Chapter 16
Relational Database Design Algorithms

Database Design Approaches

- Top-Down design (Starting with conceptual design)
- Bottom-Up Design (relational synthesis)

Top-Down Design

- Design conceptual schema in a high-level data model
- Map the conceptual schema into a set of relations
- Normalize the relations
Relational Synthesis

- Assumes that all possible functional dependencies are known.
- First construct a minimal set of FDs
- Then apply algorithms that construct a target set of 3NF or BCNF relations.
- Additional criteria may be needed to ensure that the set of relations in a relational database are satisfactory.

Designing a Set of Relations

Goals:
- Lossless join property (a must)
- Dependency preservation property – occasionally must be sacrificed to achieve 3NF or BCNF
- Additional normal forms
  - 4NF (based on multi-valued dependencies)
  - 5NF (based on join dependencies)

General Approach

- Construct a Universal relation containing ALL the attributes of the database
- Iteratively decompose the above relation into smaller relations based on the functional dependencies
Properties of Relational Decompositions

Relation Decomposition and Insufficiency of Normal Forms:
- Universal Relation Schema: a relation schema \( R = \{A_1, A_2, ..., A_n\} \) that includes all the attributes of the database.
- Universal relation assumption: every attribute name is unique.
- Decomposition: The process of decomposing the universal relation schema \( R \) into a set of relation schemas \( D = \{R_1, R_2, ..., R_m\} \) that will become the relational database schema by using the functional dependencies.

Properties of Relational Decompositions
- Attribute preservation condition: Each attribute in \( R \) will appear in at least one relation schema \( R_i \) in the decomposition so that no attributes are "lost".
- Another goal of decomposition is to have each individual relation \( R_i \) in the decomposition \( D \) be in BCNF or 3NF.
- Additional properties of decomposition are needed to prevent the generation of spurious tuples.

Properties of Relational Decompositions

Dependency Preservation Property of a Decomposition:
- Dependencies \( X \rightarrow Y \) in the universal relation \( R \) should either appear directly in some relation \( R_i \) or can be inferred from the dependencies in \( R_i \).
- It is not necessary that all dependencies from the relation \( R \) appear in some relation \( R_i \). It is sufficient that the union of the dependencies on all the relations \( R_i \) be equivalent to the dependencies on \( R \).
Properties of Relational Decompositions

- **Dependency Preservation Property**: A decomposition \( D = \{ R_1, R_2, ..., R_m \} \) of \( R \) is dependency-preserving with respect to \( F \) if the union of the projections of \( F \) on each \( R \) in \( D \) is equivalent to \( F \); that is, \( \bigcup (\pi_{R_i}(F) \cup \ldots \cup \pi_{R_m}(F)) = F \).

  **Claim 1**: It is always possible to find a dependency-preserving decomposition \( D \) with respect to \( F \) such that each relation \( R_i \) in \( D \) is in 3NF.

Properties of Relational Decompositions

- **Lossless (Non-additive) Join Property of a Decomposition**:

  A decomposition is lossless if the result of joining (natural join) all the decomposed relations together contains all the tuples from the original relation and does not contain any other tuples.

Properties of Relational Decompositions

- **Definition**: A decomposition \( D = \{ R_1, R_2, ..., R_m \} \) of \( R \) has the lossless (nonadditive) join property with respect to the set of dependencies \( F \) on \( R \) if, for every relation state \( r \) of \( R \) that satisfies \( F \), the following holds, where \( * \) is the natural join of all the relations in \( D \):

  \( \pi_{R_1}(r), ..., \pi_{R_m}(r) = r \)

  **Note**: The word loss in lossless refers to loss of information, not to loss of tuples. In fact, for “loss of information” a better term is “addition of spurious information.”
Properties of Relational Decompositions

Successive Lossless Join Decomposition:

- Claim 2 (Preservation of non-additivity in successive decompositions):
  If a decomposition $D = (R_1, R_2, ..., R_n)$ of $R$ has the lossless (non-additive) join property with respect to a set of functional dependencies $F$ on $R$, and if a decomposition $D_1 = (G_1, G_2, ..., G_m)$ of $R$ has the lossless (non-additive) join property with respect to the projection of $F$ on $R$, then the decomposition $D_2 = (R_1', R_2', ..., R_n')$ of $R$ has the non-additive join property with respect to $F$.

Algorithms for Relational Database Schema Design

Relational Synthesis into 3NF with Dependency Preservation
(Regressive Synthesis Algorithm)

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

1. Find a minimal cover $G$ for $F$.
2. For each left-hand-side $X$ of a functional dependency that appears in $G$, create a relation schema in $D$ with attributes $(X \cup \{A_1\} \cup \{A_2\} \cup ... \cup \{A_k\})$, where $X \rightarrow A_1, X \rightarrow A_2, ...$, $X$ 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.

Claim 3: Every relation schema created by the Relational Synthesis Algorithm is in 3NF.

Relational Synthesis into 3NF with Dependency Preservation and Lossless (Non-Additive) 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 $F$.
2. For each left-hand-side $X$ of a functional dependency that appears in $G$, create a relation schema in $D$ with attributes $(X \cup \{A_1\} \cup \{A_2\} \cup ... \cup \{A_k\})$, where $X \rightarrow A_1, X \rightarrow A_2, ...$, $X$ 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 $R$.
Finding a Key $K$ for $R$ Given a set $F$ of Functional Dependencies

**Input:** A universal 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 $K := K - \{A\}$.

Relational Decomposition into BCNF with Lossless (non-additive) 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 $(X \cup Y)$;
};
Issues with null-value joins. (a) Some EMPLOYEE tuples have null for the join attribute DNUM.
Algorithms for Relational Database Schema Design

Issues with null-value joins. (b) Result of applying NATURAL JOIN to the EMPLOYEE and DEPARTMENT relations. (c) Result of applying LEFT OUTER JOIN to EMPLOYEE and DEPARTMENT.

(b)

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th>DEPARTMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>ADDRESS</td>
</tr>
<tr>
<td>John Smith</td>
<td>731 Fosten, Houston, TX</td>
</tr>
<tr>
<td>Jane Doe</td>
<td>3201 Main, Houston, TX</td>
</tr>
</tbody>
</table>

Result of applying NATURAL JOIN to the EMPLOYEE and DEPARTMENT relations.

Result of applying LEFT OUTER JOIN to EMPLOYEE and DEPARTMENT.

Algorithms for Relational Database Schema Design

The "dangling tuple" problem. (a) The relation EMPLOYEE_1 (includes all attributes of EMPLOYEE from figure 11.2a except DNUM).

(a)

<table>
<thead>
<tr>
<th>EMPLOYEE_1</th>
<th>DNAME</th>
<th>SSN</th>
<th>DATE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith, John</td>
<td>123456789</td>
<td>1980-01-01</td>
<td>1923-04-05</td>
<td>731 Fosten, Houston, TX</td>
</tr>
<tr>
<td>Wong, Franklin T.</td>
<td>320456789</td>
<td>1900-12-23</td>
<td>1923-04-05</td>
<td>3201 Main, Houston, TX</td>
</tr>
<tr>
<td>Zhang, Alice J.</td>
<td>987654321</td>
<td>1950-07-10</td>
<td>1960-06-30</td>
<td>3201 Main, Houston, TX</td>
</tr>
<tr>
<td>Johnson, Jennifer</td>
<td>867543210</td>
<td>1900-06-30</td>
<td>1955-03-20</td>
<td>3201 Main, Houston, TX</td>
</tr>
</tbody>
</table>

The relation EMPLOYEE_2 (includes DNUM attribute with null values).

(b)

<table>
<thead>
<tr>
<th>EMPLOYEE_2</th>
<th>DNAME</th>
<th>SSN</th>
<th>DNUM</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith, John</td>
<td>123456789</td>
<td>1980-01-01</td>
<td>null</td>
<td>731 Fosten, Houston, TX</td>
</tr>
<tr>
<td>Zhang, Alice J.</td>
<td>987654321</td>
<td>1950-07-10</td>
<td>null</td>
<td>3201 Main, Houston, TX</td>
</tr>
<tr>
<td>Johnson, Jennifer</td>
<td>867543210</td>
<td>1900-06-30</td>
<td>null</td>
<td>3201 Main, Houston, TX</td>
</tr>
</tbody>
</table>

The relation EMPLOYEE_3 (includes DNUM attribute but does not include tuples for which DNUM has null values).

(c)

<table>
<thead>
<tr>
<th>EMPLOYEE_3</th>
<th>DNAME</th>
<th>SSN</th>
<th>DNUM</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith, John</td>
<td>123456789</td>
<td>1980-01-01</td>
<td>5</td>
<td>731 Fosten, Houston, TX</td>
</tr>
<tr>
<td>Zhang, Alice J.</td>
<td>987654321</td>
<td>1950-07-10</td>
<td>4</td>
<td>3201 Main, Houston, TX</td>
</tr>
<tr>
<td>Johnson, Jennifer</td>
<td>867543210</td>
<td>1900-06-30</td>
<td>4</td>
<td>3201 Main, Houston, TX</td>
</tr>
</tbody>
</table>

Algorithms for Relational Database Schema Design

The "dangling tuple" problem. (b) The relation EMPLOYEE_2 (includes DNUM attribute with null values).

(b)

<table>
<thead>
<tr>
<th>EMPLOYEE_2</th>
<th>SSN</th>
<th>DNUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>320456789</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>987654321</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>567890123</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>987654321</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>567890123</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>987654321</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>567890123</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

The relation EMPLOYEE_3 (includes DNUM attribute but does not include tuples for which DNUM has null values).

(c)

<table>
<thead>
<tr>
<th>EMPLOYEE_3</th>
<th>SSN</th>
<th>DNUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>320456789</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>987654321</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>567890123</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>987654321</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>567890123</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>987654321</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>567890123</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
Problems with Normalization Algorithms

- The database designer must first specify all the relevant functional dependencies among the database attributes.
- These algorithms are not deterministic in general.
- It is not always possible to find a decomposition into relational schemas that preserves dependencies and allows each relation schema in the decomposition to be in BCNF.

Exercise

- Apply the key finding algorithm to the following relation to determine a key. Apply the decomposition and synthesis algorithms to determine the equivalent BCNF and 3NF decompositions:

Consider the relation R, which has attributes that hold schedules of courses and sections at a university: R = (CourseNo, SecNo, OfferingDept, CreditHours, CourseLevel, InstructorId, Semester, Year, Day, Hour, RoomNo, NoOfStudents). Suppose that the following functional dependencies hold on R:

- (CourseNo, SecNo) → OfferingDept, CreditHours, CourseLevel
- (CourseNo, SecNo, Semester, Year) → (Day, Hour, RoomNo, NoOfStudents, InstructorId)
- (RoomNo, Day, Hour, Semester, Year) → (InstructorId, CourseNo, SecNo)