Advanced Normalization

CSC8490
Database Systems and File Management
Advanced Normalization Topics
Dr. James Dullea
james.dullea@boeing.com
professor@comcast.net

Revised: December 7, 2003

Important Concepts from This Lecture

• Third Normal Form
• Boyce-Codd Normal Form
• Lossless (Non-Additive) Join Property
• Attribute Preservation
• Dependency Preservation
More on Boyce-Codd Normal Form

Decomposing a relation to meet the Third Normal Form standard always preserves all the functional dependencies. Hence the set for FDs defined by the requirements of the unnormalized relation is always equivalent to set of FDs that result from the decomposition in 3NF.

• This statement cannot be said about higher normal forms. Decomposition into BCNF can lose FDs
• Also, decomposition is not straightforward and an incorrect decomposition can yield joins that lose or add tuples.

Decision Time

If we are faced with the loss of a functional dependency, what should we do?

If the constraint (FD) is important, then decompose no further than 3NF

If removing redundancy is more important than retaining the constraint, then decompose to BCNF.
More on Boyce-Codd Normal Form

\begin{align*}
\text{FD1} & : \{ \text{STUDENT, COURSE} \} \rightarrow \{ \text{INSTRUCTOR} \} \\
\text{FD2} & : \{ \text{INSTRUCTOR} \} \rightarrow \{ \text{COURSE} \}
\end{align*}
Student_Course_Instructor Table

```sql
DROP TABLE sci;
CREATE TABLE sci (  
    student       VARCHAR2(10),
    course        VARCHAR2(10),
    instructor    VARCHAR2(10),
    CONSTRAINT sci_pk PRIMARY KEY (student, course));

INSERT INTO sci VALUES ('Lixi','C8490','Dullea');
INSERT INTO sci VALUES ('Ioanna','C8490','Dullea');
INSERT INTO sci VALUES ('Runjing','C8490','Dullea');
INSERT INTO sci VALUES ('Kannis','C8490','Dullea');
INSERT INTO sci VALUES ('Lixi','C8350','Goelman');
INSERT INTO sci VALUES ('John','C8350','Goelman');
INSERT INTO sci VALUES ('Georgia','C8350','Goelman');
INSERT INTO sci VALUES ('Georgia','C8490','Lamprou');
INSERT INTO sci VALUES ('Runjing','C1050','Chen');
```

```sql
SELECT 'SCI', student, course, instructor
FROM sci;
```

<table>
<thead>
<tr>
<th>SCI</th>
<th>STUDENT</th>
<th>COURSE</th>
<th>INSTRUCTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lixi</td>
<td>C8490</td>
<td></td>
<td>Dullea</td>
</tr>
<tr>
<td>Ioanna</td>
<td>C8490</td>
<td></td>
<td>Dullea</td>
</tr>
<tr>
<td>Runjing</td>
<td>C8490</td>
<td></td>
<td>Dullea</td>
</tr>
<tr>
<td>Kannis</td>
<td>C8490</td>
<td></td>
<td>Dullea</td>
</tr>
<tr>
<td>Lixi</td>
<td>C8350</td>
<td></td>
<td>Goelman</td>
</tr>
<tr>
<td>John</td>
<td>C8350</td>
<td></td>
<td>Goelman</td>
</tr>
<tr>
<td>Georgia</td>
<td>C8350</td>
<td></td>
<td>Goelman</td>
</tr>
<tr>
<td>Georgia</td>
<td>C8490</td>
<td></td>
<td>Lamprou</td>
</tr>
<tr>
<td>Runjing</td>
<td>C1050</td>
<td></td>
<td>Chen</td>
</tr>
</tbody>
</table>

9 rows selected.
More on Boyce-Codd Normal Form

Three possible decompositions

{STUDENT, COURSE} and {INSTRUCTOR, COURSE}

{STUDENT, INSTRUCTOR} and {STUDENT, COURSE}

{STUDENT, INSTRUCTOR} and {INSTRUCTOR, COURSE}

Tables

DROP TABLE si;
CREATE TABLE si
    (student VARCHAR2(10),
     instructor VARCHAR2(10),
     CONSTRAINT si_pk PRIMARY KEY (student, instructor));

DROP TABLE sc;
CREATE TABLE sc
    (student VARCHAR2(10),
     course VARCHAR2(10),
     CONSTRAINT sc_pk PRIMARY KEY (student, course));

DROP TABLE ci;
CREATE TABLE ci
    (course VARCHAR2(10),
     instructor VARCHAR2(10),
     CONSTRAINT ci_pk PRIMARY KEY (instructor));
Student_Course_Instructor Table

- INSERT INTO sc VALUES ('Lixi', 'C8490');
- INSERT INTO sc VALUES ('Ioanna', 'C8490');
- INSERT INTO sc VALUES ('Runjing', 'C8490');
- INSERT INTO sc VALUES ('Kannis', 'C8490');
- INSERT INTO sc VALUES ('Lixi', 'C8350');
- INSERT INTO sc VALUES ('John', 'C8350');
- INSERT INTO sc VALUES ('Georgia', 'C8350');
- INSERT INTO sc VALUES ('Georgia', 'C8490');
- INSERT INTO sc VALUES ('Runjing', 'C1050');

- INSERT INTO ci VALUES ('C8490', 'Dullea');
- INSERT INTO ci VALUES ('C8350', 'Goelman');
- INSERT INTO ci VALUES ('C8490', 'Lamprou');
- INSERT INTO ci VALUES ('C1050', 'Chen');

DATA

- INSERT INTO si VALUES ('Lixi', 'Dullea');
- INSERT INTO si VALUES ('Ioanna', 'Dullea');
- INSERT INTO si VALUES ('Runjing', 'Dullea');
- INSERT INTO si VALUES ('Kannis', 'Dullea');
- INSERT INTO si VALUES ('Lixi', 'Goelman');
- INSERT INTO si VALUES ('John', 'Goelman');
- INSERT INTO si VALUES ('Georgia', 'Goelman');
- INSERT INTO si VALUES ('Georgia', 'Lamprou');
- INSERT INTO si VALUES ('Runjing', 'Chen');
**SELECT ` ' "SC/IC", sc.student, sc.course, ci.instructor**
FROM sc, ci
WHERE sc.course = ci.course;

<table>
<thead>
<tr>
<th>SC/IC STUDENT</th>
<th>COURSE</th>
<th>INSTRUCTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runjing</td>
<td>C1050</td>
<td>Chen</td>
</tr>
<tr>
<td>Lixi</td>
<td>C8350</td>
<td>Goelman</td>
</tr>
<tr>
<td>John</td>
<td>C8350</td>
<td>Goelman</td>
</tr>
<tr>
<td>Georgia</td>
<td>C8350</td>
<td>Goelman</td>
</tr>
<tr>
<td>Lixi</td>
<td>C8490</td>
<td>Dullea</td>
</tr>
<tr>
<td>Georgia</td>
<td>C8490</td>
<td>Dullea</td>
</tr>
<tr>
<td>Ioanna</td>
<td>C8490</td>
<td>Dullea</td>
</tr>
<tr>
<td>Runjing</td>
<td>C8490</td>
<td>Dullea</td>
</tr>
<tr>
<td>Kannis</td>
<td>C8490</td>
<td>Dullea</td>
</tr>
<tr>
<td>Lixi</td>
<td>C8490</td>
<td>Lamprou</td>
</tr>
<tr>
<td>Georgia</td>
<td>C8490</td>
<td>Lamprou</td>
</tr>
<tr>
<td>Ioanna</td>
<td>C8490</td>
<td>Lamprou</td>
</tr>
<tr>
<td>Runjing</td>
<td>C8490</td>
<td>Lamprou</td>
</tr>
<tr>
<td>Kannis</td>
<td>C8490</td>
<td>Lamprou</td>
</tr>
</tbody>
</table>

14 rows selected.

**SELECT ` ' "SC/SI", sc.student, sc.course, si.instructor**
FROM sc, si
WHERE sc.student = si.student;

<table>
<thead>
<tr>
<th>SC/SI STUDENT</th>
<th>COURSE</th>
<th>INSTRUCTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lixi</td>
<td>C8350</td>
<td>Dullea</td>
</tr>
<tr>
<td>Lixi</td>
<td>C8490</td>
<td>Dullea</td>
</tr>
<tr>
<td>Ioanna</td>
<td>C8490</td>
<td>Dullea</td>
</tr>
<tr>
<td>Runjing</td>
<td>C1050</td>
<td>Dullea</td>
</tr>
<tr>
<td>Runjing</td>
<td>C8490</td>
<td>Dullea</td>
</tr>
<tr>
<td>Kannis</td>
<td>C8490</td>
<td>Dullea</td>
</tr>
<tr>
<td>Lixi</td>
<td>C8350</td>
<td>Goelman</td>
</tr>
<tr>
<td>Lixi</td>
<td>C8490</td>
<td>Goelman</td>
</tr>
<tr>
<td>John</td>
<td>C8350</td>
<td>Goelman</td>
</tr>
<tr>
<td>Georgia</td>
<td>C8350</td>
<td>Goelman</td>
</tr>
<tr>
<td>Georgia</td>
<td>C8490</td>
<td>Goelman</td>
</tr>
<tr>
<td>Georgia</td>
<td>C8490</td>
<td>Goelman</td>
</tr>
<tr>
<td>Runjing</td>
<td>C1050</td>
<td>Chen</td>
</tr>
<tr>
<td>Runjing</td>
<td>C8490</td>
<td>Chen</td>
</tr>
</tbody>
</table>

15 rows selected.
SELECT 'SI/IC', si.student, ci.course, ci.instructor
FROM si, ci
WHERE si.instructor = ci.instructor;

<table>
<thead>
<tr>
<th>SI/IC</th>
<th>STUDENT</th>
<th>COURSE</th>
<th>INSTRUCTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----</td>
<td>-------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>Lixi</td>
<td>C8490</td>
<td>Dullea</td>
<td></td>
</tr>
<tr>
<td>Ioanna</td>
<td>C8490</td>
<td>Dullea</td>
<td></td>
</tr>
<tr>
<td>Runjing</td>
<td>C8490</td>
<td>Dullea</td>
<td></td>
</tr>
<tr>
<td>Kannis</td>
<td>C8490</td>
<td>Dullea</td>
<td></td>
</tr>
<tr>
<td>Lixi</td>
<td>C8350</td>
<td>Goelman</td>
<td></td>
</tr>
<tr>
<td>John</td>
<td>C8350</td>
<td>Goelman</td>
<td></td>
</tr>
<tr>
<td>Georgia</td>
<td>C8350</td>
<td>Goelman</td>
<td></td>
</tr>
<tr>
<td>Georgia</td>
<td>C8490</td>
<td>Lamprou</td>
<td></td>
</tr>
<tr>
<td>Runjing</td>
<td>C1050</td>
<td>Chen</td>
<td></td>
</tr>
</tbody>
</table>

9 rows selected.

How do we know which one to use?

FD1

FD2

Three possible decompositions

{ STUDENT, COURSE } and { INSTRUCTOR, COURSE }

{ STUDENT, INSTRUCTOR } and { STUDENT, COURSE }

{ STUDENT, INSTRUCTOR } and { INSTRUCTOR, COURSE }

NOTE: FD1 is lost in each of the decompositions
Design Quality

Besides Normal Forms we need additional properties to insure a quality design. Normal Form alone does not guarantee a quality design.

• Attribute Preservation
• Dependency Preservation
• Lossless (Non-additive) Joins

Attribute Preservation

Attribute preservation insures that all the attributes in the original relations appears at least once in the decomposed set of relations.

If relation $R$ is decomposed in $R_k$ relations where $k = 1, \ldots, m$ and the union of the attributes of each $R_k$ for $k=1,m$ is equal to all the attributes of $R$ then the attribute preservation condition holds.
Dependency Preservation

It is possible during decomposition to lose one of more functional dependencies from the original relation.

Recall the meaning of equivalence between sets of FDs.

The dependency preservation holds if the union of the functional dependencies of the decomposed relations is equivalent to the FDs of the original relation.

Not all decompositions preserve dependencies.

Dependency Preservation

Example

Recall in the STUDENT, COURSE, INSTRUCTOR problem when we decomposed to BCNF, we lost FD1.

Original Functional Dependencies

\{ \text{STUDENT, COURSE} \} \rightarrow \{ \text{INSTRUCTOR} \}
\{ \text{INSTRUCTOR} \} \rightarrow \{ \text{COURSE} \}
Dependency Preservation

Example

Decomposed Relations

\{ \text{STUDENT, INSTRUCTOR} \} \\
and \{ \text{INSTRUCTOR, COURSE} \}

Functional Dependencies

\{ \text{STUDENT, INSTRUCTOR} \} \rightarrow \{ \text{STUDENT, INSTRUCTOR} \}

\{ \text{INSTRUCTOR} \} \rightarrow \{ \text{COURSE} \}

Some Derived Functional Dependencies

\{ \text{INSTRUCTOR} \} \rightarrow \{ \text{COURSE} \}

\{ \text{STUDENT, INSTRUCTOR} \} \rightarrow \{ \text{STUDENT, COURSE} \}

\{ \text{STUDENT, INSTRUCTOR} \} \rightarrow \{ \text{STUDENT} \}

\{ \text{STUDENT, INSTRUCTOR} \} \rightarrow \{ \text{COURSE} \}

\{ \text{STUDENT, INSTRUCTOR} \} \rightarrow \{ \text{INSTRUCTOR} \}

Note: \text{COURSE} cannot appear on the LHS of a derived FD without appearing on the RHS.

Therefore, we cannot infer the original FD

\{ \text{STUDENT, COURSE} \} \rightarrow \{ \text{INSTRUCTOR} \}

and this FD is not preserved and cannot be enforced.
### Lossless (Non-additive) Joins

For the decomposition of $R$ into $R_k \ k=1,n$ the lossless (non-additive) join property holds if the natural join between all $R_k \ k=1,n$ returns the original relation $R$.

### Testing for Lossless Join

$R_1, R_2$ is a lossless join decomposition of $R$ with respect to $F$ iff at least one of the following dependencies is in $F^+$

$$( R_1 \cap R_2 ) \rightarrow R_1 - R_2$$

or

$$( R_1 \cap R_2 ) \rightarrow R_2 - R_1$$
\[
\begin{align*}
\text{STUDENT} & \rightarrow \text{COURSE} (\text{NOT IN F}) \\
\text{STUDENT} & \rightarrow \text{INSTRUCTOR} (\text{NOT IN F})
\end{align*}
\]
3NF versus BCNF

- There is always a lossless join, dependency preserving decomposition into 3NF.
- There is always a lossless join decomposition into BCNF; however, there may not be one that preserves dependencies.
- BCNF is a stronger form than 3NF: Every BCNF schema is also in 3NF.
- The BCNF algorithms are nondeterministic, so there is not a unique decomposition for a given schema $R$. In BCNF the order in which the dependencies is considered determines the design.
Multivalued Dependencies

- Multivalued Dependencies are a consequence of 1NF which disallows an attribute to have a set of values or repeating groups.
- Two or more multivalued independent attributes in a relation presents redundancy, storage, and updating issues.
- Notation: \( X \rightarrow Y \)

Multivalued Dependencies
Formal Definition

A multivalued dependency 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 following properties, where \( Z \) denotes \((R - (X \cup Y))\):

- \( t_3[X] = t_4[X] = t_1[X] = t_2[X] \).
- \( t_3[Y] = t_1[Y] \) and \( t_4[Y] = t_2[Y] \).
- \( t_3[Z] = t_2[Z] \) and \( t_4[Z] = t_1[Z] \).
MVD Example

\[ R(\text{LNAME, DEGREE, SKILL}) \]

\[ \text{MVD1: } \{ \text{LNAME} \} \rightarrow \{ \text{DEGREE, SKILL} \} \]
Fourth Normal Form

• A relation $R$ is in fourth normal form with respect to a set of FDs and MVDs if for every nontrivial multivalued dependency

$$X \rightarrow Y \text{ in } F^+,$$

$X$ is a superkey of $R$

Note: FDs are special cases of MVDs, so the above definition requires $R$ to be in BCNF.

In the relation $R$ (LNAME, DEGREE, SKILL)

MVD1: \{LNAME\} $\rightarrow$ \{DEGREE, SKILL\}

LNAME is not a superkey and therefore NOT in FOURTH NORMAL FORM. It should be decomposed as follows:

R1 (LNAME, DEGREE) and R2 (LNAME, SKILL)
MVD Example

<table>
<thead>
<tr>
<th>LNAME</th>
<th>DEGREE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DULLEA</td>
<td>BS</td>
</tr>
<tr>
<td>DULLEA</td>
<td>MS</td>
</tr>
<tr>
<td>DULLEA</td>
<td>PhD</td>
</tr>
<tr>
<td>MILLER</td>
<td>BS</td>
</tr>
<tr>
<td>MILLER</td>
<td>MBA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LNAME</th>
<th>SKILL</th>
</tr>
</thead>
<tbody>
<tr>
<td>DULLEA</td>
<td>INFORMATION TECHNOLOGIST</td>
</tr>
<tr>
<td>DULLEA</td>
<td>TEACHER</td>
</tr>
<tr>
<td>DULLEA</td>
<td>RESEARCHER</td>
</tr>
<tr>
<td>MILLER</td>
<td>PROGRAMMER</td>
</tr>
<tr>
<td>MILLER</td>
<td>ACCOUNTANT</td>
</tr>
</tbody>
</table>

Join Dependencies

- The join dependency constraint states that every decomposed relation $R_k$ of $R$ should be lossless.
- The projection of natural joins of all $R_k$ (independent of the state of $r$) must equal $R$. 
Fifth Normal Form

- A relation $R$ is in fifth normal form (5NF) with respect to a set of FDs, MVDs, and JDs if, for every nontrivial join dependency $JD(R_1, R_2, \ldots, R_k)$ in $FD^+$, every $R_k$ is a superkey of $R$.
  - Not very practical, hardly ever used in practice
  - Used in proofs pertaining to relations containing ternary and n-ary relationships

Design Methodologies

Bottom-Up Design
- Purist approach that views RDMS design strictly in terms of functional and other dependencies specified on attributes. Relations are developed from normalization algorithms

Top-Down Design
- Creating a conceptual schema into a set of relations using mapping procedures.

Combination of Both Methods