Database Management Systems

Database Design (I)

Example of Bad Design

Sno	Name	Cno	Title	Term	Grade
123	J King	CSC 120	Introduction of CS	Fall 2005	B
123	J King	CSC 240	Data Structures	Fall 2006	C
123	N King	CSC 340	Operating Systems	Fall 2007	A
321	M Ng	CSC 120	Introduction of CS	Fall 2005	A
222	P Smith	CSC 120	Introduction of CS	Fall 2005	B

What’s Wrong?

- Redundant data
- Update problem (change the title of a course)
- Insert problem (add a new course)
- Delete problem (remove a student)

How to Judge a Schema?

- no redundant data
- easy to query
- easy to update
- only allow well-behaved instances
- Any concrete theory basis?

Functional Dependency Definition

Let R be a relation schema, and $X,Y \subset R$, where X, Y and R are sets of attributes. The functional dependency

$X \rightarrow Y$ (X functionally determines Y (in R))

holds on R if whenever an instance of R contains two tuples t and u such that $t[X] = u[X]$, then it must also be true that $t[Y] = u[Y]$.

Example of FDs

- {Sno} → {Name}, or Sno → Name
- Cno → Title
- {Sno, Cno, Term} → {Grade},
  or Sno, Cno, Term → Grade
- Where do they come from?
  - From domain knowledge
  - From inference

Reasoning About FDs

- Armstrong's Axioms (X, Y, Z are all sets of attributes)
  - (reflexivity) $Y \subseteq X \Rightarrow X \rightarrow Y$ (trivial functional dependency)
  - (augmentation) $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$
  - (transitivity) $X \rightarrow Y, Y \rightarrow Z \Rightarrow X \rightarrow Z$
- Additional rules can be derived
  - (union) $X \rightarrow Y, X \rightarrow Z \Rightarrow X \rightarrow YZ$
  - (decomposition) $X \rightarrow YZ \Rightarrow X \rightarrow Y$ and $X \rightarrow Z$

Implication for FDs

- closure of F, denoted as F^+
  $R \models F \Leftrightarrow R \models F^+$
- Relation schema R satisfies all the functional dependencies in the set F if and only if relation schema R satisfies all the functional dependencies in the set of F closure.
- In other words, all the functional dependencies in the set F holds on relation R if and only if all the functional dependencies in the set F closure holds on relation R.
- the axiom is
  - sound
  - complete

Keys: Formal Definition

- superkey: $K \subseteq R$ is a superkey for R if functional dependency $K \rightarrow R$ holds on R.
- candidate key: $K \subseteq R$ is a candidate key for R if K is a superkey and no subset of K is a superkey.
- primary key: a candidate key chosen by the database designer(s)

Key Example

- R = {SIN, sno, name, cno, title, term, grade} = {ISNCTEG}
- F =
  {SIN → Sno, Sno → SIN, Sno → Name, Cno → Title,
  Sno, Cno, Term → Grade}
- Superkey of R?
  R, {SNCTE}, {SCE}, etc
- Candidate key of R?
  {SCE}, {ICE}
- Primary key of R?
  Either one of the candidate keys, but {SCE} would be better

Efficient Reasoning

```
// return the set of attributes, each attribute in the set is determined by X
Set ComputeXClosure(X, F) // X: a set of attributes, F: a set of FD
{
    XC = X; // XC: a set of attributes
    while (true) {
        if there exists (Y -> Z) in F such that
            1) Y is subset of XC, and
            2) Z is not a subset of XC
        then XC = XC union Z; // X -> XC, XC -> Y, Y -> Z => X -> Z
        else break;
    }
    return XC;
}
```

How to Use ComputeXClosure?

- Given R as a set of attributes, and F as a set of FDs, X, Y, K are all subset of R
- Is $X \rightarrow Y \in F+$? $\Leftrightarrow$ Is $Y \subseteq$ ComputerXClosure(X,F) ?
- Is K a superkey of R? $\Leftrightarrow$ Is $R \subseteq$ ComputerXClosure(K, F) ?
- Is K a candidate key of R?
  - Is K a superkey of R?
  - For any attribute A in K, is K-{A} still a superkey of R?
- $F \equiv G \Leftrightarrow F+ = G+$

  (Functional dependency set F is equivalent to set G if and only if F closure is identical as G closure.)