15CSE302 Database Management Systems

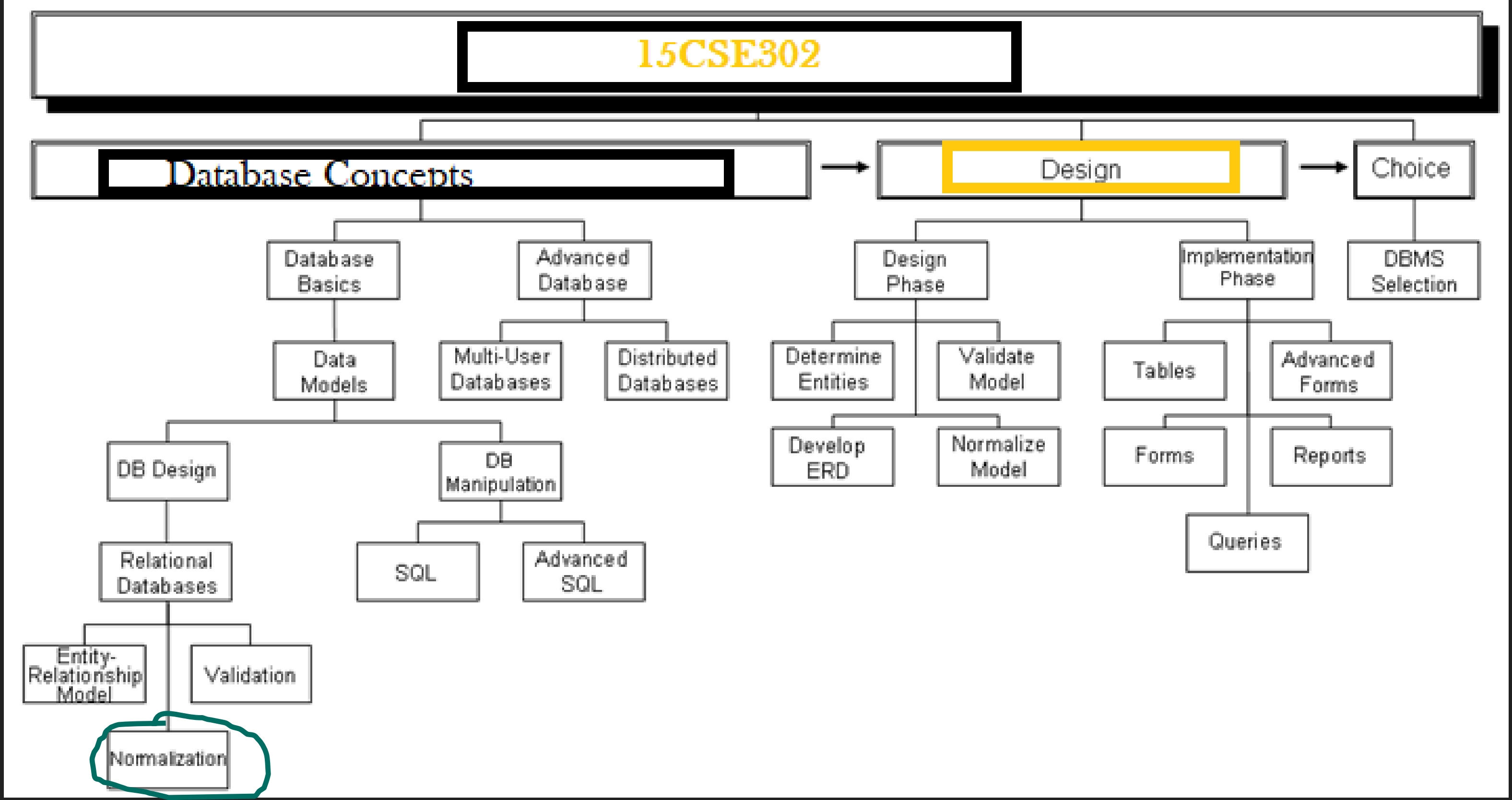
Lecture 23 Lossless Decomposition

B.Tech /III Year CSE/V Semester **L T P C 2 0 2 3**

DBMS Team
Dr G Jeyakumar
Bindu K R
Dr Priyanka Kumar
R. Manjusha
Department of CSE
Amrita School of Engineering

Slides Courtesy : Carlos Alvarado,
San Jose State University

Syllabus


15CSE302
Database Concepts
Design
Choice
Database Basics
Advanced Database
Design Phase
Implementation Phase
DBMS Selection
Data Models
Multi-User Databases
Distributed Databases
Determine Entities
Validate Model
Tables
Advanced Forms
DB Design
DB Manipulation
Develop ERD
Normalize Model
Forms
Reports
Queries
Relational Databases
SQL
Advanced SQL
Entity-Relationship Model
Validation
Normalization

Brief Recap of Previous Lecture

- Canonical Cover
- BCNF

Today we'll discuss

Lossless Decomposition
Dependency Preserving

Desirable properties of Decomposition

When we decompose a relation schema R with a set of functional dependencies F into R_1, R_2, ..., R_n ,these properties should be satisfied

- **Non-additive (Lossless) join decomposition:**
    - Otherwise decomposition would result in information loss.
- **Dependency preservation:**
    - Otherwise, checking updates for violation of functional dependencies may require computing joins, which is expensive.
- **No redundancy:**

  The relations R_i preferably should be in either Boyce-Codd Normal Form or 3NF.

Testing for Lossless Join Property

Lossless-join Decomposition

- Let F be a set of **functional dependencies** on R.
- Let R1 and R2 form a decomposition of R.
- The decomposition is a lossless-join decomposition of R if at least one of the following functional dependencies are in F^+ (where F^+ stands for the closure for every attribute or attribute sets in F
- $R1 \cap R2 \rightarrow R1$
- $R1 \cap R2 \rightarrow R2$

Lossless-join Decomposition

- Let $R = (A, B, C, D)$ with A, B , C and D attributes.
- Let $F = \{ A \rightarrow B\ C \}$ be the set of functional dependencies.
- Decomposition into

$R1 = (A, B, C)$ and $R2 = (A, D)$ is lossless under F

because $R1 \cap R2 = (A)$

A is a superkey in R1 $(A \rightarrow B\ C)$

$R1 \cap R2 \rightarrow R1$

Lossless-join Decomposition

- $R = (A, B, C)$
  $F = \{A \rightarrow B, B \rightarrow C)$
  - Can be decomposed in two different ways
- $R_1 = (A, B), \quad R_2 = (B, C)$
  - Lossless-join decomposition:
    $$R_1 \cap R_2 = \{B\} \text{ and } B \rightarrow BC$$
  - Dependency preserving
- $R_1 = (A, B), \quad R_2 = (A, C)$
  - Lossless-join decomposition:
    $$R_1 \cap R_2 = \{A\} \text{ and } A \rightarrow AB$$
  - Not dependency preserving
    (cannot check $B \rightarrow C$ without computing $R_1 \bowtie R_2$)

Lossless-join Decomposition

- Example of **spurious** tuples.
- Decomposition of $R = (A, B)$
  - $R_1 = (A)$ $R_2 = (B)$

A	B
α	1
α	2
β	1

r

A
α
β

$\Pi_A(r)$

B
1
2

$\Pi_B(r)$

$\Pi_A(r) \bowtie \Pi_B(r)$

A	B
α	1
α	2
β	1
β	2

Lossless-join Decomposition

- **Lossless (Non-additive) Join Property of a Decomposition**:
- This property ensures that no **spurious** tuples are generated when a NATURAL JOIN operation is applied to the relations in the decomposition.
- **Lossless join property**: a decomposition $D = \{R_1, R_2, \ldots, R_m\}$ 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:
  - $* (\pi_{R1}(r), \ldots, \pi_{Rm}(r)) = r$
  - $r \subseteq r_1 * r_2 * \ldots * r_m$ and
  - $r_1 * r_2 * \ldots * r_m \subseteq r$

Lossless-join Decomposition

Consider

Id#	Name	Address	C#	Description	Grade
124	Jones	Phila	Phil7	Plato	A
789	Brown	Boston	Math8	Topology	C

What happens if we decompose on
R1(Id#, Name, Address) and
R2(C#, Description, Grade)?
When we join the relations R1 and R2
Spurious tuples will be generated

Chase Method: Testing for Lossless Join

Consider

Id#	Name	Address	C#	Description	Grade
124	Jones	Phila	Phil7	Plato	A
789	Brown	Boston	Math8	Topology	C

SSN	Name	Address
1111	Joe	1 Pine
2222	Alice	2 Oak
3333	Alice	3 Pine

SSN	Name
1111	Joe
2222	Alice
3333	Alice

Name	Address
Joe	1 Pine
Alice	2 Oak
Alice	3 Pine

Lossy Decomposition: R NOT $\supseteq$ R1 $\bowtie$ R2

Problem: Name is not a key

Chase Method: Testing for Lossless Join Property

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

1. Create an initial matrix S with one row i for each relation R_i in D, and one column j for each attribute A_j in R.

2. Set $S(i, j) = b_{ij}$ for all matrix entries.
   // each b_{ij} is a symbol associated with indices (i, j)

3. For each row i representing relation schema R_i
   For each column j representing attribute A_j
   if (relation R_i includes attribute A_j) then $S(i, j) = a_j$
   // each a_j is a symbol associated with index j

Chase Method: Testing for Lossless Join Property

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

4. Repeat until a *complete loop execution* results in no changes to *S*
for each functional dependency $X \to Y$ 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.

Testing for Lossless Join Property

Consider the schema R=AB CD, subjected to FDs F= { A → B, B → C }, and the Non-binary partition D_1 = {ACD, AB, BC}.

Question Is D_1 a Lossless decomposition?

	A	B	C	D
ACD	b_{11}	b_{12}	b_{13}	b_{14}
AB	b_{21}	b_{22}	b_{23}	b_{24}
BC	b_{31}	b_{32}	b_{33}	b_{34}

Step 1. Create initial table, entering b_{ij} values in each cell

Testing for Lossless Join Property

Consider the schema R=ABCD, subjected to FDs F={ A → B, B → C }, and t
Non-binary partition D_1 = {ACD, AB, BC}.

	A	B	C	D
ACD	a_1	b_{12}	a_3	a_4
AB	a_1	a_2	b_{23}	b_{24}
BC	b_{31}	a_2	a_3	b_{34}

Step 2. For each attribute mentioned in each partition R_i introduce a_j on column j

Testing for Lossless Join Property

Consider the schema R=ABCD, subjected to FDs F={ A → B, B → C }, and t
Non-binary partition D_1 = {ACD, AB, BC}.

	A	B	C	D
ACD	a_1	a_2	a_3	a_4
AB	a_1	a_2	a_3	b_{24}
BC	b_{31}	a_2	a_3	b_{34}

Step 3. Apply FDs to unify a_j, b_{ij} symbols.
Using B→C we discover a_2→{a_3, b_{23}}
therefore b_{23} becomes a_3.
Using A→B we discover a_1→{a_2, b_{12}}
hence b_{12} can be replaced by a_2.
First row now has a_j values in each cell. Stop!

Conclusion: Decomposition D_1= {ACD, AB, BC} on <R, F> has a lossless-join

Lossless (**non-additive**) join test for n-ary decompositions

R={SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS} $D=\{R_1, R_2\}$

R_1=EMP_LOCS={ENAME, PLOCATION}

R_2=EMP_PROJ1={SSN, PNUMBER, HOURS, PNAME, PLOCATION}

F={SSN→ENAME;PNUMBER→{PNAME, PLOCATION} ;{SSN,PNUMBER}→HOURS}

Lossless (**non-additive**) join test for n-ary decompositions

R={SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS} $D=\{R_1, R_2\}$

R_1=EMP_LOCS={ENAME, PLOCATION}

R_2=EMP_PROJ1={SSN, PNUMBER, HOURS, PNAME, PLOCATION}

F={SSN→ENAME;PNUMBER→{PNAME, PLOCATION} ;{SSN,PNUMBER}→HOURS}

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	b_{11}	a_2	b_{13}	b_{14}	a_5	b_{16}
R_2	a_1	b_{22}	a_3	a_4	a_5	a_6

Lossless (**non-additive**) join test for n-ary decompositions

R={SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS} $D=\{R_1, R_2\}$

R_1=EMP_LOCS={ENAME, PLOCATION}

R_2=EMP_PROJ1={SSN, PNUMBER, HOURS, PNAME, PLOCATION}

F={SSN→ENAME;PNUMBER→{PNAME, PLOCATION} ;{SSN,PNUMBER}→HOURS}

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	b_{11}	a_2	b_{13}	b_{14}	a_5	b_{16}
R_2	a_1	b_{22}	a_3	a_4	a_5	a_6

(no changes to matrix after applying functional dependencies)

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

Case 2: Decomposition of EMP_PROJ into EMP, PROJECT, and WORKS_ON satisfies test

R={SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS} $D=\{R_1, R_2, R_3\}$
R_1=EMP={SSN, ENAME}
R_2=PROJ={PNUMBER, PNAME, PLOCATION}
R_3=WORKS_ON={SSN, PNUMBER, HOURS}

F={SSN→{ENAME;PNUMBER→{PNAME, PLOCATION} ;{SSN,PNUMBER}→HOURS}

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

Case 2: Decomposition of EMP_PROJ into EMP, PROJECT, and WORKS_ON satisfies test

R={SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS} D={R_1, R_2, R_3}
R_1=EMP={SSN, ENAME}
R_2=PROJ={PNUMBER, PNAME, PLOCATION}
R_3=WORKS_ON={SSN, PNUMBER, HOURS}

F={SSN→{ENAME;PNUMBER→{PNAME, PLOCATION} ;{SSN,PNUMBER}→HOURS}

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

(original matrix S at start of algorithm)

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

Case 2: Decomposition of EMP_PROJ into EMP, PROJECT, and WORKS_ON satisfies test

R={SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS} $D=\{R_1, R_2, R_3\}$

R_1=EMP={SSN, ENAME}

R_2=PROJ={PNUMBER, PNAME, PLOCATION}

R_3=WORKS_ON={SSN, PNUMBER, HOURS}

F={SSN→{ENAME;PNUMBER→{PNAME, PLOCATION} ;{SSN,PNUMBER}→HOURS}

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

(original matrix S at start of algorithm)

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	$\cancel{b_{32}}$ a_2	a_3	$\cancel{b_{34}}$ a_4	$\cancel{b_{35}}$ a_5	a_6

(matrix S after applying the first two functional dependencies - last row is all "a" symbols, so we stop)

References

- Hillyer Mike, MySQL AB. An Introduction to Database Normalization, http://dev.mysql.com/tech-resources/articles/intro-to-normalization.html, accessed October 17, 2006.

- Microsoft. Description of the database normalization basics, http://support.microsoft.com/kb/283878 , accessed October 17, 2006.

- Wikipedia. Database Normalization. http://en.wikipedia.org/wiki/Database_normalization.html , accessed October 17, 2006. https://www.db-book.com/db6/index.html
- https://www.youtube.com/watch?v=mfVCesoMaGA&list=PLroEs25KGvwzmvIxYHRhoGTz9w8LeXek0&index=22

Summary

- Lossless join

Next Lecture

Dependency Preservation

Thank You

Happy to answer any questions ! ! !