Lecture 13
Functional Dependencies

The Evils of Redundancy

- *Redundancy* is at the root of several problems associated with relational schemas:
  - *Redundant storage, insert/delete/update anomalies*
- *Functional dependencies* can be used to identify schemas with such problems and to suggest refinements

Functional Dependencies

- Most important kind of constraint in the relational model
- Knowledge about functional dependencies is vital for redesign of database schemas to eliminate anomalies & redundancies

Functional Dependencies (FDs)

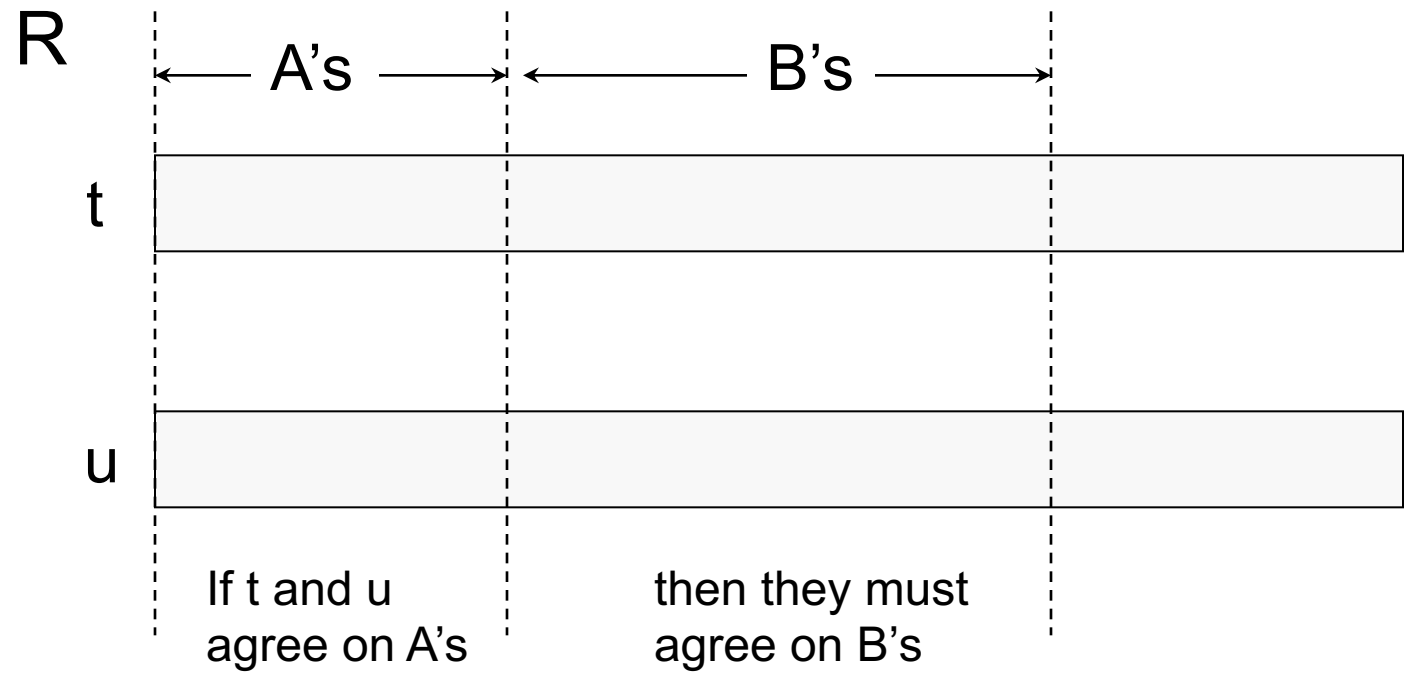
- A functional dependency $X \rightarrow Y$ holds over relation R, if for every allowable instance r of R:
  - $t_1 \in r$, $t_2 \in r$, $t_1[X] = t_2[X]$ implies $t_1[Y] = t_2[Y]$
  - i.e., given two tuples in r, if the X values agree, then the Y values must also agree (X and Y are sets of attributes)

Formal Definition

- Formally: $A \rightarrow B$
  - "A functionally determines B"
  - "B is functionally dependent on A"
  - A is called the left-hand side (LHS) of the FD
  - B is called the right-hand side (RHS) of the FD
- If two tuples of r(R) agree on a set of attributes A of R, then they must also agree on another set of attribute B
  - i.e., the tuples have the same values in their respective components for each of these attributes

Pictorially Speaking...

- Assume A $\rightarrow$ B



- A FD tells us about *any* two tuples *t* and *u* in relation R

Example 1

Movies

title	*year*	*length*	*filmType*	*studioName*	*starName*
Star Wars	1977	124	color	Fox	Carrie Fisher
Star Wars	1977	124	color	Fox	Mark Hamill
Star Wars	1977	124	color	Fox	Harrison Ford
Mighty Ducks	1991	104	color	Disney	Emilio Estevez
Wayne's World	1992	95	color	Paramount	Dana Carvey
Wayne's World	1992	95	color	Paramount	Mike Myers

FDs that might hold:

{title, year} → length

{title, year} → filmType

{title, year} → studioName

FD that doesn't hold:

{title, year} → starName

What are functional dependencies associated with?

Relational schema

A particular instance

What are functional dependencies associated with?

Relational schema

0%

A particular instance

0%

What are functional dependencies associated with?

Relational schema

0%

A particular instance

0%

Example 2

- Consider relation obtained from Hourly_Emps:
  - Hourly_Emps (*ssn, name, lot, rating, hrly_wages*, *hrs_worked*)
- We will denote this relation schema by listing the attributes as SNLRWH, which is the *set* of attributes {S,N,L,R,W,H}
- Some FDs on Hourly_Emps:
  - *ssn* is the key: S $\rightarrow$ SNLRWH
  - *rating* determines *hrly_wages*: R $\rightarrow$ W

Example 2 (Cont.)

- Problems due to R $\rightarrow$ W :
  - Update anomaly: Can we change W in just the 1st tuple of SNLRWH?

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Example 2 (Cont.)

- Insertion anomaly: If we insert an employee with R=5, we must enter the values of W correctly
- Deletion anomaly: If we delete all employees with rating 5, we lose the information about the wage for rating 5!

Hourly_Emps1

S	N	L	R	H
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40

Hourly_Emps2

R	W
8	10
5	7

FDs and Schema

- FDs are assertions about the schema of a relation, NOT about a particular instance
  - Must be identified based on semantics of application
  - All data must conform
  - Given some allowable instance r_1 of R, we can check if it violates some FD *f*, but we cannot tell if *f* holds over R!
- Relation extensions r(R) that satisfy the FD constraints are called legal relation states or legal extensions

FDs and Keys

- $K=\{A_1, A_2,\ldots, A_n\}$ is a candidate key for relation R if:
  - K functionally determines ALL other attributes of R

  AND
  - No proper subset of K functionally determines all other attributes of R

Trivial Dependencies

- A functional dependency $\{A_1,A_2,\ldots,A_n\} \rightarrow B$ is trivial if B is one of the A's
  - E.g., {title, year} $\rightarrow$ title
- Every trivial dependency holds in every relation
- We may assume trivial dependencies without having to justify them
  - Trivial: B is a subset of A's
  - Nontrivial: at least one of the B's is not among A's
  - Completely nontrivial: none of the B's is part of A's

Reasoning About FDs

- Given some FDs, we can usually infer additional FDs:
  - {*ssn* $\rightarrow$ *did*, *did* $\rightarrow$ lot} implies {*ssn* $\rightarrow$ *lot*}
- An FD *f* is implied by a set of FDs *F* if *f* holds whenever all FDs in *F* hold
  - The closure of *F* (F^+): the set of all FDs that are implied by *F*

Inference Rules

- Armstrong's Axioms (X, Y, Z are sets of attributes):
  - Reflexive rule: If Y is a subset of X, then $X \rightarrow Y$
  - Augmentation rule: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
  - Transitive rule: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- These are *sound* and *complete* inference rules for FDs

Additional Inference Rules

- Union rule: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
- Decomposition rule: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
- Pseudotransitive rule: If $X \rightarrow Y$ and $WY \rightarrow Z$, then $WX \rightarrow Z$

Closure of Attributes

- Given a relation R, a set of FDs for R (denoted by F), and a set of attributes $\{A_1, A_2, ..., A_m\}$ of R (denoted by X):
  - Find all attributes B in R such that $\{A_1, A_2, ..., A_m\} \rightarrow B$
  - B is called the closure of X under F and is denoted by $\{A_1, A_2, ..., A_m\}^+$

Algorithm for Computing Closure of Attributes

- Start with $\{A_1, A_2, ..., A_m\}$
- Repeat until no change:
  - If current set of attributes includes LHS of a dependency, add RHS attributes to the set

Example: Calculating Attribute Closure

- R=(A, B, C, D, E, G)
- F: {AB $\rightarrow$ C, BC $\rightarrow$ AD, D $\rightarrow$ E, CE $\rightarrow$ B}
- How to compute the closure of {A, B} under F, i.e., $\{A, B\}^+$?
  - Start with X={A, B}
  - Add C to X due to AB $\rightarrow$ C; X={A, B, C}
  - Add A,D to X due to BC $\rightarrow$ AD; X={A, B, C, D}
  - Add E to X due to D $\rightarrow$ E; X={A, B, C, D, E}
  - No more attributes can be added to X
  - $\{A, B\}^+$ = {A, B, C, D, E}

Attribute Closure Usage

- If we can compute closure of any set of attributes, we can test whether any given functional dependency $\{A_1, A_2,\ldots, A_n\} \rightarrow B$ follows from a set of dependencies F
  - Compute closure of $\{A_1, A_2, \ldots, A_n\}^+$ using F
  - If B is in $\{A_1, A_2, \ldots, A_n\}^+$, then $\{A_1, A_2, \ldots, A_n\} \rightarrow B$ does follow from F
- We can also test if $K=\{A_1, A_2,\ldots, A_n\}$ is a candidate key for relation R
  - If $\{A_1, A_2,\ldots, A_n\}^+$ is the set of *all* attributes in R and if K is minimal

Specifying FDs for a Relation

- Let F be set of FDs specified on R
- Must be able to *reason* about FDs in F
  - Designer usually explicitly states only FDs which are obvious
  - Without knowing exactly what all tuples are, must be able to deduce other/all FDs that hold on R
  - Essential when we discuss design of “good” relational schemas
- How can we tell if one FD follows from others?
  - Use Armstrong’s axioms and reason it out
  - OR use attribute closure algorithm