Relational Database Decomposition Criteria

Robert Keller

11 February 2004

Type of Decomposition

- The type of decomposition considered here consists of **projecting** relations onto subsets of their attributes.

- It is desired to be able to reconstruct the original relation from the components of the decomposition using the **natural join** operator, described elsewhere.

Projection

- To project a relation on a subset of its attributes, simply discard columns that are not in the set, then eliminate duplicates from the result.

- If S is the subset, the projection is denoted $\pi_{\mathbf{S}}(\mathbf{R})$.

Projection Example

R

S **student**	**D** **dept.**	**C** **course**	**X** **section**	**Y** **year**	**M** **semester**	**I** **instructor**
John	CS	60	1	2003	fall	Dodds
Susan	CS	60	1	2003	fall	Dodds
Fred	CS	60	1	2004	spring	Stone
John	CS	60	2	2004	spring	Hadas
Susan	CS	70	1	2004	spring	O'Neill
Susan	Math	55	1	2004	spring	Yong

$\pi_{\{D,C,Y\}}(R)$

D **dept.**	**C** **course**	**Y** **year**	
CS	60	2003	
~~CS~~	~~60~~	~~2003~~	dup
CS	60	2004	
~~CS~~	~~60~~	~~2004~~	dup
CS	70	2004	
Math	55	2004	

D **dept.**	**C** **course**	**Y** **year**
CS	60	2003
CS	60	2004
CS	70	2004
Math	55	2004

Join

- The join of two relations has as its set of **attributes** the **union** of the sets of attributes of the original relations.

- The set of tuples in the join are all those tuples formed from pairs of tuples, one from each of the original relations, where the values of the attributes common to both tuples are the same and the values of all attributes are the same as in the two tuples.

Join Example

D C Y
(CS, 60, 2003)

(John, CS, 60)
S D C

(John, CS, 60, 2003)

{D, C} are common

Join Example

S

D **dept.**	**C** **course**	**Y** **year**
CS	60	2003
CS	60	2004
CS	70	2004
Math	55	2004

T

S **student**	**D** **dept.**	**C** **course**
John	CS	60
Susan	CS	60
Fred	CS	60
Susan	CS	70
Susan	Math	55

{D, C} in common

join(S, T)

S **student**	**D** **dept.**	**C** **course**	**Y** **year**
John	CS	60	2003
John	CS	60	2004
Susan	CS	60	2003
Susan	CS	60	2004
Fred	CS	60	2003
Fred	CS	60	2004
Susan	CS	70	2004
Susan	Math	55	2004

Degenerate Joins

- Normally, those sets have at least one attribute in comment. If not, we have a degenerate join, a **Cartesian product**.

Two Desirable Properties when Decomposing

- Losslessness (what UW calls “recovery”)
- Dependency-preserving

Lossless Decomposition

- This means that when the projections are joined, we get exactly the original relation and **no more**.

- It is not difficult to see for oneself that we can't get *less*.

A Relation that Projects to the Previous Example

S **student**	D **dept.**	C **course**	Y **year**
John	CS	60	2004
Susan	CS	60	2003
Fred	CS	60	2003
Susan	CS	70	2004
Susan	Math	55	2004

S

D **dept.**	C **course**	Y **year**
CS	60	2003
CS	60	2004
CS	70	2004
Math	55	2004

T

S **student**	D **dept.**	C **course**
John	CS	60
Susan	CS	60
Fred	CS	60
Susan	CS	70
Susan	Math	55

The Previous Example: Not Lossless

S

D dept.	C course	Y year
CS	60	2003
CS	60	2004
CS	70	2004
Math	55	2004

T

S student	D dept.	C course
John	CS	60
Susan	CS	60
Fred	CS	60
Susan	CS	70
Susan	Math	55

{D, C} in common

join(S, T)

S student	D dept.	C course	Y year
John	CS	60	2003
John	CS	60	2004
Susan	CS	60	2003
Susan	CS	60	2004
Fred	CS	60	2003
Fred	CS	60	2004
Susan	CS	70	2004
Susan	Math	55	2004

The Preceding Decomposition is Not Lossless

Original

S student	D dept.	C course	Y year
John	CS	60	2004
Susan	CS	60	2003
Fred	CS	60	2003
Susan	CS	70	2004
Susan	Math	55	2004

Join

S student	D dept.	C course	Y year
John	CS	60	2003
John	CS	60	2004
Susan	CS	60	2003
Susan	CS	60	2004
Fred	CS	60	2003
Fred	CS	60	2004
Susan	CS	70	2004
Susan	Math	55	2004

Dependency Preserving

- When a relation is projected, some of its FD's might not be representable in the projection, because the attributes are no longer present.

- The effective FD's in the decomposed scheme are those derived from the FD's in the projections.

- If these are the same as the originals, the decomposition is dependency preserving.

Dependency Preserving Example

- Consider relation ABCD, with FD's:
  A->B, B->C, C->D
- Decompose into two relations: ABC and CD.
- ABC supports the FD's A->B, B->C.
- CD supports the FD C->D.
- All the original dependencies are preserved.

Non-Dependency Preserving Example

- Consider relation ABCD, with FD's:
  A->B, B->C, C->D
- Decompose into two relations: ACD and BC.
- ACD supports the FD C->D and the **implied** FD A->C.
- BC supports the FD B->C.
- However, **no** relation supports A->B, so that dependency is not preserved.

Normal Form Decompositions Contrasted

- 3NF Decomposition:
  - Lossless
  - Dependency preserving
- BCNF Decomposition:
  - Lossless
  - Not necessarily dependency-preserving
  - Component relations are all BCNF, and thus 3NF
- 4NF Decomposition:
  - Lossless
  - Not necessarily dependency-preserving
  - Component relations are all 4NF, and thus BCNf and 3NF
- No decomposition is guaranteed to preserve all MVD's

Remembering Definitions

- Let FD(X, A) mean X -> A **non-trivially**!
- Let SK(X) mean X is a SuperKey.
- Let P(X) mean X is *prime* (in some key)

- BCNF: $(\forall X,F)\ FD(X, A) \rightarrow SK(X)$
- 3NF: $(\forall X,F)\ FD(X, A) \rightarrow (SK(X) \lor P(A))$

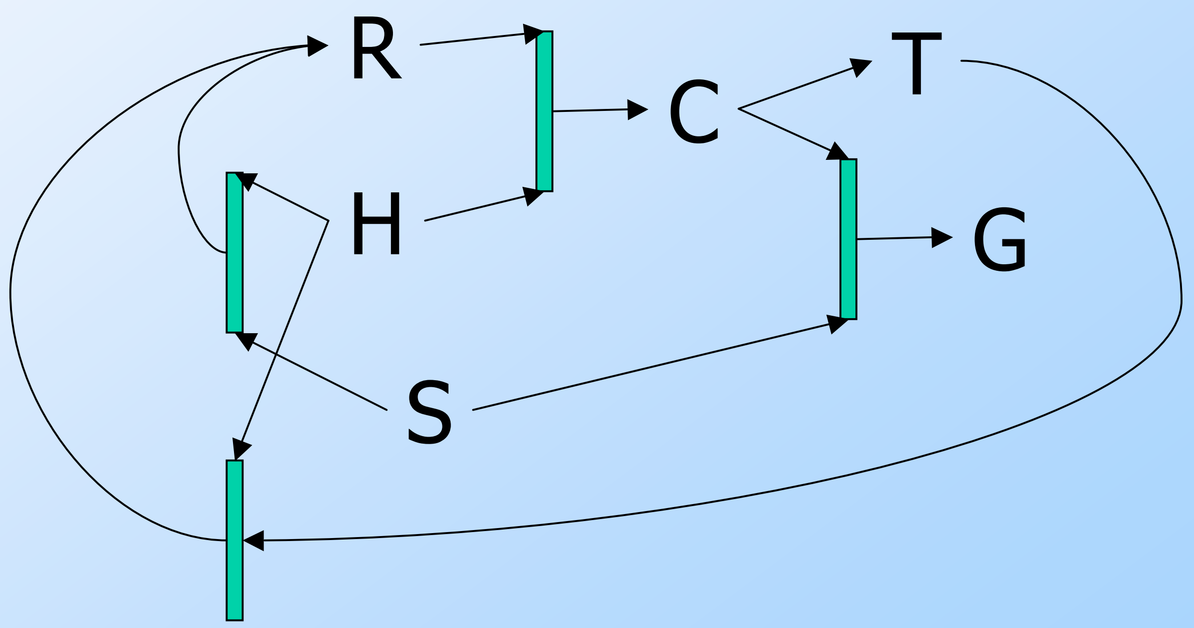
- $\neg$BCNF: $(\exists X,F)\ FD(X, A) \land \neg SK(X)$
- $\neg$3NF: $(\exists X,F)\ FD(X, A) \land (\neg SK(X) \land \neg P(A))$

3NF Decomposition

- Any set of attributes not involved in any FD becomes a relation by itself.
- Assume a ***minimal*** set of dependencies (no implied dependencies in the set).
- A relation violates 3NF when there is an FD X–>A with X not a superkey and A non-prime.
- For each such FD, create a relation consisting of only the attributes in the FD.
- Add a separate relation for the **overall key** if needed, and delete any relation subsumed.

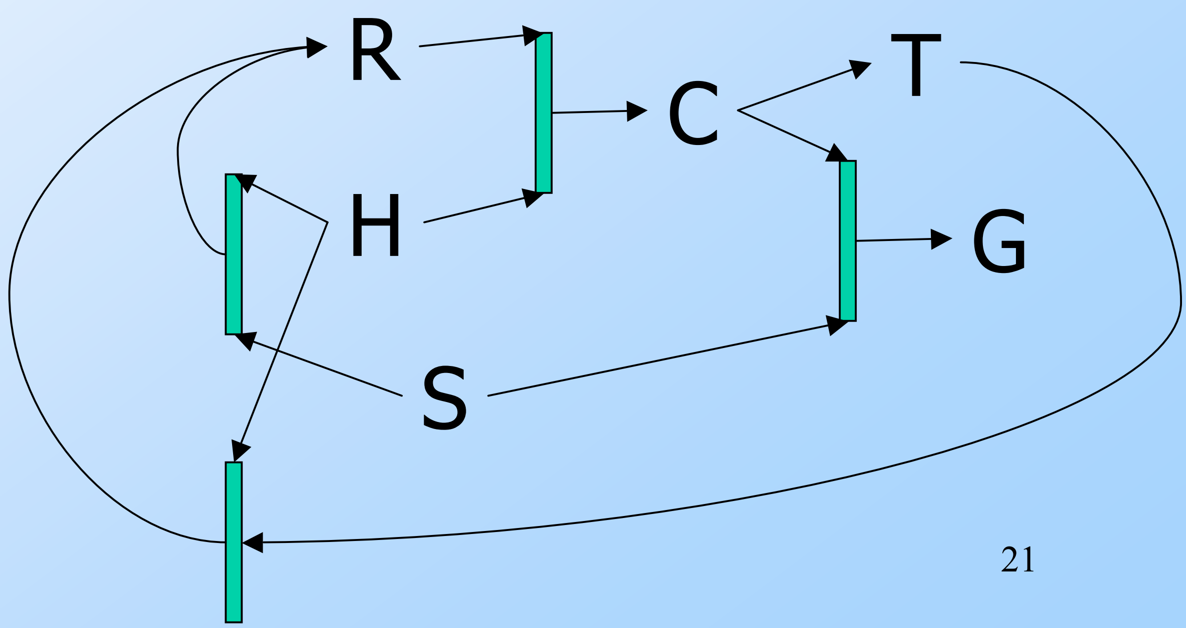
3NF Example

- CTHRSG (course-teacher-hour-room-student-grade) relation
- C -> T HR -> C HT -> R CS -> G HS -> R



3NF Example

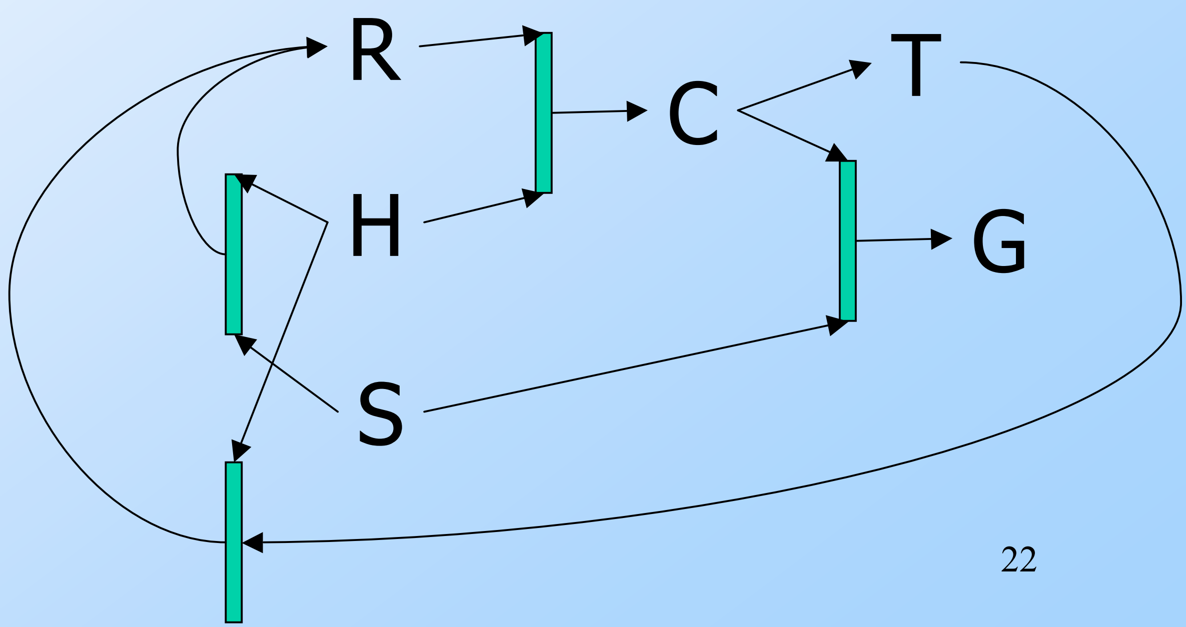
- C -> T HR -> C HT -> R CS -> G HS -> R
- Keys: HS
- Violations: all but HS -> R
- 3NF decomp: HRS, CT, HRC, HRT, CSG
- key is present in HRS



Same Example but BCNF

- C -> T HR -> C HT -> R CS -> G HS -> R
- Keys: HS
- BCNF decomp: HRS, CT, CHS, CSG
- vs. 3NF decomp: HRC, HRT, CSG

  (worth checking)



Fifth Normal Form?

- Also known as Projection-Join Normal Form (PJNF).
- 3NF, BCNF, and 4NF decompose **pairwise**.
- Accomodates relations having no pairwise lossless decomposition, but with a 3-way decomposition.

S	P	J
S1	P1	J2
S1	P2	J1
S2	P1	J1
S1	P1	J1

How to Check for Dependency Preservation

- Compute all implied FD's.
- Determine which FD's are represented by the projections.
- Compute the FD's that are implied by the ones represented by the projections.
- See if they are the same as the original implied FD's.

How to Check Losslessness

- Tableau Method
  - The collective set of attributes head the columns of a table.
  - The relation names head the rows of the table.
  - In the i^{th} row corresponding to each relation:
    - Put a_j in the j^{th} column if the j^{th} attribute is in the relation.
    - Put b_{ij} otherwise.

Lossless Check (cont'd)

- Repeatedly consider the FD's. Whenever two rows agree in their LHS columns, force the RHS columns to agree by changing one to the other.
- Continue until no more changes can be made to the table.
- If any row ends up with all a's, the decomposition is lossless.

Lossless Check Example

- Five attributes: ABCDE
- Three relations: ABC, AD, BDE
- FD’s: A -> BD, B -> E

	A	B	C	D	E
ABC	a1	a2	a3	b14	b15
AD	a1	b22	b23	a4	b25
BDE	b21	a2	b33	a4	a5

Lossless Check Example

- FD's: A -> BD, B -> E

	A	B	C	D	E
ABC	a1	a2	a3	b14	b15
AD	a1	b22	b23	a4	b25
BDE	b21	a2	b33	a4	a5

consider FD: A -> BD

	A	B	C	D	E
ABC	a1	a2	a3	a4	b15
AD	a1	a2	b23	a4	b25
BDE	b21	a2	b33	a4	a5

Lossless Check Example

- FD's: A -> BD, B -> E

	A	B	C	D	E
ABC	a1	a2	a3	a4	b15
AD	a1	a2	b23	a4	b25
BDE	b21	a2	b33	a4	a5

consider FD: B -> E

	A	B	C	D	E
ABC	a1	a2	a3	a4	a5
AD	a1	a2	b23	a4	a5
BDE	b21	a2	b33	a4	a5

lossless