteed to hold whenever the projected dependencies within relations \( R_1 \) and \( R_2 \) are enforced.

Closure of a Set of FDs

■ This is most easily done by converting it to the problem of computing the closure of a set of attributes.

■ For each FD defined on the base relations, pick the attribute (or set of attributes) that appear on its left-hand-side
  ● Find their closure which gives the set of attributes that are dependent on that attribute
    \[
    \text{Theorem 1: } X \rightarrow Y \in F^+ \text{ iff } Y \subseteq \text{Compute}X^+(X, F).
    \]
    \[
    \text{Theorem 2: } X \text{ is a superkey of } R \text{ iff } \text{Compute}X^+(X, F) = R.
    \]
  ● This also gives the set of FDs that can be inferred from the original FD.
Closure of a Set of Attributes

function $ComputeX^+(X, F)$
begin
$X^+ \leftarrow X$
while there exists $Y \rightarrow Z \in F$ such that
\quad $Y \subseteq X^+$ and $Z \nsubseteq X^+$
\quad then $X^+ \leftarrow X^+ \cup Z$
return($X^+$)
end

Attribute Closure Example

- Let $F$ consist of
  - $A \rightarrow B$
  - $C \rightarrow D, E$
  - $E, G \rightarrow H$
- $ComputeX^+\{C, G\}, F$
  - Initial: $X^+ = \{C, G\}$
  - Iteration 1 ($C \rightarrow D, E$): $X^+ = \{C, G, D, E\}$
  - Iteration 2 ($E, G \rightarrow H$): $X^+ = \{C, G, D, E, H\}$
Lossless Join BCNF Decomposition

Input: Relation $R < U, F >$ /* $U$={attributes}, $F$: {FDs} */
Output: Decomposition $D$ for $R$
Step 1. $D \leftarrow \{ R \}$; /* We are talking about attributes of $R$*/
Step 2. While there is a relation schema $Q \in D$ that is not in BCNF do
   if $X \rightarrow Y$ is the FD causing violation
   then $D \leftarrow (D - Q) \cup (Q - Y) \cup (X \cup Y)$

BCNF Decomposition Example

- Consider the relation and $F$
  - EMP(ENO, ENAME, TITLE, PNO, PNAME, RESP)
  - $F = \{ ENO \rightarrow ENAME, TITLE,$
    PNO \rightarrow PNAME,
    ENO, PNO \rightarrow RESP \}$
- EMP is not in BCNF, because ENO and PNO are individually not superkeys. Thus,
  - ENO \rightarrow ENAME, TITLE
  - PNO \rightarrow PNAME
  - both cause violation of BCNF.
BCNF Decomposition Example

- We start with $D = \{\text{ENO, ENAME, TITLE, PNO, PNAME, RESP}\}$
- Iteration 1
  - Pick one of the FDs that violate BCNF
  - $\text{ENO} \rightarrow \text{ENAME, TITLE}$
  - $D = \{R_1, R_2\}$ where
    - $R_1(\text{ENO, PNO, PNAME, RESP})$
    - $R_2(\text{ENO, ENAME, TITLE})$
  - $R_2$ is in BCNF, but $R_1$ is not

BCNF Decomposition Example

- Iteration 2
  - $D$ has $R_1$ which is not in BCNF
  - Pick one of the FDs that violate BCNF
  - $\text{PNO} \rightarrow \text{PNAME}$
  - $D = \{R_2, R_3, R_4\}$ where
    - $R_3(\text{ENO, PNO, RESP})$
    - $R_4(\text{PNO, PNAME})$
  - Both relations are in BCNF
  - Therefore, replace EMP with $R_2, R_3, R_4$
Complexity of Normalization

- Assume we are given a set of attributes $A$ and a set of FDs $F$, and let $n = \text{the size of this input (at most } O(|A|*|F|))$.
  - The number of dependencies in $F^+$ may be exponential in $n$.
  - $A^+$ can be found in linear time.
  - Testing whether $X \rightarrow Y$ is in $F^+$ can be done in linear time.
  - Testing whether a decomposition is lossless can be done in linear time.
  - Testing whether a decomposition is dependency preserving can be done in polynomial time.
  - Testing whether a relation scheme is in BCNF is NP-complete.
  - There is a quadratic algorithm to find a set of relations over attributes $A$ where
    - Each is in 3NF
    - The set preserves all dependencies in $F$, and
    - The set correspond to a lossless decomposition of the universal relation covering all of $A$. 