Schema Refinement and Normal Forms

Chapter 19

The Evils of Redundancy

- *Redundancy* is at the root of several problems associated with relational schemas

SSN	Name	Lot	Rate	W	H
123223666	Attishoo	48	8	10	40
23315368	Smiley	22	8	10	30
131243650	Smethurst	35	5	7	30
434263751	Guldu	35	5	7	32
612674134	Madayan	35	8	10	40

The Evils of Redundancy

SSN	Name	Lot	Rate	W	H
123223666	Attishoo	48	8	10	40
23315368	Smiley	22	8	10	30
131243650	Smethurst	35	5	7	30
434263751	Guldu	35	5	7	32
612674134	Madayan	35	8	10	40

- *Redundancy* is at the root of several problems associated with relational schemas
- *Functional dependency:* R → W. R *determines* W

The Evils of Redundancy

- *Redundancy* is at the root of several problems associated with relational schemas

SSN	Name	Lot	Rate	W	H
123223666	Attishoo	48	8	10	40
23315368	Smiley	22	8	10	30
131243650	Smethurst	35	5	7	30
434263751	Guldu	35	5	7	32
612674134	Madayan	35	8	10	40

- *Functional dependency:* R → W. R *determines* W
- Problems due to R →W :
  - *Update anomaly*: Can we change W in just the first tuple
  - *Insertion anomaly*: What if we want to insert an employee and don't know the hourly wage for his rating?
  - *Deletion anomaly*: If we delete all employees with rating 5, we lose the information about the wage for rating 5!

Notations

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

Example: Decomposition

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

→ Wages

R	W
8	10
5	7

→ Hourly_Emps2

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

Will 2 smaller tables be better?

Functional Dependencies (FDs)

- A functional dependency $X \rightarrow Y$ holds over relation R if, for every allowable instance *r* of R:
  - $t1 \in r,\ t2 \in r,\ \pi_X(t1) = \pi_X(t2)$ implies $\pi_Y(t1) = \pi_Y(t2)$
  - 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.)
- An FD is a statement about *all* allowable relations.
  - Must be identified based on semantics of application.
  - Given some allowable instance *r1* of R, we *can* check if it violates some FD *f*, but we *cannot* tell if *f* holds over R!
- K is a candidate key for R means that $K \rightarrow R$

Reasoning About FDs

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

Reasoning About FDs

- **Armstrong's Axioms** (X, Y, Z are sets of attributes):
  - *Reflexivity*: If $X \subseteq Y$, then $Y \rightarrow X$
  - *Augmentation*: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
  - *Transitivity*: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

- Couple of additional rules (that follow from AA):
  - *Union*: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
  - *Decomposition*: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

Reasoning About FDs (Contd.)

- Example: Contracts(*cid,sid,jid,did,pid,qty,value*), and:
  - C is the key: C $\rightarrow$ CSJDPQV
  - Project purchases each part using single contract: JP $\rightarrow$ C
  - Dept purchases at most one part from a supplier: SD $\rightarrow$ P
- JP $\rightarrow$ C, C $\rightarrow$ CSJDPQV imply JP $\rightarrow$ CSJDPQV
- SD $\rightarrow$ P implies SDJ $\rightarrow$ JP
- SDJ $\rightarrow$ JP, JP $\rightarrow$ CSJDPQV imply SDJ $\rightarrow$ CSJDPQV

Reasoning About FDs (Contd.)

- Computing the closure of a set of FDs can be expensive. (Size of closure is exponential in # attrs!)
- Typically, we just want to check if a given FD $X \rightarrow Y$ is in the closure of a set of FDs *F*. An efficient check:
  - Compute *attribute closure* of X (denoted X^+) wrt *F:*
    - Set of all attributes A such that X $\rightarrow$ A is in F^+
    - There is a linear time algorithm to compute this.
  - Check if Y is in X^+
- Does F = {A $\rightarrow$ B, B $\rightarrow$ C, C D $\rightarrow$ E } imply A $\rightarrow$ E?
  - i.e, is A $\rightarrow$ E in the closure F^+? Equivalently, is E in A^+ ?

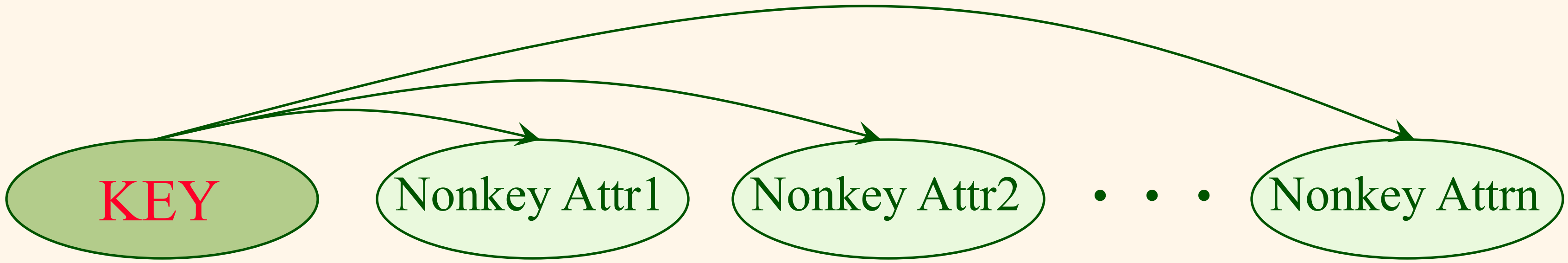
Normal Forms

- Returning to the issue of schema refinement, the first question to ask is whether any refinement is needed!
- If a relation is in a certain *normal form*, then we know that certain kinds of problems cannot arise:
  - *First normal form* (1NF)
  - *Second normal form* (2NF)
  - *Third normal form* (3NF)
  - *Boyce-Codd normal form* (BCNF)
  - *Fourth normal form* (4NF)
  - *Fifth normal form* (5NF)

BCNF $\subset$ 3NF $\subset$ 2NF $\subset$ 1NF

Boyce-Codd Normal Form (BCNF)

- Reln R with FDs *F* is in BCNF if, for all possible $X \rightarrow A$
  - $A \in X$ (called a *trivial* FD), or
  - X contains a key for R.
- In other words, R is in BCNF if the only non-trivial FDs that hold over R are key constraints.
- BCNF ensures that no redundancy can be detected using FD information alone.



Third Normal Form (3NF)

- Reln R with FDs F is in 3NF if, for all X$\rightarrow$ A in F^+
  - A $\in$ X (called a *trivial* FD), or
  - X contains a key for R, or
  - A is part of some key for R.
- *Minimality* of a key is crucial in third condition above!
- If R is in BCNF, obviously in 3NF.
- If R is in 3NF, some redundancy is possible. It is a compromise, used when BCNF not achievable (e.g., no ``good'' decomp, or performance considerations).

Normal Forms

Normal Form	Condition
1NF	Atomic values
2NF	All non-key attributes functionally depend on keys
3NF	LHS of FDs contain key, or RHS is part of key
BCNF	LHS of FDs contain key

Normal Form Example

EID	Name	Job_Code	Job	Zip	State
E001	Alice	J01	Chef	53706	Wisconsin
E001	Alice	J02	Waiter	53706	Wisconsin
E002	Bob	J02	Waiter	55455	Minnesota
E002	Bob	J03	Bartender	55455	Minnesota
E003	Alice	J01	Chef	55411	Minnesota

Two FDs are known to be true:

1) EID –> Name, Zip, State
2) Zip –> State

Normal Form Example

Zip –> State

EID	Job_Code
E001	J01
E001	J02
E002	J02
E002	J03
E003	J01

EID	Name	Zip	State
E001	Alice	53706	Wisconsin
E002	Bob	55455	Minnesota
E003	Alice	55411	Minnesota

Job_Code	Job
J01	Chef
J02	Waiter
J03	Bartender

Normal Form	Condition
1NF	Atomic values
2NF	All non-key attributes functionally depend on keys
3NF	LHS of FDs contain key, or RHS is part of key
BCNF	LHS of FDs contain key

Normal Form Example

EID	Job_Code
E001	J01
E001	J02
E002	J02
E002	J03
E003	J01

EID	Name	Zip
E001	Alice	53706
E002	Bob	55455
E003	Alice	55411

Zip	State
53706	Wisconsin
55455	Minnesota
55411	Minnesota

Job_Code	Job
J01	Chef
J02	Waiter
J03	Bartender

Normal Form	Condition
1NF	Atomic values
2NF	All non-key attributes functionally depend on keys
3NF	LHS of FDs contain key, or RHS is part of key
BCNF	LHS of FDs contain key

Normal Form Example

Manager	Project	Branch
Alan	APEX	Minneapolis
Bob	TECO	St. Paul
Bob	APEX	St. Paul
Charlie	QUBE	St. Paul
Charlie	TELESCOPE	St. Paul

PK: (Project, Branch)

Two FDs are known to be true:

1) Manager –> Branch
2) Project, Branch –> Manager

Normal Form	Condition
1NF	Atomic values
2NF	All non-key attributes functionally depend on keys
3NF	LHS of FDs contain key, or RHS is part of key
BCNF	LHS of FDs contain key