CSE 414: Section 8
BCNF and Views

November 29th, 2018

Outline

BCNF decomposition

1) Check whether chosen FD violates BCNF
2) Use any FD that violates BCNF to decompose.

View construction and query processing

1) From vertically partitioned tables
2) From horizontally partitioned tables

Keys

We call an attribute that determines all other attributes in a schema to be a **superkey**.

If it is the smallest set of attributes (in terms of cardinality) that does this we call that set a **minimal key** or just **key**

Closure Algorithm

Repeat until X doesn't change **do**:
if $B_1, \ldots, B_n \rightarrow C$ is a FD **and**
$B_1, \ldots, B_n$ are all in X
then add C to X.

Goal:
We want everything that an attribute/set of attributes determine

Observation:

- If we have A -> B and B -> C, then A -> C
- So really, A -> B and C
- Formal notation is $\{A\}^+ = \{A, B, C\}$
- **Since the closure of A is all attributes, A is a key**

Conceptual Design

SSN → Name, City

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield

Conceptual Design

Anomalies:

- Redundancy = repeat data
- Update anomalies = what if Fred moves to “Bellevue”?
- Deletion anomalies = what if Joe deletes his phone number?

Conceptual Design

- The BCNF (Boyce-Codd Normal Form) ---- A relation R is in BCNF if every set of attributes is either a superkey or its closure is the same set

BCNF Decomposition Algorithm

Normalize(R)

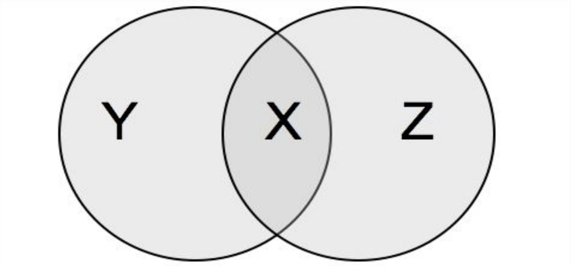
find X s.t.: $X \neq X^+$ and $X^+ \neq$ [all attributes]

<u>if</u> (not found) **<u>then</u>** R is in BCNF

<u>let</u> $Y = X^+ - X$; $Z =$ [all attributes] $- X^+$

decompose R into $R1(X \cup Y)$ and $R2(X \cup Z)$

Normalize(R1); Normalize(R2);



X +

Example

The relation is R (A, B, C, D, E)

FDs : $A \rightarrow E$, $BC \rightarrow A$, and $DE \rightarrow B$

Question : Decompose R into BCNF.

Solution

Notice that $\{A\}^+ = \{A,E\}$, which violates the BCNF condition.

We split R to R1(A,E) and R2(A,B,C,D).

R1 satisfies BCNF now, but R2 does not because: $\{B,C\}^+ = \{B,C,A\}$.

Notice that there is no E in R2 table so we don't need to consider the FD DE $\rightarrow$ B!

Split R2 to: R21(B,C,A) and R22(B,C,D)

Lossless Decomposition

Consider the relation R(A,B,C,D,E)

FDs: {AB → C, BC → D, AD → E}

$S1 = \Pi_{ABC}(R)$, $S2 = \Pi_{BCD}(R)$, $S3 = \Pi_{ADE}(R)$

We need to show that $R = S1 \bowtie S2 \bowtie S3$

S1(ABC)
S2(BCD)
S3(ADE)

Vertical Partitioning

Resumes

SSN	Name	Address	Resume	Picture
234234	Mary	Houston	Doc1...	JPG1...
345345	Sue	Seattle	Doc2...	JPG2...
345343	Joan	Seattle	Doc3...	JPG3...
432432	Ann	Portland	Doc4...	JPG4...

T1

SSN	Name	Address
234234	Mary	Houston
345345	Sue	Seattle
...		

T2

SSN	Resume
234234	Doc1...
345345	Doc2...

T3

SSN	Picture
234234	JPG1...
345345	JPG2...

T2.SSN is a key *and* a foreign key to **T1**.SSN. Same for **T3**.SSN

Vertical Partitioning

```
CREATE VIEW  Resumes  AS
  SELECT  T1.ssn, T1.name, T1.address,
          T2.resume, T3.picture
  FROM    T1,T2,T3
  WHERE   T1.ssn=T2.ssn AND T1.ssn=T3.ssn
```

```
SELECT address
FROM   Resumes
WHERE  name = ‘Sue’
```

Vertical Partitioning

Original query:

```
SELECT T1.address
FROM T1, T2, T3
WHERE T1.name = ‘Sue’
    AND T1.SSN=T2.SSN
    AND T1.SSN = T3.SSN
```

Final query:

```
SELECT T1.address
FROM T1
WHERE T1.name = ‘Sue’
```

Vertical Partitioning Applications

- Advantages
  - Speeds up queries that touch only a small fraction of columns
  - Single column can be compressed effectively, reducing disk I/O

- Disadvantages
  - Updates are very expensive!
  - Need many joins to access many columns
  - Repeated key columns add overhead

Horizontal Partitioning

Customers

SSN	Name	City
234234	Mary	Houston
345345	Sue	Seattle
345343	Joan	Seattle
234234	Ann	Portland
--	Frank	Calgary
--	Jean	Montreal

CustomersInHouston

SSN	Name	City
234234	Mary	Houston

CustomersInSeattle

SSN	Name	City
345345	Sue	Seattle
345343	Joan	Seattle

. . . . . .

Horizontal Partitioning

```
CREATE VIEW  Customers  AS
    (SELECT SSN, name, ‘Houston’ as city
     FROM CustomersInHouston)
       UNION ALL
    (SELECT SSN, name, ‘Seattle’ as city
     FROM CustomersInSeattle)
       UNION ALL

     . . .
```

Horizontal Partitioning

```
SELECT  name
FROM    Customers
WHERE   city = 'Seattle'
```

```
SELECT name
FROM   CustomersInSeattle
```

Horizontal Partitioning Applications

- Performance optimization
  - Especially for data warehousing
  - E.g., one partition per month
  - E.g., archived applications and active applications
- Distributed and parallel databases