More Functional Dependencies and Normalization

CS157A

Chris Pollett

Nov. 16, 2005.

Last Day

- We talked about some informal design criteria we'd like our relation schemas to satisfy:
  - easy to explain meaning
  - no anomalies
  - minimize nulls
  - no spurious tuples
- Then talked about function dependencies (FDs) and Armstrong's axioms.

Today’s Outline

- Closure of a set of attributes
- FD Covers
- Minimal Covers
- Normal Forms

Closure of a Set of Attributes

- In order to define our normal forms, we need to be able to define the closure of a set of attributes under a set of FDs.
- Definition: Let X^+ denote all those attributes which functionally depend on X.
- Here's an algorithm to compute X^+ from X and F:

```
initialize X+ := X;
repeat
        old X+ := X+;
        for each FD, Y--> Z, in F do
           if X+ ⊇ Y then X+ := X+ ∪ Z;
until X+ = old X+ ;
```

Covers

- Definition: A set of FDs F is said to **cover** another set of FDs E if every FD in E is in F^+. Two FDs F and E are **equivalent** if they cover each other.
- One can check if F covers E by looking at each FD X-->Y in E and checking whether X^+ calculated according to F contains Y.

Minimal Covers

- Intuitively, a minimal cover of a set of functional dependencies F is a smallest subset of F^+ which covers F.
- We also would like to have a nice form for our minimal covers.
- So we say a set of FDs is **minimal** if:
  - Every FD in F has a single attribute on its right hand side.
  - We cannot replace an X-->A in F by Y-->A where Y is strictly contained in X and get an equivalent set of dependencies
  - We cannot remove any dependency from F and get an equivalent set of FDs.
- A **minimal cover for FDs E** is a set F of FDs which covers E and is minimal.

Minimal Cover Algorithm

INPUT: A set E of FDs

OUTPUT: A set F of FDs so that F is a minimal cover of E.

Algorithm:

Set F := E

Replace each FD X-->{A_1,…A_n} in F by the n FDs X-->A_1,…, X-->A_n.

For each FD X-->A in F, for each attribute B in X

if {{F-{X-->A} ∪{(X-{B})-->A}} is equivalent to F then replace X-->A by (X-{B})-->A in F.

For each remaining FD X-->A in F if {F-{X-->A}} is equivalent to F remove X-->A from F.

Normalization of Relations

- We now consider some normal forms for our tables which will allow us to judge if we’ve split our attributes among tables reasonably.
- We give algorithms both for checking normal forms as well as for putting tables into normal forms.
- The latter process is called **normalization**.
- Sometimes for efficiency of queries, etc we might later choose weaker normal forms over stronger ones and do the reverse process known as **denormalization**.
- We need one last definition first. Call an attribute of a relation schema R **prime** if it is a member of some candidate key of R. If an attribute is not prime call it **nonprime**.

First Normal Form (1NF)

- This normal form has actually been incorporated into the definition of the relational model.
- It says that the domain of an attribute must include only atomic (simple, indivisible) values.
- Hence, 1NF disallows multivalued attributes.

Second Normal Form (2NF)

- A functional dependency X-->Y is a **full functional dependency** if removal of any attribute A from X means that the dependency doesn’t hold any more.
- A relation schema R is in 2NF if every nonprime attribute A in R is fully functionally dependent on the primary key of R.

2NF Example

- Consider:

  EMP_PROJ(SSN, PNUMBER, HOURS, ENAME, PNAME, PLOCATION)

- Suppose our FDs are:

  SSN, PNUMBER--> HOURS

  SSN --> ENAME

  PNUMBER --> PNAME, PLOCATION.

- Then this is not in 2NF as ENAME for instance only depends on SSN and not both SSN and PNUMBER.
- Not being in 2NF suggests redundancy in the data.
- To normalize this table we could split it into:

  EMP(SSN, ENAME), WORKS(SSN, PNUMBER, HOURS), PROJ(PNUMBER, PNAME, PLOCATION)

Third Normal Form

- An FD X-->Y is called a **transitive dependency** if there is a Z that is neither a candidate key nor a subset of a key in R such that X-->Z and Z-->Y both hold on R.
- A relation is in 3NF if it is in 2NF and no nonprime attribute of R is transitively dependent on the primary key.

3NF Example

- Consider:

  EMP_DEPT(ENAME, SSN, BDATE, ADDRESS, DNUMBER, DNAME, DMGRSSN)
- Suppose our FDs are:

  SSN --> ENAME, BDATE, ADDRESS, DNUMBER

  DNUMBER --> DNAME, DMGRSSN.
- In this case DNAME is a nonprime attribute which depends transitively on SSN through DNUMBER.
- Anomalies can occur if not in 3NF.
- To put this in 3NF we could split it into the tables

  EMP(ENAME, SSN, BDATE, ADDRESS)

  DEPT(DNUMBER, DNAME, DMGRSSN)