DATABASE DESIGN -2

DESIGNING A SET OF RELATIONS

The Approach of Relational Synthesis (Bottom-up Design):

Assumes that all possible functional dependencies are known.

First constructs a minimal set of FDs

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 (see Algorithms 11.2 and 11.4).

Goals:

Lossless join property (a must)

Algorithm 11.1 tests for general losslessness.

Dependency preservation property

Algorithm 11.3 decomposes a relation into BCNF components by sacrificing the dependency preservation.

Additional normal forms

4NF (based on multi-valued dependencies)

5NF (based on join dependencies)

Properties of Relational Decompositions

Relation Decomposition and Insufficiency of Normal Forms:

Universal Relation Schema:

A relation schema \( R = \{ A_1, A_2, \ldots, A_n \} \) 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 \( D = \{R_1, R_2, \ldots, R_m\} \) that will become the relational database schema by using the functional dependencies.

**Attribute preservation condition:**

Each attribute in R will appear in at least one relation schema Ri in the decomposition so that no attributes are “lost”.

Another goal of decomposition is to have each individual relation Ri in the decomposition D be in BCNF or 3NF.

Additional properties of decomposition are needed to prevent from generating spurious tuples.

**Dependency Preservation Property of a Decomposition:**

Definition: Given a set of dependencies F on R, the projection of F on Ri, denoted by \( p_{R_i}(F) \) where Ri is a subset of R, is the set of dependencies \( X \rightarrow Y \) in \( F^+ \) such that the attributes in \( X \cup Y \) are all contained in Ri.

Hence, the projection of F on each relation schema Ri in the decomposition D is the set of functional dependencies in \( F^+ \), the closure of F, such that all their left- and right-hand-side attributes are in Ri.

**Dependency Preservation Property:**

A decomposition \( D = \{R_1, R_2, \ldots, R_m\} \) of R is dependency-preserving with respect to F if the union of the projections of F on each Ri in D is equivalent to F; that is
Claim 1:

It is always possible to find a dependency preserving decomposition D with respect to F such that each relation Ri in D is in 3nf.

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

Definition: Lossless join property: a decomposition \( D = \{R1, R2, \ldots, Rm\} \) 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:

\[
* (\pi R1(r), \ldots, \pi Rm(r)) = r
\]

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”.

**Algorithm 11.1: Testing for Lossless Join Property**

Input: A universal relation R, a decomposition \( D = \{R1, R2, \ldots, Rm\} \) of R, and a set \( F \) of functional dependencies.

1. Create an initial matrix \( S \) with one row \( i \) for each relation \( Ri \) in \( D \), and one column \( j \) for each attribute \( Aj \) in R.

2. Set \( S(i,j) := bij \) for all matrix entries. (\( * \) each \( bij \) is a distinct symbol associated with indices \( (i,j) \)).

3. For each row \( i \) representing relation schema \( Ri \) \{for each column \( j \) representing attribute \( Aj \)
{if (relation Ri includes attribute Aj) then set S(i,j) := aj;};
(* each aj is a distinct symbol associated with index (j) *)

4. Repeat the following loop until a complete loop execution results in no changes to S

{for each functional dependency X→Y in F

{for all rows in S which have the same symbols in the columns corresponding to attributes in X

{make the symbols in each column that correspond to an attribute in Y be the same in all these rows as follows:

If any of the rows has an “a” symbol for the column, set the other rows to that same “a” symbol in the column.

If no “a” symbol exists for the attribute in any of the rows, choose one of the “b” symbols that appear in one of the rows for the attribute and set the other rows to that same “b” symbol in the column ;}; }; };

5. If a row is made up entirely of “a” symbols, then the decomposition has the lossless join property; otherwise it does not.

**Lossless (nonadditive) join test for n-ary decompositions.**

(a) Case 1: Decomposition of EMP_PROJ into EMP_PROJ1 and EMP_LOCS fails test.

(b) A decomposition of EMP_PROJ that has the lossless join property.
Lossless (nonadditive) join test for n-ary decompositions.

(c) Case 2: Decomposition of EMP_PROJ into EMP, PROJECT, and WORKS_ON satisfies test
Testing Binary Decompositions for Lossless Join Property

**Binary Decomposition:** Decomposition of a relation R into two relations.

PROPERTY LJ1 (lossless join test for binary decompositions): A decomposition D = \{R1, R2\} of R has the lossless join property with respect to a set of functional dependencies F on R if and only if either

The f.d. \((R1 \cap R2) \rightarrow (R1- R2)\) is in F+, or

The f.d. \((R1 \cap R2) \rightarrow (R2 - R1)\) is in F+.

**Successive Lossless Join Decomposition:**

Claim 2 (Preservation of non-additivity in successive decompositions):

If a decomposition D = \{R1, R2, ..., Rm\} of R has the lossless (non-additive) join property with respect to a set of functional dependencies F on R, and if a decomposition Di = \{Q1, Q2, ..., Qk\} of Ri has the lossless (non-additive) join property with respect to the projection of F on Ri, then the decomposition D2 = \{R1, R2, ..., Ri-1, Q1, Q2, ..., Qk, Ri+1, ..., Rm\} of R has the non-additive join property with respect to F

2. Algorithms for Relational Database Schema Design

**Algorithm 11.2: Relational Synthesis into 3NF with Dependency**

Preservation (Relational 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 (use Algorithm 10.2);

2. For each left-hand-side X of a functional dependency that appears in G, create a relation schema in D with attributes \(X \cup \{A1\} \cup \{A2\} \ldots\)
υ {Ak}, where X → A1, X → A2, ..., X Ak are the only dependencies in
G with X as left-hand-side (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 Algorithm 11.2 is in 3NF.

Algorithm 11.3: 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 → Y in Q that violates BCNF;

replace Q in D by two relation schemas (Q - Y) and (X υ Y);

}; Assumption: No null values are allowed for the join attributes.

Algorithm 11.4 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 (Use Algorithm 10.2).
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 \{A1\} \cup \{A2\} ...
\cup \{Ak\}}, where X \rightarrow A1, X \rightarrow A2, ..., X \rightarrow Ak are the only dependencies in
G with X as left-hand-side (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. (Use
Algorithm 11.4a to find the key of R)

**Algorithm 11.4a 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\};

}
### Database Management System

### (a) EMPLOYEE

<table>
<thead>
<tr>
<th>Ename, John B.</th>
<th>Ssn</th>
<th>Edate</th>
<th>Address</th>
<th>Drum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>123456789</td>
<td>1965-01-09</td>
<td>731 Fondren, Houston, TX</td>
<td>5</td>
</tr>
<tr>
<td>Wong, Franklin T.</td>
<td>333445555</td>
<td>1955-12-08</td>
<td>638 Vose, Houston, TX</td>
<td>5</td>
</tr>
<tr>
<td>Zelaya, Alicia J.</td>
<td>999887777</td>
<td>1968-07-19</td>
<td>3321 Castle, Spring, TX</td>
<td>4</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>
</tr>
<tr>
<td>Narayan, Ramesh K.</td>
<td>665664444</td>
<td>1962-09-15</td>
<td>975 Fire Oak, Humble, TX</td>
<td>5</td>
</tr>
<tr>
<td>English, Joyce A.</td>
<td>453453453</td>
<td>1972-07-31</td>
<td>5031 Rice, Houston, TX</td>
<td>5</td>
</tr>
<tr>
<td>Jabbar, Ahmad V.</td>
<td>087087087</td>
<td>1960-03-20</td>
<td>980 Dallas, Houston, TX</td>
<td>4</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>
</tr>
<tr>
<td>Berger, Anders C.</td>
<td>999775555</td>
<td>1965-04-26</td>
<td>6530 Braes, Bellaire, TX</td>
<td>NULL</td>
</tr>
<tr>
<td>Bentz, Carlos M.</td>
<td>888666444</td>
<td>1963-01-09</td>
<td>7554 Beech, Houston, TX</td>
<td>NULL</td>
</tr>
</tbody>
</table>

### DEPARTMENT

<table>
<thead>
<tr>
<th>Dname</th>
<th>Drum</th>
<th>Dmgr_ssn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research</td>
<td>5</td>
<td>333445555</td>
</tr>
<tr>
<td>Administration</td>
<td>4</td>
<td>987654321</td>
</tr>
<tr>
<td>Headquarters</td>
<td>1</td>
<td>888665555</td>
</tr>
</tbody>
</table>

### (b) EMPLOYEE JOIN DEPARTMENT

<table>
<thead>
<tr>
<th>Ename, John B.</th>
<th>Ssn</th>
<th>Edate</th>
<th>Address</th>
<th>Drum</th>
<th>Dname</th>
<th>Dmgr_ssn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</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 Vose, Houston, TX</td>
<td>5</td>
<td>Research</td>
<td>333445555</td>
</tr>
<tr>
<td>Zelaya, Alicia J.</td>
<td>999887777</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>665664444</td>
<td>1962-09-15</td>
<td>975 Fire Oak, 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>5031 Rice, Houston, TX</td>
<td>5</td>
<td>Research</td>
<td>333445555</td>
</tr>
<tr>
<td>Jabbar, Ahmad V.</td>
<td>087087087</td>
<td>1960-03-20</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>

### (c) EMPLOYEE JOIN DEPARTMENT

<table>
<thead>
<tr>
<th>Ename, John B.</th>
<th>Ssn</th>
<th>Edate</th>
<th>Address</th>
<th>Drum</th>
<th>Dname</th>
<th>Dmgr_ssn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</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 Vose, Houston, TX</td>
<td>5</td>
<td>Research</td>
<td>333445555</td>
</tr>
<tr>
<td>Zelaya, Alicia J.</td>
<td>999887777</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>665664444</td>
<td>1962-09-15</td>
<td>975 Fire Oak, 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>5031 Rice, Houston, TX</td>
<td>5</td>
<td>Research</td>
<td>333445555</td>
</tr>
<tr>
<td>Jabbar, Ahmad V.</td>
<td>087087087</td>
<td>1960-03-20</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>
<tr>
<td>Berger, Anders C.</td>
<td>999775555</td>
<td>1965-04-26</td>
<td>6530 Braes, Bellaire, TX</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>Bentz, Carlos M.</td>
<td>888666555</td>
<td>1963-01-09</td>
<td>7554 Beech, Houston, TX</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
</tr>
</tbody>
</table>
Figure 11.3
The dangling tuple problem.
(a) The relation EMPLOYEE_1 (includes all attributes of EMPLOYEE from Figure 11.2(a) except Dnum).
(b) The relation EMPLOYEE_2 (includes Dnum attribute with NULL values).
(c) The relation EMPLOYEE_3 (includes Dnum attribute but does not include tuples for which Dnum has NULL values).

(a) EMPLOYEE_1

<table>
<thead>
<tr>
<th>Name</th>
<th>Ssn</th>
<th>Bdate</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith, John B.</td>
<td>123456789</td>
<td>1965-01-09</td>
<td>731 Fondren, Houston, TX</td>
</tr>
<tr>
<td>Wong, Franklin T.</td>
<td>333445555</td>
<td>1955-12-08</td>
<td>638 Voss, Houston, TX</td>
</tr>
<tr>
<td>Zelaya, Alicia J.</td>
<td>999887777</td>
<td>1968-07-19</td>
<td>3321 Castle, Spring, TX</td>
</tr>
<tr>
<td>Wallace, Jennifer S.</td>
<td>987854321</td>
<td>1941-08-20</td>
<td>291 Berry, Bellaire, TX</td>
</tr>
<tr>
<td>Narayen, Ramesh K.</td>
<td>666884444</td>
<td>1982-09-15</td>
<td>975 Fire Oak, Humble, TX</td>
</tr>
<tr>
<td>English, Joyce A.</td>
<td>453453453</td>
<td>1972-07-31</td>
<td>6631 Rice, Houston, TX</td>
</tr>
<tr>
<td>Jabbar, Ahmad V.</td>
<td>987987987</td>
<td>1969-03-29</td>
<td>980 Dallas, Houston, TX</td>
</tr>
<tr>
<td>Borg, James E.</td>
<td>888665555</td>
<td>1957-11-10</td>
<td>450 Stone, Houston, TX</td>
</tr>
<tr>
<td>Berger, Anders C.</td>
<td>999775555</td>
<td>1965-04-26</td>
<td>6530 Braes, Bellaire, TX</td>
</tr>
<tr>
<td>Benitez, Carlos M.</td>
<td>888665555</td>
<td>1963-01-09</td>
<td>7654 Beech, Houston, TX</td>
</tr>
</tbody>
</table>

(b) EMPLOYEE_2

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Dnum</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>5</td>
</tr>
<tr>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>999887777</td>
<td>4</td>
</tr>
<tr>
<td>987654321</td>
<td>4</td>
</tr>
<tr>
<td>666884444</td>
<td>5</td>
</tr>
<tr>
<td>453453453</td>
<td>5</td>
</tr>
<tr>
<td>987987987</td>
<td>4</td>
</tr>
<tr>
<td>888665555</td>
<td>1</td>
</tr>
<tr>
<td>999775555</td>
<td>NULL</td>
</tr>
<tr>
<td>888664444</td>
<td>NULL</td>
</tr>
</tbody>
</table>

(c) EMPLOYEE_3

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Dnum</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>5</td>
</tr>
<tr>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>999887777</td>
<td>4</td>
</tr>
<tr>
<td>987654321</td>
<td>4</td>
</tr>
<tr>
<td>666884444</td>
<td>5</td>
</tr>
<tr>
<td>453453453</td>
<td>5</td>
</tr>
<tr>
<td>987987987</td>
<td>4</td>
</tr>
<tr>
<td>888665555</td>
<td>1</td>
</tr>
</tbody>
</table>
Discussion of Normalization Algorithms:

Problems:

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 relation schemas that preserves dependencies and allows each relation schema in the decomposition to be in BCNF (instead of 3NF as in Algorithm 11.4).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Input</th>
<th>Output</th>
<th>Properties/Purpose</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td>A decomposition $D$ of $R$ and a set $F$ of functional dependencies</td>
<td>Boolean result: yes or no for nonadditive join property</td>
<td>Testing for nonadditive join decomposition</td>
<td>See a simpler test in Section 11.1.4 for binary decompositions</td>
</tr>
<tr>
<td>11.2</td>
<td>Set of functional dependencies $F$</td>
<td>A set of relations in 3NF</td>
<td>Dependency preservation</td>
<td>No guarantee of satisfying lossless join property</td>
</tr>
<tr>
<td>11.3</td>
<td>Set of functional dependencies $F$</td>
<td>A set of relations in BCNF</td>
<td>Nonadditive join decomposition</td>
<td>No guarantee of dependency preservation</td>
</tr>
<tr>
<td>11.4</td>
<td>Set of functional dependencies $F$</td>
<td>A set of relations in 3NF</td>
<td>Nonadditive join and dependency-preserving decomposition</td>
<td>May not achieve BCNF, but achieves all desirable properties and 3NF</td>
</tr>
<tr>
<td>11.4a</td>
<td>Relation schema $R$ with a set of functional dependencies $F$</td>
<td>Key K of $R$ (that is a subset of $R$)</td>
<td>To find a key K</td>
<td>The entire relation $R$ is always a default superkey</td>
</tr>
</tbody>
</table>
Multi-valued Dependencies and Fourth Normal Form

(a) The EMP relation with two MVDs: ENAME —>> PNAME and ENAME —>> DNAME.

(b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.

(c) The relation SUPPLY with no MVDs is in 4NF but not in 5NF if it has the JD(R1, R2, R3).

(d) Decomposing the relation SUPPLY into the 5NF relations R1, R2, and R3.
Multi-valued Dependencies and Fourth Normal Form

Definition:

A multi-valued dependency (MVD) \( X \rightarrow\rightarrow Y \) specified on relation schema \( R \), where \( X \) and \( Y \) are both subsets of \( R \), specifies the following constraint on any relation state \( r \) of \( R \): If two tuples \( t_1 \) and \( t_2 \) exist in \( r \) such that \( t_1[X] = t_2[X] \), then two tuples \( t_3 \) and \( t_4 \) should also exist in \( r \) with the following properties, where we use \( Z \) to denote

\[(R - (X \cup Y)):\]

\[t_3[X] = t_4[X] = t_1[X] = t_2[X].\]

\[t_3[Y] = t_1[Y] \text{ and } t_4[Y] = t_2[Y].\]

\[t_3[Z] = t_2[Z] \text{ and } t_4[Z] = t_1[Z].\]

An MVD \( X \rightarrow\rightarrow Y \) in \( R \) is called a trivial MVD if (a) \( Y \) is a subset of \( X \), or (b) \( X \cup Y = R \).

Inference Rules for Functional and Multi-valued Dependencies:

IR1 (reflexive rule for FDs): If \( X \supseteq Y \), then \( X \rightarrow Y \).
IR2 (augmentation rule for FDs): \{X \rightarrow Y\} \sqsupseteq XZ \rightarrow YZ.

IR3 (transitive rule for FDs): \{X \rightarrow Y, Y \rightarrow Z\} \sqsupseteq X \rightarrow Z.

IR4 (complementation rule for MVDs): \{X \rightarrow\rightarrow Y\} \sqsupseteq X \rightarrow\rightarrow (R - (X \cup Y)).

IR5 (augmentation rule for MVDs): If X \rightarrow\rightarrow Y and W \supseteq Z then WX \rightarrow\rightarrow YZ.

IR6 (transitive rule for MVDs): \{X \rightarrow\rightarrow Y, Y \rightarrow\rightarrow Z\} \cdot = X \rightarrow\rightarrow (Z \cup Y).

IR7 (replication rule for FD to MVD): \{X \rightarrow Y\} \cdot = X \rightarrow\rightarrow Y.

IR8 (coalescence rule for FDs and MVDs): If X \rightarrow\rightarrow Y and there exists W with the properties that

(a) W \cap Y is empty, (b) W \rightarrow Z, and (c) Y \supseteq Z, then X \rightarrow Z.

Definition:

A relation schema R is in 4NF with respect to a set of dependencies F (that includes functional dependencies and multivalued dependencies) if, for every nontrivial multivalued dependency X \rightarrow\rightarrow Y in F+, X is a superkey for R.

Note: F+ is the (complete) set of all dependencies (functional or multivalued) that will hold in every relation state r of R that satisfies F. It is also called the closure of F.

Decomposing a relation state of EMP that is not in 4NF:

(a) EMP relation with additional tuples.

(b) Two corresponding 4NF relations EMP_PROJETS and EMP_DEPENDENTS.
Lossless (Non-additive) Join Decomposition into 4NF Relations:

PROPERTY LJ1’

The relation schemas R1 and R2 form a lossless (non-additive) join decomposition of R with respect to a set F of functional and multi-valued dependencies if and only if

\[(R1 \cap R2) \rightarrow\rightarrow (R1 - R2)\]

or by symmetry, if and only if

\[(R1 \cap R2) \rightarrow\rightarrow (R2 - R1)).\]

Algorithm 11.5: Relational decomposition into 4NF relations with non-additive join property

Input: A universal relation R and a set of functional and multi-valued dependencies F.

1. Set D := \{ R \};

2. While there is a relation schema Q in D that is not in 4NF do { choose a relation schema Q in D that is not in 4NF;
find a nontrivial MVD X $\rightarrow\rightarrow$ Y in Q that violates 4NF;

replace Q in D by two relation schemas (Q - Y) and (X $\cup$ Y);

};

**Join Dependencies and Fifth Normal Form**

Definition:

A join dependency (JD), denoted by JD(R1, R2, ..., Rn), specified on relation schema R, specifies a constraint on the states r of R.

The constraint states that every legal state r of R should have a non-additive join decomposition into R1, R2, ..., Rn; that is, for every such r we have $\pi_{R1}(r), \pi_{R2}(r), ..., \pi_{Rn}(r) = r$

Note: an MVD is a special case of a JD where n = 2.

A join dependency JD(R1, R2, ..., Rn), specified on relation schema R, is a trivial JD if one of the relation schemas Ri in JD(R1, R2, ..., Rn) is equal to R.

Definition:

A relation schema R is in fifth normal form (5NF) (or Project-Join Normal Form (PJNF)) with respect to a set F of functional, multivalued, and join dependencies if, for every nontrivial join dependency JD(R1, R2, ..., Rn) in F+ (that is, implied by F), every Ri is a superkey of R.
Relation SUPPLY with Join Dependency and conversion to Fifth Normal Form

Inclusion Dependencies

Definition:

An inclusion dependency R.X < S.Y between two sets of attributes—X of relation schema R, and Y of relation schema S—specifies the constraint that, at any specific time when r is a relation state of R and s a relation state of S, we must have \( \pi_X(r(R)) \supseteq \pi_Y(s(S)) \)

Note:

The \( \subseteq \) (subset) relationship does not necessarily have to be a proper subset.

The sets of attributes on which the inclusion dependency is specified—X of R and Y of S—must have the same number of attributes.

In addition, the domains for each pair of corresponding attributes should be compatible.

Objective of Inclusion Dependencies:
To formalize two types of interrelational constraints which cannot be expressed using F.D.s or MVDs:

Referential integrity constraints

Class/subclass relationships

**Inclusion dependency inference rules**

IDIR1 (reflexivity): \( R.X < R.X \).

IDIR2 (attribute correspondence): If \( R.X < S.Y \)

where \( X = \{ A_1, A_2, ..., A_n \} \) and \( Y = \{ B_1, B_2, ..., B_n \} \) and \( A_i \) Corresponds-to \( B_i \), then \( R.A_i < S.B_i \) for \( 1 \leq i \leq n \).

IDIR3 (transitivity): If \( R.X < S.Y \) and \( S.Y < T.Z \), then \( R.X < T.Z \).

**Other Dependencies and Normal Forms**

**Template Dependencies:**

Template dependencies provide a technique for representing constraints in relations that typically have no easy and formal definitions.

The idea is to specify a template—or example—that defines each constraint or dependency.

**There are two types of templates:**

- tuple-generating templates

- constraint-generating templates.

A template consists of a number of hypothesis tuples that are meant to show an example of the tuples that may appear in one or more relations. The other part of the template is the template conclusion.
Domain-Key Normal Form (DKNF):

Definition:

A relation schema is said to be in DKNF if all constraints and dependencies that should hold on the valid relation states can be enforced simply by enforcing the domain constraints and key constraints on the relation.
The idea is to specify (theoretically, at least) the “ultimate normal form” that takes into account all possible types of dependencies and constraints.

For a relation in DKNF, it becomes very straightforward to enforce all database constraints by simply checking that each attribute value in a tuple is of the appropriate domain and that every key constraint is enforced.

The practical utility of DKNF is limited