Normalization

Multivalued Dependencies (MVDs)

- **Persons**(*Man*(*M*), *Phone*(*P*), *Dogs_Like*(*D*))

Person:			**Meaning of the tuples**
Man (M)	**Phone (P)**	**Dogs_Like (D)**	Man M have phones P and likes the dog D
M1	P1/P2	D1/D2	M1 have phones P1 and P2, and likes the dogs D1 and D2
M2	P3	D2	M2 have phones P3, and likes the dog D2
Key: MPD			

- There are no non-trivial FDs because all tributes are combined forming *candidate key*, i.e., MDP
- In the above relation, two multivalued dependency exists:
  - *Man* →→ *Phones*
  - *Man* →→ *Dogs_Like*
- A Man's phone are independent of the dogs they like
- But, after converting the above relation in *single valued attribute*, each of a man's phones appears with each of the dogs they like in all combinations

Post 1NF normalization		
Man (M)	**Phone (P)**	**Dogs_Like (D)**
M1	P1	D1
M1	P2	D2
M2	P3	D2
M2	P3	D2
M1	P1	D2
M1	P2	D1

Multivalued Dependencies (MVDs)

- If two or more independent relations are kept in a single relation, then *multivalued dependency* is possible
- For example, let there are two relations:
  - **Student(**SID, Sname**)** where *SID* → *Sname*
  - **Course(**CID, Cname**)** where *CID* → *Cname*

- There is no relation defined between Student and Course
- If we kept them in a single relations, named **Student_Course**, then MVD will exists because of *m:n cardinality*

- If two or more *MVDs* exists in a relation, then while converting into *SVAs*, *MVDs* exits

Student:

SID	Sname
S1	A
S2	B

Course:

CID	Cname
C1	C
C2	B

SID	Sname	CID	Cname
S1	A	C1	C
S1	A	C2	B
S2	B	C1	C
S2	B	C2	B

Multivalued Dependencies (MVDs)

- Suppose we record names of children, and phone numbers for instructors:
  - *inst_child*(*ID*, *child_name*)
  - *inst_phone*(*ID*, *phone_number*)
- If we were to combine these schemas to get
  - *inst_info*(*ID*, *child_name*, *phone_number*)
  - Example data:

    (99999, David, 512-555-1234)
    (99999, David, 512-555-4321)
    (99999, William, 512-555-1234)
    (99999, William, 512-555-4321)
- This relation is in BCNF
  - Why?

Multivalued Dependencies (MVDs)

- Let R be a relation schema and let $\alpha \subseteq R$ and $\beta \subseteq R$. The **multivalued dependency**

$$\alpha \to\to \beta$$

holds on R if in any legal relation $r(R)$, for all pairs for tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$, there exist tuples t_3 and t_4 in r such that:

$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$
$t_3[\beta] = t_1[\beta]$
$t_3[R - \beta] = t_2[R - \beta]$
$t_4[\beta] = t_2[\beta]$
$t_4[R - \beta] = t_1[R - \beta]$

Example:
A relation of university courses, the books recommended for the course, and the lectures who will be teaching the course:
course $\to\to$ *book*
course $\to\to$ *lecturer*

Course	Book	Lecturer	Tuples
AHA	Silberschatz	John D	t1
AHA	Nederpelt	William M	t2
AHA	Silberschatz	William M	t3
AHA	Nederpelt	John D	t4
AHA	Silberschatz	Christian G	
AHA	Nederpelt	Christian G	
OSO	Silberschatz	John D	
OSO	Silberschatz	William M	

MVD: Tabular Representation

- Tabular representation of $\alpha \rightarrow\rightarrow \beta$

	α	β	$R - \alpha - \beta$
t_1	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$a_{j+1} \ldots a_n$
t_2	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$b_{j+1} \ldots b_n$
t_3	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$b_{j+1} \ldots b_n$
t_4	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$a_{j+1} \ldots a_n$

MVD

- Let R be a relation schema with a set of attributes that are partitioned into 3 nonempty subsets

$$Y, Z, W$$

- We say that $Y \rightarrow\rightarrow Z$ (Y **multidetermines** Z) if and only if for all possible relations r (R)

$$< y_1, z_1, w_1 > \in r \text{ and } < y_1, z_2, w_2 > \in r$$

- Then

$$< y_1, z_1, w_2 > \in r \text{ and } < y_1, z_2, w_1 > \in r$$

- Note that since the behavior of Z and W are identical it follows that

$$Y \rightarrow\rightarrow Z \text{ if } Y \rightarrow\rightarrow W$$

Example

- In our example:

$$ID \rightarrow\rightarrow child\_name$$
$$ID \rightarrow\rightarrow phone\_number$$

- The above formal definition is supposed to formalize the notion that given a particular value of Y (ID) it has associated with it a set of values of Z *(child_name)* and a set of values of W *(phone_number)*, and these two sets are in some sense independent of each other

- **Note:**
  – If $Y \rightarrow Z$ then $Y \rightarrow\rightarrow Z$
  – Indeed we have (in above notation) $Z_1 = Z_2$
  The claim follows

Use of Multivalued Dependencies

- We use multivalued dependencies in two ways:
  - To test relations to **determine** whether they are legal under a given set of functional and multivalued dependencies
  - To specify **constraints** on the set of legal relations
  - We shall concern ourselves *only* with relations that satisfy a given set of functional and multivalued dependencies

- If a relation r fails to satisfy a given multivalued dependency, we can construct a relations r' that does satisfy the multivalued dependency by adding tuples to r

Theory of MVDs

	Name		Rule
C-	**Complementation**	**:**	If $X \rightarrow\rightarrow Y$, then $X \rightarrow\rightarrow \{\mathbf{R} - (X \cup Y)\}$
A-	**Augmentation**	**:**	If $X \rightarrow\rightarrow Y$ and $W \supseteq Z$, then $WX \rightarrow\rightarrow YZ$
T-	**Transitivity**	**:**	If $X \rightarrow\rightarrow Y$ and $Y \rightarrow\rightarrow Z$, then $X \rightarrow\rightarrow (Z - Y)$
	Replication	**:**	If $X \rightarrow Y$ and $X \rightarrow\rightarrow Y$ but the reverse is not true
	Coalescence	**:**	If $X \rightarrow\rightarrow Y$ and there is a W such that $W \cap Y$ is empty, $W \rightarrow Z$, and $Y \supseteq Z$, then $X \rightarrow Z$

- A MVD $X \rightarrow\rightarrow Y$ in **R** is called a trivial MVD is:
  - Y is a subset of X ($X \supseteq Y$) or
  - $X \cup Y = \mathbf{R}$
  - Otherwise, it is a non-trivial *MVD* and we have to repeat values redundancy in the tuples

Theory of MVDs

- From the definition of multivalued dependency, we can derive the following rule:
  - If $\alpha \rightarrow \beta$, then $\alpha \rightarrow\rightarrow \beta$

  That is, every functional dependency is also a multivalued dependency

- The **closure** D^+ of *D* is the set of all functional and multivalued dependencies logically implied by *D*
  - We can compute D^+ from *D*, using the formal definitions of functional dependencies and multivalued dependencies
  - We can manage with such reasoning for very simple multivalued dependencies, which seem to be most common in practice
  - For complex dependencies, it is better to reason about sets of dependencies using a system of inference rules

Fourth Normal Form

- A relation schema R is in **4NF** with respect to a set D of functional and multivalued dependencies if for all multivalued dependencies in $D+$ of the form $\alpha \rightarrow\rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following hold:
  - $\alpha \rightarrow\rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$)
  - α is a *superkey* for schema R

- If a relation is in 4NF, it is in BCNF

Restriction of Multivalued Dependencies

- The restriction of D to R_i is the set D_i consisting of
  - All functional dependencies in D^+ that include only attributes of R_i
  - All multivalued dependencies of the form

$$\alpha \rightarrow\rightarrow (\beta \cap R_i)$$

where $\alpha \subseteq R_i$ and $\alpha \rightarrow\rightarrow \beta$ is in D^+

4NF Decomposition Algorithm

- For all dependencies $A \rightarrow\rightarrow B$ in D^+, check if A is a *superkey*
  - By using attribute closure

- If not, then
  - Choose a dependency in F^+ that breaks the 4NF rules, say $A \rightarrow\rightarrow B$
  - Create $\mathbf{R_1} = AB$
  - Create $\mathbf{R_2} = A\ (\mathbf{R} - (B - A))$
  - Note that: $\mathbf{R_1} \cap \mathbf{R_2} = A$ and $A \rightarrow\rightarrow AB$ (=$\mathbf{R_1}$), so this is lossless decomposition

- Repeat for $\mathbf{R_1}$ and $\mathbf{R_2}$
  - By defining D_1^+ to be all dependencies in F that contain only attributes in $\mathbf{R_1}$
  - Similarly D_2^+

4NF Decomposition Algorithm

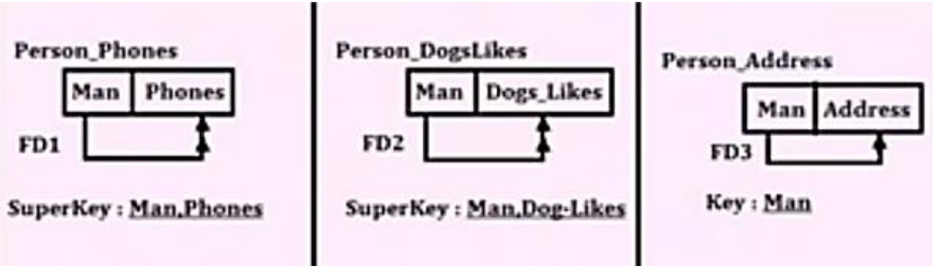
result := {*R*};
done := false;
compute D^+;
Let D_i denote the restriction of D^+ to R_i
while (**not** *done*)
  if (there is a schema $\mathbf{R}_i$ in *result* that is not in 4NF) **then**
    begin
      let $\alpha \rightarrow\rightarrow \beta$ be a nontrivial multivalued dependency that holdson R_i such that $\alpha \rightarrow R_i$
        is not in D_i, and $\alpha \cap \beta = \phi$;
      result := (*result* - R_i) $\cup$ (R_i - β) $\cup$ (α, β);
  end
  else *done*:= true;

Note: Each R_i is in 4NF, and decomposition is lossless-join

Example of 4NF Decomposition

- Example:
- **Persons_Modify**(*Man*(*M*), *Phone*(*P*), *Dogs_Like*(*D*), *Address*(*A*))
- *FDs*:
  - FD1: *Man* $\twoheadrightarrow$ *Phones*
  - FD2: *Man* $\twoheadrightarrow$ *Dogs_Like*
  - FD3: *Man* $\rightarrow$ *Address*
- Key: *MPD*
- All dependencies violate 4NF

Post normalization



Man (M)	Phone (P)	Dogs_Like (D)	Address (A)
M1	P1	D1	49-ABC, Bhiwani (HR.)
M1	P2	D2	49-ABC, Bhiwani (HR.)
M2	P3	D2	36-XYZ, Rohtak (HR.)
M1	P1	D2	49-ABC, Bhiwani (HR.)
M1	P2	D1	49-ABC, Bhiwani (HR.)

- In the above relations for both the MVD's, **"*X*" is Man**, which is again not the *superkey*, but as *X* ∪ *Y* = **R**, i.e., (*Man* & *Phone*) together make thee relations

- So, the above MVDs are trivial and in FD3, *Address* is functionally dependent on *Man*, where *Man* is the key in **Person_Address**, hence all the three relations are in 4NF

Example

- $R = (A, B, C, G, H, I)$
- $F = \{ A \rightarrow\rightarrow B$
  $B \rightarrow\rightarrow HI$
  $CG \rightarrow\rightarrow H \}$

- R is not in 4NF since $A \rightarrow\rightarrow B$ and A is not a superkey for R Decomposition

a) $R_1 = (A, B)$ (R_1 is in 4NF)
b) $R_2 = (A, C, G, H, I)$ (R_2 is not in 4NF, decompose into R_3 and R_4)
c) $R_3 = (C, G, H)$ (R_3 is in 4NF)
d) $R_4 = (A, C, G, I)$ (R_4 is not in 4NF, decompose into R_5 and R_6)

- $A \rightarrow\rightarrow B$ and $B \rightarrow\rightarrow HI$ ➔ $A \rightarrow\rightarrow HI$, (MVD transitivity)
- And hence $A \rightarrow\rightarrow I$ *(MVD restriction to R_4)*

e) $R_5 = (A, I)$ (R_5 is in 4NF)
f) $R_6 = (A, C, G)$ (R_6 is in 4NF)

Design Goals

- Goal for a relational database design is:
  - BCNF/ 4NF
  - Lossless join
  - Dependency preservation

- If we cannot achieve this, we accept one of the following:
  - Lack of dependency preservation
  - Redundancy due to use of 3NF

- Interestingly, SQL does not provide a direct way of specifying functional dependencies other than superkeys

- Can specify FDs using assertions, but they are expensive to test, (and currently not supported by any of the widely used databases!)

- Even if we had a dependency preserving decomposition, using SQL we would not be able to efficiently test a functional dependency whose left hand side is not a key

Further Normal Forms

- **Further NFs**
  - Elementary key normal form (EKNF)
  - Essential tuple normal form (ETNF)
  - Join dependencies and Fifth normal form (5NF)
  - Sixth normal form (6NF)
  - Domain/key normal form (DKNF)

- **Join dependencies** generalize multivalued dependencies
  - Lead to **project-join normal form (PJNF)** (also called **fifth normal form**)

- A class of even more general constraints, leads to a normal form called **domain-key normal form**

- Problem with these generalized constraints: are hard to reason with, and no set of sound and complete set of inference rules exists

- Hence rarely used

Overall Database Design Process

We have assumed schema *R* is given

- *R* could have been generated when converting ER diagram to a set of tables
- *R* could have been a single relation containing *all* attributes that are of interest (called **universal relation**)
- Normalization breaks *R* into smaller relations
- *R* could have been the result of some ad hoc design of relations, which we then test/convert to normal form

ER Model and Normalization

- When an ER diagram is carefully designed, identifying all entities correctly, the tables generated from the ER diagram should not need further normalization

- However, in a real (imperfect) design, there can be functional dependencies from non-key attributes of an entity to other attributes of the entity
  - Example: An *employee* entity with
    - Attributes: *department_name* and *building*
    - Functional dependency: *department_name* $\rightarrow$ *building*

- Good design would have made department an entity

- Functional dependencies from non-key attributes of a relationship set possible, but rare --- most relationships are binary

Denormalization for Performance

- May want to use non-normalized schema for performance

- For example, displaying *prereqs* along with *course_id,* and *title* requires join of *course* with *prereq*
  - **Course**(*course_id, title,…*)
  - **Prerequisite**(*course_id, prereq*)

- Alternative 1: Use denormalized relation containing attributes of *course* as well as *prereq* with all above attributes: **Course**(*course_id, title, prereq*)
  - Faster lookup
  - Extra space and extra execution time for updates
  - Extra coding work for programmer and possibility of error in extra code

- Alternative 2: Use a materialized view defined a *course prereq*
  - **Course** $\bowtie$ **Prerequisite**
  - Benefits and drawbacks same as above, except no extra coding work for programmer and avoids possible errors

Other Design Issues

- Some aspects of database design are not caught by normalization
- Examples of bad database design, to be avoided:
  Instead of *earnings* (*company_id, year, amount*), use
  - *earnings_2004, earnings_2005, earnings_2006*, etc., all on the schema (*company_id, earnings*)
    - Above are in BCNF, but make querying across years difficult and needs new table each year
  - *company_year* (*company_id, earnings_2004, earnings_2005, earnings_2006*)
    - Also in BCNF, but also makes querying across years difficult and requires new attribute each year
    - Is an example of a **crosstab**, where values for one attribute become column names
    - Used in spreadsheets, and in data analysis tools

Modeling Temporal Data

- **Temporal data** have an association time interval during which the data are *valid*
- A **snapshot** is the value of the data at a particular point in time
- Several proposals to extend ER model by adding valid time to
  - Attributes, e.g., Address of an instructor at different points in time
  - Entities, e.g., Time duration when a student entity exists
  - Relationships, e.g., Time during which an instructor was associated with a student as an advisor
- But no accepted standard
- Adding a temporal component results in functional dependencies like

  $$ID \rightarrow street, city$$

  not holding, because the address varies over time
- A **temporal functional dependency** $X \rightarrow Y$ holds on schema R if the functional dependency $X \rightarrow Y$ holds on all snapshots for all legal instances $r(R)$

Modeling Temporal Data

- In practice, database designers may add start and end time attributes to relations
  - E.g., *course*(*course_id, course_title*) is replaced by

    course(*course_id, course_title, start, end*)
  - Constraint: No two tuples can have overlapping valid times
    - Hard to enforce efficiently

- Foreign key references may be to current version of data, or to data at a point in time
  - E.g., Student transcript should refer to course information at the time the course was taken

Extra

Correctness of 3NF Decomposition Algorithm

- 3NF decomposition algorithm is dependency preserving (since there is a relation for every *FD* in F_c)
- Decomposition is lossless
  - A candidate key (C) is in one of the relations R_i in decomposition
  - Closure of candidate key under F_c must contain all attributes in R
  - Follow the steps of attribute closure algorithm to show there is only one tuple in the join result for each tuple in R_i

Correctness of 3NF Decomposition Algorithm

- Claim: If a relation R_i is in the decomposition generated by the above algorithm, then R_i satisfies 3NF
- Proof:
  - Let R_i be generated from the dependency $\alpha \rightarrow \beta$
  - Let $\gamma \rightarrow$ B be any non-trivial functional dependency on R_i

    (We need only consider FDs whose right-hand side is a single attribute)
  - Now, B can be in either β or α but not in both
  - Consider each case separately

Correctness of 3NF Decomposition Algorithm

- Case 1: If B in β:
  - If γ is a superkey, the 2nd condition of 3NF is satisfied
  - Otherwise α must contain some attribute not in γ
  - Since $\gamma \rightarrow B$ is in F^+ it must be derivable from F_c, by using attribute closure on γ
  - Attribute closure not have used $\alpha \rightarrow \beta$
  - If it had been used, α must be contained in the attribute closure of γ, which is not possible, since we assumed γ is not a superkey
  - Now, using $\alpha \rightarrow (\beta - \{B\})$ and $\gamma \rightarrow B$, we can derive $\alpha \rightarrow B$ (since $\gamma \subseteq \alpha\, \beta$, and B $\notin \gamma$ since $\gamma \rightarrow B$ is non-trivial)
  - Then, B is extraneous in the right-hand side of $\alpha \rightarrow \beta$; which is not possible since $\alpha \rightarrow \beta$ is in F_c
  - Thus, if B is in β then γ must be a superkey, and the second condition of 3NF must be satisfied

Correctness of 3NF Decomposition Algorithm

- Case 2: *B* is in α
  - Since α is a candidate key, the third alternative in the definition of 3NF is trivially satisfied
  - In fact, we cannot show that γ is a superkey
  - This shows exactly why the third alternative is present in the definition of 3NF

Next Lecture

Transactions

Thank you for your attention...

Any question?

Contact:
Department of Information Technology, NITK Surathkal, India
6th Floor, Room: 13
Phone: +91-9477678768
E-mail: shrutilipi@nitk.edu.in