Attribute Closure

- Calculating *attribute closure* is a more efficient way of checking entailment
- The *attribute closure* of a set of attributes, X, with respect to a set of functional dependencies, F, (denoted X^+_F) is the set of all attributes, A, such that $X \rightarrow A$
  - X^+_{F1} is not necessarily the same as X^+_{F2} if $F1 \neq F2$
- *Attribute closure and entailment*:
  - Given a set of FDs, F, then $X \rightarrow Y$ if and only if $X^+_F \supseteq Y$

19

Example - Computing Attribute Closure

$F: AB \rightarrow C$
$A \rightarrow D$
$D \rightarrow E$
$AC \rightarrow B$

X	X_F^+
A	$\{A, D, E\}$
AB	$\{A, B, C, D, E\}$ (Hence AB is a key)
B	$\{B\}$
D	$\{D, E\}$

Is $AB \rightarrow E$ entailed by F? *Yes*
Is $D \rightarrow C$ entailed by F? *No*

Result: X_F^+ allows us to determine FDs of the form $X \rightarrow Y$ entailed by F

20

Computation of Attribute Closure X^+_F

closure := X; --since $X \subseteq X^+_F$
repeat
  old := *closure*;
  if there is an FD $Z \rightarrow V$ in F such that $Z \subseteq$ *closure* **and** $V \not\subseteq$ *closure*
    then *closure* := *closure* $\cup$ V
until *old* = *closure*

– If $T \subseteq$ *closure* then $X \rightarrow T$ is entailed by F

21

Example: Computation of Attribute Closure

Problem: Compute the attribute closure of AB with respect to the set of FDs :

$AB \rightarrow C$ (a)
$A \rightarrow D$ (b)
$D \rightarrow E$ (c)
$AC \rightarrow B$ (d)

Solution:

Initially *closure* = $\{AB\}$
Using (a) *closure* = $\{ABC\}$
Using (b) *closure* = $\{ABCD\}$
Using (c) *closure* = $\{ABCDE\}$

22

Normal Forms

- Each normal form is a set of conditions on a schema that guarantees certain properties (relating to redundancy and update anomalies)
- First normal form (1NF) is the same as the definition of relational model (relations = sets of tuples; each tuple = sequence of atomic values)
- Second normal form (2NF) – a scientific accident; has no practical or theoretical value – won't discuss
- The two commonly used normal forms are *third normal form* (3NF) and *Boyce-Codd normal form* (BCNF)

23

BCNF

- **Definition**: A relation schema **R** is in BCNF if for every FD $X \rightarrow Y$ associated with **R** either
  - $Y \subseteq X$ (i.e., the FD is trivial) or
  - X is a superkey of **R**
- **Example**: Person1(*SSN*, *Name*, *Address*)
  - The only FD is *SSN* $\rightarrow$ *Name*, *Address*
  - Since *SSN* is a key, Person1 is in BCNF

24

BCNF - Examples

- Person (*SSN*, *Name*, *Address*, *Hobby*)
  - The FD *SSN* → *Name, Address* does not satisfy requirements of BCNF (since the key is (*SSN, Hobby*))
- HasAccount (*AccountNumber*, *ClientId, OfficeId*)
  - The FD *AcctNum*→ *OfficeId* does not satisfy BCNF requirements (since keys are (*ClientId*, *OfficeId*) and (*AcctNum, ClientId*))

25

Redundancy

- Suppose **R** has a FD $A \rightarrow B$. If an instance has 2 rows with same value in *A*, they *must* also have same value in *B* (=> redundancy, if the A-*value* repeats twice)

redundancy

SSN → *Name, Address*

SSN	*Name*	*Address*	*Hobby*
1111	Joe	123 Main	stamps
1111	Joe	123 Main	coins

- If *A* is a superkey, there cannot be two rows with same value of *A*
  - Hence, BCNF eliminates redundancy

26

Third Normal Form

- A relational schema **R** is in 3NF if for every FD $X \rightarrow Y$ associated with **R** either:
  - $Y \subseteq X$ (i.e., the FD is trivial); or
  - *X* is a superkey of **R;** or
  - Every $A \in Y$ is part of some key of **R**
- 3NF is weaker than BCNF (every schema that is in BCNF is also in 3NF)

BCNF conditions

27

3NF Example

- HasAccount (*AcctNum, ClientId, OfficeId*)
  - *ClientId, OfficeId* → *AcctNum*
    - OK since LHS contains a key
  - *AcctNum* → *OfficeId*
    - OK since RHS is part of a key
- HasAccount is in 3NF but it might still contain redundant information due to *AcctNum* → *OfficeId* (which is not allowed by BCNF)

28

3NF Example

- HasAccount might store redundant data:

ClientId	*OfficeId*	*AcctNum*
1111	Stony Brook	28315
2222	Stony Brook	28315
3333	Stony Brook	28315

3NF: *OfficeId* part of key
FD: *AcctNum* → *OfficeId*

redundancy

- Decompose to eliminate redundancy:

ClientId	*AcctNum*
1111	28315
2222	28315
3333	28315

BCNF (only trivial FDs)

OfficeId	*AcctNum*
Stony Brook	28315

BCNF: *AcctNum* is key
FD: *AcctNum* → *OfficeId*

29

3NF (Non) Example

- Person (*SSN*, *Name*, *Address*, *Hobby*)
  - (*SSN*, *Hobby*) is the only key.
  - *SSN*→ *Name* violates 3NF conditions since *Name* is not part of a key and *SSN* is not a superkey

30

Decompositions

- **Goal**: Eliminate redundancy by decomposing a relation into several relations in a higher normal form
- Decomposition must be *lossless*: it must be possible to reconstruct the original relation from the relations in the decomposition

31

Decomposition

- Schema **R** = (*R, F*)
  - *R* is set a of attributes
  - *F* is a set of functional dependencies over *R*
    - Each key is described by a FD
- The *decomposition of schema* **R** is a collection of schemas $\mathbf{R}_i = (R_i, F_i)$ where
  - $R = \cup_i R_i$ for all *i* (*no new attributes*)
  - F_i is a set of functional dependences involving only attributes of R_i
  - *F* entails F_i for all *i* (*no new FDs*)
- The *decomposition of an instance*, **r**, of **R** is a set of relations $\mathbf{r}_i = \pi_{R_i}(\mathbf{r})$ for all *i*

32

Example Decomposition

Schema *(R, F)* where

R = {SSN, Name, Address, Hobby}

F = {SSN→ Name, Address}

can be decomposed into

R_1 *= {SSN, Name, Address}*

F_1 *= {SSN →Name, Address}*

and

R_2 *= {SSN, Hobby}*

F_2 *= { }*

33