CSE 412/598
DATABASE MANAGEMENT
COURSE NOTES

7. RELATIONAL DATABASE DESIGN

Department of Computer Science & Engineering
Arizona State University

RELATIONAL DATABASE DESIGN

Goal: Generate a set of relation schemes that allow us to store information without unnecessary redundancy, yet allows us to retrieve information easily.

Undesirable Properties

- Repetition of information
- Inability to represent certain information
- Loss of information

Desirable Properties

- No Repetition of Information (Normal Forms)
- Dependency Preservation
- Lossless-join Decomposition

UNDESIRABLE PROPERTIES
Repetition of information

suppliers(SNAME, SADDR, ITEM, PRICE)

The address of the supplier is repeated for each item supplied:

- ⇒ wasted space
- ⇒ every tuple for a supplier must be updated when there is an address change

UNDESIRABLE PROPERTIES
Inability to represent certain information

suppliers(SNAME, SADDR, ITEM, PRICE)

⇒ Insertion:

we can't record an address for a supplier if that supplier does not currently supply at least one item.

⇒ Deletion:

if all the items supplied by one supplier are deleted, we lose the address.

UNDESIRABLE PROPERTIES
Loss of information

addresses(NAME, CITY, ZIP) ≠
addr1(NAME, CITY) ⋈ addr2(CITY, ZIP)

The join of addr1 and addr2 is not equal to addresses

addresses ≠ **addr1** ⋈ **addr2**

N	C	Z
bill	phx	85050
jane	phx	85051

N	C
bill	phx
jane	phx

⋈

C	Z
phx	85050
phx	85051

=

N	C	Z
bill	phx	85050
bill	phx	85051
jane	phx	85050
jane	phx	85051

FUNCTIONAL DEPENDENCIES
Definition

Recall superkey defn:

Let R be a relation scheme. A subset K of R is a superkey of R if, in any legal relation r(R), for all pairs t_1 and t_2 tuples in r s.t. $t_1 \neq t_2$

$$t_1[K] \neq t_2[K]$$

"No two tuples in any legal relation r(R) may have the same value on attribute set K".

Let $V \subseteq R$ and $W \subseteq R$. The **Functional Dependency** (FD) $V \rightarrow W$ holds on R if in any legal relation r(R), for all pairs of tuples t_1 and t_2 in r s.t. $t_1[V] = t_2[V]$ it is also the case that $t_1[W] = t_2[W]$.

An FD $X \rightarrow Y$ is **trivial** if $Y \subseteq X$.

Note: if K is a superkey of R, then $K \rightarrow R$

FUNCTIONAL DEPENDENCIES
Examples

1. suppliers(SNAME, SADDR, ITEM, PRICE)

   R = {SNAME, SADDR, ITEM, PRICE}

   F = {f_1: SNAME $\rightarrow$ SADDR,

   f_2: SNAME,ITEM $\rightarrow$ PRICE}

2. addresses(NAME, CITY, ZIP)

   F = {f_1: NAME $\rightarrow$ ZIP,

   f_2: ZIP $\rightarrow$ CITY}

MORE ON FUNCTIONAL DEPENDENCIES

- A FD is a property of the meaning or semantics of the attributes in a relation schema.
- Legal relation instances r of R satisfy the functional dependency constraints.

FDs are used to

- provide a set of constraints on the attributes of a relation schema that must hold at all times
- provide additional information to be used in the design process

FUNCTIONAL DEPENDENCIES
Company Training Courses Enterprise

Most of the FDs for an enterprise are already included in the conceptual model. Using the ER diagram and the semantics of the enterprise, give the FDs for the company training courses example.

THEORY OF FUNCTIONAL DEPENDENCIES

Example: F = {X→Y, Y→Z}

By definition of FDs, X→Z is "logically implied" by F.

Let F be a set of FDs. Let F^+ denote the closure of F, which is the set of all FDs logically implied by F.

<u>Rules of Inference for FDs (FD rules)</u>

1.	Reflexivity	if $Y \subseteq X$, then X→Y
2.	Augmentation	if X→Y, then WX→WY
3.	Transitivity	if X→Y & Y→Z, then X→Z
4.	Union	if X→Y & X→Z, then X→YZ
5.	Decomposition	if X→YZ, then X→Y & X→Z
6.	Pseudotransitivity	if X→Y & WY→Z, then XW→Z

- Rules 1, 2 & 3 are Armstrong's axioms (complete)
- Rules 4, 5 & 6 follow from Armstrong's axioms

EXAMPLE PROOF OF INFERENCE

Let F = {f_1: Z→A, f_2: B→X, f_3: AX→Y}

Prove that ZB→Y is in F^+?

ZB→AB	Augmentation(f_1,B)
AB→AX	Augmentation(f_2, A)
ZB→AX	Transitivity(ZB→AB, AB→AX)
ZB→Y	Transitivity(ZB→AX, f_3)

What if we just wanted to know whether a FD is in F^+?

Computing F^+ is costly. Rather than computing F^+, we can find the set of attributes functionally determined by the given set of attributes on the left-hand side of the FD to check.

ATTRIBUTE CLOSURE

Let X be a set of attributes.

X^+ be the closure of X under F, i.e., the set of all attributes functionally determined by X under a set F of FDs.

Algorithm: compute X^+

Input: F - set of FDs

X - set of Attributes

Output: X^+ - set of attributes

Result := X; /*reflexivity */

while (changes to Result) do

for each FD $Y \rightarrow Z$ in F do

if $Y \subseteq$ Result then Result := Result $\cup$ Z; /* transitivity */

X^+ := Result;

EXAMPLE: ATTRIBUTE CLOSURE

F = { f_1: $Z \rightarrow A$,
f_2: $B \rightarrow X$,
f_3: $AX \rightarrow Y$}

Is $ZB \rightarrow Y$ in F^+?

Equivalently, is Y in $(ZB)^+$?

Result:	ZB,	Given
	A,	f_1
	X,	f_2
	Y	f_3

Yes!

DECOMPOSITION

Let U be a relation scheme.

A set of relation schemes $\{R_1, R_2, ..., R_n\}$ is a **decomposition** of U if

$$\bigcup_{i=1}^{n} R_i = U$$

Let u be a relation on scheme U.

Let $r_i = \pi_{Ri}(u)$, for $1 \leq i \leq n$

$$u \subseteq \bowtie_{i=1}^{n} r_i$$

DESIRABLE PROPERTIES
Lossless-join Decomposition

Let C be a set of constraints on the DB. A decomposition $\{R_1, R_2, ..., R_n\}$ of a relation scheme U is a **lossless-join** decomposition if for all relations u on scheme U that are legal under C

$$u = \bowtie_{i=1}^{n} \prod_{R_i}(u)$$

Let R be a relation scheme, F be a set of FDs on R, R_1 & R_2 form a decomposition of R. This decomposition is a lossless-join decomposition of R if at least one of the following holds:

- $(R_1 \cap R_2) \rightarrow R_1$ in F^+
- $(R_1 \cap R_2) \rightarrow R_2$ in F^+

LOSSLESS-JOIN DECOMPOSITION
Suppliers Example

1. suppliers(SNAME, SADDR, ITEM, PRICE)

   R = {SNAME, SADDR, ITEM, PRICE}

   F = {f_1: SNAME$\rightarrow$SADDR, f_2: SNAME,ITEM$\rightarrow$PRICE}

   Consider the decomposition {R_1,R_2} where

   R_1 = {SNAME, SADDR}

   R_2 = {SNAME, ITEM, PRICE}

   $R_1 \cap R_2$ = {SNAME}

   $SNAME^+$ = {SNAME, SADDR} = R_1

   $\therefore$ lossless-join decomposition

   r_1 = suppaddr(SNAME, SADDR)

   r_2 = suppitem(SNAME, ITEM, PRICE)

LOSSLESS-JOIN DECOMPOSITION
Addresses Example

2. addresses(NAME, CITY, ZIP) $F=\{f_1, f_2\}$

(a) f_1: NAME→CITY
f_2: ZIP→CITY

Consider the decomposition
R_1 = {NAME, CITY}
R_2 = {CITY, ZIP}
$R_1 \cap R_2$ = {CITY}
$(CITY)^+$ = {CITY}
∴ NOT a lossless-join decomposition

(b) f_1: NAME→ZIP
f_2: ZIP→CITY

Consider the decomposition
R_1 = {NAME, ZIP}
R_2 = {CITY, ZIP}
$R_1 \cap R_2$ = {ZIP}
$(ZIP)^+$ = {ZIP, CITY} = R2
∴ lossless-join decomposition

TESTING LOSSLESS-JOIN

INPUT: R = Relation Scheme, having j attributes
D = {R1, ... , Ri} a decomposition of R
F = FDs on R

1. create matrix S with i rows and j columns
2. S(i,j) := b_ij; /* b_ij a distinct symbol */
3. for each row i /* representing a relation in D */
   for each column j /* representing an attribute of R */
   if Ri includes attribute Aj, then S(i,j) := a_j;
4. repeat until no changes
   for each FD X→Y in F
   for all rows in S which have the same symbols in the columns corresponding to attrs in X
   make these rows have the same value in Y
   (choose an “a” symbol over a “b” symbol)
5. lossless-join := exists a row having all “a” symbols;

DESIRABLE PROPERTIES
Dependency Preservation

"Update validation without computation of joins"

Let F be a set of FDs on a scheme R, and R_1, ..., R_n be a decomposition of R. The restriction of F to R_i is the set F_i of all FDs in F^+ that include only attrs of R_i. The set of restrictions F_1, F_2, ..., F_n is the set of dependencies that can be checked efficiently. Let

$$F' = \bigcup_{i=1}^{n} F_i$$

In general, $F \neq F'$.

A dependency preserving decomposition has the property that $(F')^+ = F^+$, i.e., every dependency in F is logically implied by F'.

TESTING DEPENDENCY PRESERVATION

compute F^+

For each scheme R_i in D do

F_i := the restriction of F^+ to R_i;

$$F' := \bigcup_{j=1}^{n} F_j$$

compute$(F')^+$

return$((F')^+ = F^+)$

... Not always necessary to use this algorithm, we can check that each FD in F - F' is logically implied by F'

DEPENDENCY PRESERVATION
Suppliers EXAMPLES

1. suppliers(SNAME, SADDR, ITEM, PRICE)

R = {SNAME, SADDR, ITEM, PRICE}

F = {f_1: SNAME$\rightarrow$SADDR, f_2: SNAME,ITEM$\rightarrow$PRICE}

R_1 = {SNAME, SADDR} F_1 = {f_1}

R_2 = {SNAME, ITEM, PRICE} F_2 = {f_2}

F' = F,

$\therefore$ dependency preserving

DEPENDENCY PRESERVATION
Addresses Example

2. addresses(NAME, CITY, ZIP)

(a) $F = \{f_1: NAME \rightarrow CITY,\ f_2: ZIP \rightarrow CITY\}$

$R_1 = \{NAME, ZIP\}$ $F_1 = \varnothing$

$R_2 = \{ZIP, CITY\}$ $F_2 = \{f_2\}$

$F' = \{f_2\}$ $F - F' = \{f_1\}$

Is f_1 logically implied by F' ?

No, $\therefore$ NOT dependency preserving

(b) $F = \{f_1: NAME \rightarrow ZIP,\ f_2: ZIP \rightarrow CITY\}$

$R_1 = \{NAME, ZIP\}$ $F_1 = \{f_1\}$

$R_2 = \{CITY, ZIP\}$ $F_2 = \{f_2\}$

$F' = F$, $\therefore$ dependency preserving

Note:

1. A decomposition may have a lossless-join w.r.t. F yet not preserve F.
2. A decomposition could preserve F yet not have a lossless-join.

NORMAL FORMS
Boyce-Codd Normal Form (BCNF)

A relation scheme is in BCNF if for all FDs that hold on R of the form $X \rightarrow Y$, where $X \subseteq R$ & $Y \subseteq R$, at least one of the following holds:

- $X \rightarrow Y$ is a trivial FD
- X is a superkey for scheme R

Example: R = {EMP, PHONE, PROJECT, HRS}

F = {EMP→PHONE, EMP PROJECT→HRS,
PHONE→EMP, PHONE PROJECT→HRS}

candidate keys: EMP PROJECT and PHONE PROJECT

not BCNF: EMP/PHONE not superkey

BCNF: R_1 = {EMP, PHONE} F_1 = {EMP→PHONE, PHONE→EMP}

R_2 = {EMP, PROJECT, HRS} F_2 = {EMP,PROJECT→HRS}

BCNF DECOMPOSITION ALGORITHM

Let R be a given relation scheme, F be the given set of FDs on R.

Result := {R};

Done := FALSE;

compute F^+;

while (not DONE) do

  if there is a scheme R_i in Result that is not in BCNF

    {Let $X \rightarrow Y$ be a nontrivial FD on R_i s.t.

      $X \rightarrow R_i$ is not in F^+ /* not already in BCNF */

    Result := (Result - R_i) $\cup$ (R_i - Y) $\cup$ (X $\cup$ Y) }

  else

    DONE := TRUE;

Note: The order in which the Fds are chosen for decomposition affects the resulting schema.

BCNF DECOMPOSITION ALGORITHM

Guarantees Lossless-join property

$$\text{Result} := (\text{Result} - Ri) \cup (Ri - Y) \cup (X \cup Y)$$

The new relations added at each step are:

$$(Ri - Y) \;\&\; (X \cup Y)$$

Lossless-join check on new relations:

$$(Ri - Y) \cap (X \cup Y) = \{X\}$$

We know that $X \rightarrow Y$.

The attribute closure of X: $(X)^+ \rightarrow XY$

functionally determines the scheme of $(X \cup Y)$

$\therefore$ lossless-join decomposition

However, not every BCNF decomposition is dependency preserving.

BCNF & DEPENDENCY PRESERVATION

Counter-example

Let S = {J, K, L}

F = {JK→L, L→K}

S is not in BCNF since L→K but L is not a superkey.

$(L)^+$ = {L, K}, missing J.

BCNF decomposition of S using L→K:

S1 = {L, K} L→K

S2 = {J, L}

S1 & S2 preserve only L→K (& trivial dependencies)

but closure of {L→K} does not include JK→L

∴ not dependency preserving, but BCNF and lossless-join