Today's Lecture

- Designing relational schemas
  - Anomalies caused by data redundancies
  - Functional Dependencies
  - Reasoning about FDs
  - Normal Forms

Recommended Readings

- Chapter 19
  - All sections up to and including Section 19.4
- Silberschatz, Korth, Sudarshan
  - Section 7.3.4

Schema design

- Recall that conceptual database design from ER diagrams gives
  - A set of relation schemas
  - A set of integrity constraints
- But they are not good enough. Why?
  - Integrity constraints are usually not taken into full account in ER designs
- Typical schema design steps
  - Conceptual database design (through the use of ER diagrams)
  - Schema Refinement through the use of ICs
  - Typically performance criteria and workload information are also taken into account. Redundancy vs. Efficiency tradeoffs

Example

- If we know that rank determines the salary scale, which is a better design? Why?
- Employee(<u>eid</u>, name, addr, rank, salary-scale)
- Employee(<u>eid</u>, name, addr, rank)
- Salary-Scale(<u>rank</u>, salary-scale)

Lots of Duplicates

eid	name	addr	rank	salary-scale
34-133	Jane	Elm St.	6	70-90
33-112	Hugh	Pine St.	3	30-40
26-002	Gary	Elm St.	4	35-50
51-994	Ann	South St.	4	35-50
45-990	Jim	Main St.	6	70-90
98-762	Paul	Walnut St.	4	35-50

- Lots of duplicate information
  - Employees who have the same rank have the same salary scale

Update Anomaly

eid	name	addr	rank	salary-scale
34-133	Jane	Elm St.	6	70-90
33-112	Hugh	Pine St.	3	30-40
26-002	Gary	Elm St.	4	35-50
51-994	Ann	South St.	4	35-50
45-990	Jim	Main St.	6	70-90
98-762	Paul	Walnut St.	4	35-50

- Update anomaly
  - If one copy of salary scale is changed, all copies of salary scale (of the same rank) have to be changed

Insertion Anomaly

eid	name	addr	rank	salary-scale
34-133	Jane	Elm St.	6	70-90
33-112	Hugh	Pine St.	3	30-40
26-002	Gary	Elm St.	4	35-50
51-994	Ann	South St.	4	35-50
45-990	Jim	Main St.	6	70-90
98-762	Paul	Walnut St.	4	35-50

- Insertion anomaly
  - How can we store a new rank and salary scale information if no employee has that rank?
  - Use NULLS?

Deletion Anomaly

eid	name	addr	rank	salary-scale
34-133	Jane	Elm St.	6	70-90
33-112	Hugh	Pine St.	3	30-40
26-002	Gary	Elm St.	4	35-50
51-994	Ann	South St.	4	35-50
45-990	Jim	Main St.	6	70-90
98-762	Paul	Walnut St.	4	35-50

- Deletion anomaly
  - If Hugh is deleted, how can we retain the rank and salary scale information?
  - Use NULLS?

What is a good design?

- Intuitively, salary scale is dependent only on rank and therefore, making the associations between employee information such as name, addr with salary-scale is unnatural and causes redundancy
- Based on the constraints given, we would like to refine the schema so that such redundancies cannot occur
- However, sometimes we may choose to live with redundancy in order to improve query performance. Ultimately, a good design is depends on the query workload

Functional Dependencies

- The information that rank determines salary-scale is a type of integrity constraint known as functional dependencies
- Functional dependencies can help us detect anomalies that may exist in a given schema
- The FD rank$\rightarrow$ salary-scale suggests that Employee(eid, name, addr, rank, salary-scale) should be decomposed into two relations:
  - Employee(eid, name, addr, rank)
  - Salary-Scale(rank, salary-scale).

Meaning

- We have seen a kind of functional dependency before
  - Emp(<u>ssn</u>, name, addr) (Key)
  - If two tuples agree on the ssn value, then they must also agree on the name and address values. (ssn $\rightarrow$ name, addr)
- Let **R** be a relation schema. A functional dependency (FD) is an integrity constraint of the form

  $X \rightarrow Y$

  where X and Y are non-empty subsets of attributes of **R**.
- A relation instance R of **R** satisfies the FD $X \rightarrow Y$ if

  for every pair of tuples t and t' in R, if t.X = t'.X, then t.Y = t'.Y

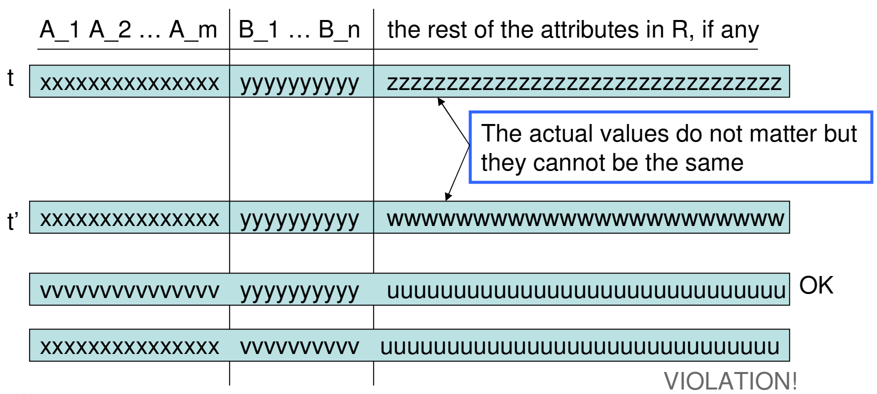
Denotes the X value(s) of tuple t, i.e., project t on the attributes in X. Alternatively, you can write as t[X]

Meaning

- X$\rightarrow$ Y ("X functionally determines Y")
  - If two tuples agree on the X attributes, they must also agree on the Y attributes
  - The above must hold for every possible pair of tuples in a relation R if R satisfies $X \rightarrow Y$
  - (see next picture)
- An FD is a statement about all possible legal instances of a schema. We cannot look at an instance to determine which FDs hold (although we can tell which FDs are not satisfied)

Illustration of a FD

- Relation Schema R with the FD $A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n$ where $\{ A_1, \ldots, A_m, B_1, \ldots, B_n \} \subseteq$ attributes(R)



Example

- Emp(ssn, name, addr)
- If X $\rightarrow$ Y and Y is all the attributes in R, then X is a superkey of R.
  - ssn, name $\rightarrow$ ssn, name, addr
- If X$\rightarrow$ Y, Y is all the attributes in R, and X is minimal, then X is the key of R
  - ssn $\rightarrow$ ssn, name, addr

- Other "trivial" FDs
  - addr $\rightarrow$ addr
  - name, addr $\rightarrow$ addr

Example

- But an FD is more general than a key constraint
- AB $\rightarrow$ C (note that AB is not a key or superkey of the relation)

A	B	C	D
a1	b1	c1	d1
a1	b1	c1	d2
a1	b2	c2	d2
a2	b1	c3	d1
a2	b1	c3	d2

Reasoning about FDs

- R(A,B,C,D,E)
- Suppose A $\rightarrow$ C and C $\rightarrow$ E, is it also true that A $\rightarrow$ E ?

- Given any instance that satisfies A $\rightarrow$ C and C $\rightarrow$ E, this means that
  - For any two tuples, if they agree on the A value, they also agree on the C value (by A $\rightarrow$ C)
  - If two tuples agree on the C value, they also agree on the E value (by C $\rightarrow$ E)
  - Therefore, for any two tuples, if they agree on the A value, they agree on the E value. A $\rightarrow$ E

Implication of FDs

- An FD F is implied by a given set $\mathcal{F}$ of FDs (or "$\mathcal{F}$ implies F") if for every instance that satisfies $\mathcal{F}$, F is also satisfied
  Notation: $\mathcal{F} \models F$
- Note that it is not sufficient if only for some instance that satisfies $\mathcal{F}$, F is also satisfied
- How can we determine whether $\mathcal{F}$ implies F?

Armstrong's Axioms

- Let X, Y, and Z denote sets of attributes over a relation schema R
- **Reflexivity:** If $Y \subseteq X$, then $X \rightarrow Y$
  ssn, name → name
  - FDs in this category are called trivial FDs
- **Augmentation**: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
  ssn, name, addr → name addr
- **Transitivity**: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
  ssn → rank and rank → sal-scale
  Then ssn → sal-scale

Additional Rules

- **Union**: If X → Y and X → Z, then X → YZ
- **Decomposition**: If X → YZ, then X → Y and X → Z

- These rules are not essential (they can be derived) but useful
- Proof of Union Rule:
  X → Z implies XY → YZ (augmentation)
  X → Y implies X → XY (augmentation)
  Therefore, X → YZ (transitivity)

Additional Rules

- Proof of Decomposition Rule:
  YZ → Y (reflexivity)
  YZ → Z (reflexivity)
  Therefore, X → Y and X → Z (transitivity)
- We use the notation $\mathcal{F} \vdash F$ to mean that F can be derived from $\mathcal{F}$ using Armstrong's axioms

- **Pseudotransitivity**: If X → Y and WY → Z, then XW → Z
- Try to prove this!

Armstrong's Axioms

- Completeness: If a set $\mathcal{F}$ of FDs implies F, then F can be derived from $\mathcal{F}$ by applying Armstrong's axioms
  If $\mathcal{F} \models$ F, then $\mathcal{F} \vdash$ F

  semantic   syntactic
- Soundness: If F is derived from a set $\mathcal{F}$ of FDs through Armstrong's axioms, then $\mathcal{F}$ implies F
  If $\mathcal{F} \vdash$ F, then $\mathcal{F} \models$ F
- In other words, Armstrong's axioms derive exactly all the FDs that should hold under $\mathcal{F}$
- Still, how can we decide if $\mathcal{F}$ implies F?

Closure of FDs

- $\mathcal{F}^+$: The set of all FDs implied by a given set $\mathcal{F}$ of FDs. Also called the closure of $\mathcal{F}$.
- To decide if $\mathcal{F}$ implies F, compute $\mathcal{F}^+$ and check if an FD F $\in \mathcal{F}^+$
- Compute $\mathcal{F}^+$ for the set { A $\rightarrow$ B, B $\rightarrow$ C} of FDs.
- Trivial dependencies
  - A $\rightarrow$ A, B $\rightarrow$ B, C $\rightarrow$ C, AB $\rightarrow$ A, AB $\rightarrow$ B, BC $\rightarrow$ B, BC $\rightarrow$ C, AC $\rightarrow$ A, AC $\rightarrow$ C, ABC $\rightarrow$ A, ABC $\rightarrow$ B, ABC $\rightarrow$ C
- Augmentation & transitivity (non-trivial dependencies)
  - AC $\rightarrow$ B, AB $\rightarrow$ C
- Transitivity
  - A $\rightarrow$ C

Expensive and tedious!

Attribute Closure

- Let X be a set of attributes
- Attribute closure X^+ with respect to a set $\mathcal{F}$ of FDs is the set of all attributes A such that $X \to A$ is derivable from $\mathcal{F}$

  Input: A set of attributes X and a set of FDs $\mathcal{F}$

  Output: X^+

  C = X; // initialize C to the set X
  repeat until no change in C {
  if there is an FD $U \to V$ in $\mathcal{F}$ such that $U \subseteq C$,
  then $C = C \cup V$;
  }

Example

- With $\mathcal{F} = \{ A \to B, B \to C \}$
- Compute A^+

- Closure = { A }
- Closure = { A, B } (due to $A \to B$)
- Closure = { A, B, C } (due to $B \to C$)
- Closure = { A, B, C }
  - no change, stop

Attribute Closure

- Prove that the algorithm indeed computes X^+
  - Show that for any attribute $A \in X^+$, $X \rightarrow A$ is derivable from $\mathcal{F}$
  - Show if $X \rightarrow A$ is derivable from $\mathcal{F}$, $A \in X^+$
- To determine if an FD $X \rightarrow Y$ is implied by $\mathcal{F}$, compute X^+ and check if $Y \subseteq X^+$.
- Notice that attribute closure is less expensive to compute
- Algorithm can be easily modified to compute candidate keys

Minimal Cover

- Naturally, given a set $\mathcal{F}$ of FDs, it is more desirable to work with the minimal equivalent set of FDs of $\mathcal{F}$
- A set $\mathcal{F}$ of FDs is equivalent to a set $\mathcal{G}$ of FDs if $\mathcal{F}^+ = \mathcal{G}^+$
- Given a set $\mathcal{F}$ of FDs, what is the minimal cover for F and how do we compute it?

Example

$\mathcal{F}$ = { $A \rightarrow B$, $AB \rightarrow C$ }

$\mathcal{G}$ = { $A \rightarrow B$, $A \rightarrow C$ }

Is $\mathcal{F}^+ = \mathcal{G}^+$? Notice that $A \rightarrow C$ can be derived from $\mathcal{F}$ and $AB \rightarrow C$ can be derived from $\mathcal{G}$.

Minimal Cover

- A set $\mathcal{F}$ of FDs is minimal if
  - For every FD $X \rightarrow Y$ and an attribute $A \in Y$, it is not the case that $\mathcal{F} - \{ X \rightarrow Y \} \cup \{ X \rightarrow (Y - \{A\}) \}$ is equivalent to $\mathcal{F}$
  - For every FD $X \rightarrow Y$ and an attribute $A \in X$, it is not the case that $\mathcal{F} - \{ X \rightarrow Y\} \cup \{ ((X - \{A\}) \rightarrow Y) \}$ is equivalent to $\mathcal{F}$
  - Each left hand side of a FD in $\mathcal{F}$ is unique. Take any two FDs $X \rightarrow Y$ and $X' \rightarrow Y'$, it must be that $X \neq X'$

A is an extraneous attribute

Determining extraneous attributes

- Example
  - The set of FDs $\{ A \rightarrow B, AB \rightarrow C \}$ is not minimal as it is equivalent to $\{ A \rightarrow B, A \rightarrow C \}$
  - $\{ A \rightarrow B, AB \rightarrow C , A \rightarrow C \}$ is not minimal as the LHS of $A \rightarrow B$ and $A \rightarrow C$ are not unique

- Consider an FD $X \rightarrow Y$ in $\mathcal{F}$
  - To check if A is an extraneous attribute on the RHS of $X \rightarrow Y$,
    - Let $\mathcal{F}' = F - \{ X \rightarrow Y \} \cup \{ X \rightarrow (Y - \{A\})\}$
    - Compute X^+ using $\mathcal{F}'$ to check if A can be inferred
    - If A can be inferred from X^+, A is extraneous

Determining extraneous attributes

- Consider an FD $X \rightarrow Y$ in $\mathcal{F}$
  - To check if A is an extraneous attribute on the LHS of $X \rightarrow Y$,
    - Compute $(X\text{-}\{A\})^+$ using $\mathcal{F}$ to check if X-{A} can infer Y
    - If Y can be inferred, A is extraneous

Examples

- Let $\mathcal{F} = \{ ABC \rightarrow E, A \rightarrow F, A \rightarrow BE, B \rightarrow DE \}$
- A is extraneous in $ABC \rightarrow E$ because $BC^+ = \{ B, C, D, E \}$
- $\mathcal{F}_1 = \{ BC \rightarrow E, A \rightarrow F, A \rightarrow BE, B \rightarrow DE \}$
- B is not extraneous in $BC \rightarrow E$ because $C^+ = \{ C \}$
- C is extraneous in $BC \rightarrow E$ because $B^+ = \{ B, D, E \}$
- $\mathcal{F}_2 = \{ B \rightarrow E, A \rightarrow F, A \rightarrow BE, B \rightarrow DE \}$
- B is not extraneous in $A \rightarrow BE$ because $A^+ = \{ A, E, F \}$ w.r.t $\mathcal{F}' = \{ B \rightarrow E, A \rightarrow F, A \rightarrow E, B \rightarrow DE \}$
- E is extraneous in $A \rightarrow BE$ because $A^+ = \{A, B, D, E, F\}$ w.r.t $\mathcal{F}' = \{ B \rightarrow E, A \rightarrow F, A \rightarrow B, B \rightarrow DE \}$
- $\mathcal{F}_3 = \{ B \rightarrow E, A \rightarrow F, A \rightarrow B, B \rightarrow DE \}$
- D is not extraneous in $B \rightarrow DE$. Why?
- E is extraneous in $B \rightarrow E$. Why?

Computing the minimal cover

Min_Cover = $\mathcal{F}$

Repeat {

Apply union rule to merge FDs with the same LHS in Min_Cover;

*Find an FD with an extraneous attribute in LHS or RHS;

Delete extraneous attribute from the FD;

} until no change in Min_Cover;

* Enumerate each FD in $\mathcal{F}$ and check for each attribute in the FD, whether they are extraneous

Example

- Consider the following set of FDs:
  { A → BC, A → B, B → AC, C → AB }
- Apply union rule:
  { A → BC, B → AC, C → AB }
- Consider A → BC.
  - Is A extraneous?
  - Is B extraneous?
    - A^+ (using F') is {C, A, B}. YES since B can be inferred
- The set of FDs: { A → C, B → AC, C → AB }
- …
- Min_cover = { A → C, B → C, C → AB }
- Another minimal cover (by considering different extraneous attributes):
  { A → B, B → C, C → A }

Minimal cover is not unique

Normal Forms

- Given a relation schema, we need to understand whether it is a good design.
- Intuitively, a good design is one that does not store data redundantly.
- Recall previous example:

Employee(eid, name, addr, rank, salary-scale)

Employee(eid, name, addr, rank)
Salary-Scale(rank, salary-scale)

- Normal forms allow us to store data non-redundantly, given certain constraints we know about the data

First Normal Norm (1NF)

- A relation schema is in 1NF if the type of every attribute is atomic
- Example:
  R(ssn: char(9), name: string, age: int)
- All our examples so far have been in 1NF.
- Non-first normal form relation:
  R(ssn: char(9), name: Record[firstname: string, lastname: string], age: int, children: Set(string))
- Very basic requirement on relations. Not based on FDs.

Boyce-Codd Normal Form (BCNF)

- Let R be a relation schema, $\mathcal{F}$ be a set of FDs given to hold over R, A is an attribute in R, and X is a subset of attributes in R
- R is in BCNF if
  - for every FD X $\rightarrow$ A in $\mathcal{F}$, either
    - X $\rightarrow$ A is a trivial dependency (i.e., A $\subseteq$ X) or,
    - X is a superkey
- BNCF is desirable from redundancy point of view

BCNF example

A	B	C
a1	b1	c1
a1	b2	c1

- Given that A $\rightarrow$ C, we can infer that C value of second tuple is also c1
- But a1 and c1 are obviously redundantly stored
- The relation is not in BCNF because
  - Given that A $\rightarrow$ C is not a trivial dependency, A must be a superkey.
  - If A is a key, the B value of second tuple should be b1. This means we have two identical copies of the tuple (a1, b1, c1) which is disallowed with set semantics

Third Normal Form (3NF)

- Let R be a relation schema, $\mathcal{F}$ be a set of FDs given to hold over R, A is an attribute in R, and X is a subset of attributes in R
- R is in 3NF if
  - For every FD $X \rightarrow A$ in $\mathcal{F}$, one of the following is true
    - $X \rightarrow A$ is a trivial dependency (i.e., $A \subseteq X$)
    - X is a superkey
    - A is part of some key for R
- Note that A has to be the part of some minimal key for R

3NF example

- 3NF is not as strict as BCNF. Some redundancy may still be there.
- Consider R(A, B, C, D) and $A \rightarrow D$.
- This relation schema is not in 3NF since
  - $A \rightarrow D$ is not a trivial dependency, A is not a superkey, and D is not part of the key
- A,D values may occur redundantly

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d1
a1	b2	c3	d1
a2	b2	c3	d2

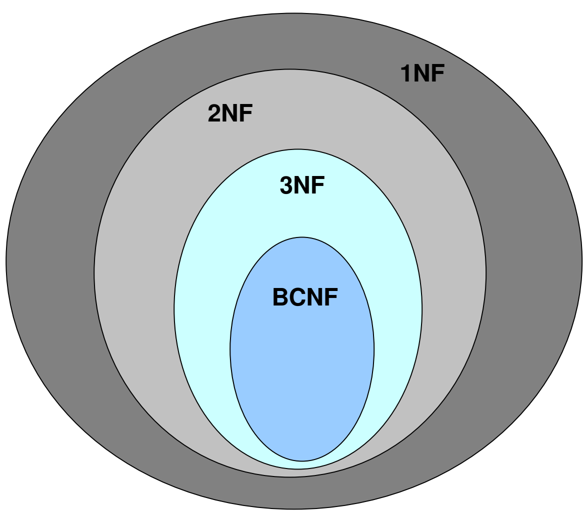
3NF example

- Now consider R(A, B, C, D), A → D, and D → A.
- BCD is also a key for R
- Therefore R is in 3NF
- However, A and D values may still occur redundantly

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d1
a1	b2	c3	d1
a2	b2	c3	d2

- An example where the relation is in 3NF but not in BCNF

The big picture



Summary

- Anomalies caused by redundancy
- Functional dependencies
  - Closure of FDs
  - Armstrong's axioms
  - Attribute closure
  - Minimal cover
- Normal forms
  - 1NF, BCNF, 3NF