Chapter 8

Relational Database Design
Relational Database Design: Goals

- Reduce *data redundancy* (undesirable *replication* of data values)

- Minimize *anomaly* problems (data model is structured in an improper manner)

- Maintain (correct) information

- Enforce *semantic* and *integrity* constraints, e.g., using dependency and domain constraints
Relational Database Design: Problems

- **Data replication**: extra storage, update anomalies
- **Anomaly**: costly and data inconsistency
- **Loss of information**: lossy decomposition
- **Un-enforced dependency constraints**: dependencies are lost
Integrity Constraints: Functional Dependency (FD)

- FD constraints derived from the *intra-relationships* among attributes in a relation that models the real-world enterprise
- Used to specify integrity/semantic constraints of legal relations
- **Definition:** Let $X \subseteq R$ and $Y \subseteq R$.
  - $X \rightarrow Y$, i.e., *X functionally determines Y* iff $\forall t_1, t_2 \in r(R)$, $t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$
  - A set of attributes $X$ *functionally determines* a set of attributes $Y$ if the value of $X$ **uniquely determines** the value of $Y$
  - If $X \rightarrow R$, then $X$ is a *superkey* of $R$, i.e., $\forall t_1, t_2 \in r(R)$, $t_1[X] = t_2[X] \Rightarrow t_1[R] = t_2[R]$, which holds for all instances of $R$
  - Total # of possible FDs in $R$: $2^{|R|} \times 2^{|R|}$ including $\emptyset \rightarrow R$, $R \rightarrow \emptyset$
  - **Trivial** FDs: FDs satisfied by all relations, e.g., $R \rightarrow A$ since $t_1[R] = t_2[R] \Rightarrow t_1[A] = t_2[A]$
    (general from) $X \rightarrow Y$ is **trivial** if $Y \subseteq X$
  - A *schema constraint* instead of an instance constraint
Integrity Constraints

- Inference Rules for FDs
- Determines FDs that are logically implied by a set of FDs, e.g.,

  if \( A \rightarrow B \) and \( B \rightarrow C \), then \( A \rightarrow C \), since by definition
  - if \( t_1[A] = t_2[A] \), then \( t_1[B] = t_2[B] \), and
  - if \( t_1[B] = t_2[B] \), then \( t_1[C] = t_2[C] \) implies
    - if \( t_1[A] = t_2[A] \), then \( t_1[C] = t_2[C] \)

- Closure of a set of FDs, \( F \), is the set \( F^+ \) of all FDs that can be inferred from (or implied by) \( F \)

- Given a set of FDs, \( F \), inferred FDs hold whenever the FDs in \( F \) hold, e.g., \( X \rightarrow AB \Rightarrow X \rightarrow A \) and \( X \rightarrow B \)

- Determine all the FDs of \( F^+ \): use Armstrong’s axioms
Integrity Constraints

Armstrong’s Axioms (AA). Let $\alpha$, $\beta$, $\gamma$, $\delta$ be sets of attributes

- A1. Reflexivity: If $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$
- A2. Augmentation: If $\alpha \rightarrow \beta$, then $\delta\alpha \rightarrow \delta\beta$
- A3. Transitivity: If $\alpha \rightarrow \beta$ and $\beta \rightarrow \delta$, then $\alpha \rightarrow \delta$

AA is sound: If $\alpha \rightarrow \beta$ is inferred from a set of FDs $F$ using AA, then $\alpha \rightarrow \beta$ holds in any relation in which FDs in $F$ hold

AA is complete: it generates all the FDs in $F^+$

Additional inference rules follow from AA:

- A4. Union: If $\alpha \rightarrow \beta$ and $\alpha \rightarrow \delta$, then $\alpha \rightarrow \beta\delta$
- A5. Decomposition: If $\alpha \rightarrow \beta\delta$, then $\alpha \rightarrow \beta$ and $\alpha \rightarrow \delta$
- A6. Pseudotransitivity: If $\alpha \rightarrow \beta$ and $\gamma\beta \rightarrow \delta$, then $\gamma\alpha \rightarrow \delta$
Integrity Constraints

- Closure of attribute sets
  - Closure of a set of attributes $\alpha$ with respect to $F$ is the set $\alpha^+$ of all attributes that are functionally determined by $\alpha$
  - $\alpha^+$ can be calculated by applying AA repeatedly using the FDs in $F$
  - **Algorithm**: Compute the closure of a set of attrs.
    - **Input**: A set of attributes, $\alpha$, and a set of FDs, $F$.
    - **Output**: The closure of $\alpha$, $\alpha^+$ (i.e., $\omega$).
    - **BEGIN**
      $\omega := \alpha$;
      REPEAT
        for each FD $\beta \rightarrow \gamma \in F$ do
          if $\beta \subseteq \omega$ then $\omega := \omega \cup \gamma$;
      UNTIL there are no changes to $\omega$;
    **END**;
Key Finding

Algorithm: Determine all (candidate) keys for a relation schema.

Input: A set of FDs, $F$, and a relation schema, $R$.

Output: The set of keys for $R$.

BEGIN

Remark 1. If an attribute $A$ appears only on the left hand side (L.H.S.) of the FDs in $F$, then attribute $A$ has to be in all keys because it cannot be inferred.

Remark 2. If an attribute $A$ appears only on the right hand side (R.H.S.) of the FDs in $F$, then it is highly unlikely (unless in a very special circumstance) that $A$ is an attribute in a key.

Step 1. Determine those attributes that appear only on the L.H.S. of the FDs in $F$. This set of attributes, called $X$, must be in all keys.

Step 2. Determine whether $X$ in Step 1 is a key. If so, $X$ is the only key. If not, start adding attributes (one at a time) that appear on both L.H.S. and R.H.S. of the FDs in $F$ to $X$.

Step 3. If Step 2 fails to find a key, then add attributes (one at a time) which appear only on the R.H.S. of the FDs in $F$ to $X$.

END
Relational Database Design

Normalization

- A DB schema design tool
- A process of replacing associations among attributes in a relation schema
- An approximation of the relation schemas that should be created
- Objectives: accomplish the goals of relational DB design
- 2 approaches: decomposition and synthesis
Relational Database Design

- Decomposition
  - A process to *split* or *decompose* a relation until the resultant relations no longer exhibit the undesirable problems, e.g., data redundancy, data inconsistency, anomaly, etc.
  - Decomposing a relation schema $R$ means breaking $R$ into a pair of schemas, possibly intersecting
    - this process is repeated until all the decomposed relation schemas are in the desired (normal) form.
Relational Database Design Normal Forms (NFs)

- Restrictions on the DB schema that preclude certain undesirable properties (*data redundancy, update anomaly, loss of information*, etc.) from the DB.

- A relation schema $R$ is in *PJNF* if
  
  $R$ is in *4NF* if
  
  $R$ is in *BCNF* if
  
  $R$ is in *3NF* if
  
  $R$ is in *2NF* if
  
  $R$ is in *1NF*
Relational Database Design Normal Forms (NFs)

- **Definition.** A data value $v$ is **atomic** if $v$ is *not* (i) a set of values or (ii) composite value; otherwise, $v$ is **non-atomic**.

- **First Normal Form (1NF).** A relation schema $R$ is in **1NF** if for every attribute $A$ in $R$, the values in the domain of $A$, i.e., $\text{dom}(A)$, are **atomic**.

- **Boyce-Codd Normal Form (BCNF).** A relation schema $R$ is in **BCNF** if for every non-trivial FD $X \rightarrow Y$ applied to $R$, $X$ is a superkey for $R$.

- **Definition.** Let $A$ be an attribute in a relation schema $R$, and let $F$ be a set of FDs over $R$. $A$ is a **prime** attribute in $R$ if $A$ is contained in some candidate key of $R$; otherwise, $A$ is a **non-prime** attribute in $R$. 
Relational Database Design Normal Forms (NFs)

- **Third Normal Form (3NF).** A relation schema $R$ is in 3NF if
  1. $R$ is in 1NF, and
  2. For every non-trivial FD $X \rightarrow Y$ applied to $R$, either
     - $X$ is a superkey for $R$, or
     - every attribute in $Y$ is an attribute of some candidate key for $R$, i.e., prime.

- **Lossy decomposition:** a decomposition is *lossy* if the natural join of all the decomposed relations contain *additional* tuples and the original relation is lost.

- **Lossless decomposition:** a decomposition is *lossless* if the natural join of all the decomposed relations always yields the original relation without any extra tuples, i.e.,

  a decomposition $\{R_1, R_2, \ldots, R_n\}$ of $R$ is lossless if $\forall r(R)$,
  
  $$r(R) = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \ldots \bowtie \pi_{R_n}(r)$$
Relational Database Design

- **2-Relational lossless-join decomposition:**

  Let $R_1$ and $R_2$ be decomposed schemas of $R$. Let $F$ be a set of FDs on $R$. The decomposition is *lossless* if

  $$R_1 \cap R_2 \rightarrow R_1 \in F^+ \text{ or } R_1 \cap R_2 \rightarrow R_2 \in F^+$$

- **Dependency preserving:** a decomposition is *dependency preserving* if no dependency is lost in the process.

Let $F$ be the set of FDs on $R$.

Let $R_1, \ldots, R_n$ be a decomposition of $R$.

Let $F_i$, $1 \leq i \leq n$, be the set of FDs in $F^+$ which applies to $R_i$.

Let $F' = \bigcup_{i=1}^{n} F_i$.

If $F'^+ = F^+$, then the decomposition is *dependency preserving*. 