FUNCTIONAL DEPENDENCIES

CHAPTER 15.1-15.2, 15.5 (6/E)
CHAPTER 10.1-10.2, 10.5 (5/E)
LECTURE OUTLINE

- Design guidelines for relation schemas
- Functional dependencies
  - Definition and interpretation
  - Formal definition of keys
- Boyce-Codd Normal Form (BCNF)
  - Application of dependency theory to checking DB design
Clarity of attributes provides semantics for relation schema.

- Naming of attributes
- *Fit* of attributes with each other

**Guideline 1**
- Design each relation schema so that it is easy to explain its meaning.
  - Natural result of good ER design
  - Do not arbitrarily combine attributes from multiple entity types and relationship types into a single relation.

How can we measure how well attributes fit together?

- Amount of redundant information in tuples
- Amount of NULL values in tuples
- Possibility of generating spurious tuples
MIS-PACKAGED ATTRIBUTES

Every tuple includes employee data and department data

Redundancy
  • Dept name and manager id repeated for every employee in dept

Potential for too many NULL values
  • Departments with no employees need to pad tuple with NULLS
  • Employees not in any department need to pad tuples with NULLS

Update anomalies
  • Deleting the last employee in a dept should not delete dept
  • Changing the dept name/mgr requires many tuples to be updated
  • Inserting employees requires checking for consistency of its dept name and manager

Guideline 2
  • Design relational DB schema so that every fact can be stored in one and only one tuple.
Assume that no two actors have the same name.

Each actor has a unique date and city of birth.

Therefore, given an actor’s name, there is only one possible value for birth and for city.

- name → birth
- name → city

However, given a birth year, we do not have a unique corresponding name or city.

- birth ↭ name
- birth ↭ city

Cannot tell from example whether or not city determines name or birth
FUNCTIONAL DEPENDENCY

- Constraint between two sets of attributes from the database

  Given relation scheme \( R(A_1,A_2,\ldots,A_n) \) and sets of attributes \( X \subseteq \{A_1,A_2,\ldots,A_n\} \), \( Y \subseteq \{A_1,A_2,\ldots,A_n\} \), \( X \rightarrow Y \) specifies the following constraint: for any tuples \( t_1 \) and \( t_2 \) in any valid relation state \( r \) of \( R \), if \( t_1[X] = t_2[X] \) then \( t_1[Y] = t_2[Y] \).

- Property of semantics or meaning of the attributes

- Recognized and recorded as part of database design

- Given a relation state
  - Cannot determine which functional dependencies hold
  - Can state that functional dependency does not hold if there are tuples that show violation of the dependency

- Write \( \{B_1,B_2,\ldots,B_i\} \rightarrow \{C_1,C_2,\ldots,C_j\} \) but can omit set braces if \( i=1 \) or \( j=1 \), respectively.

  - \( \{\text{name}\} \rightarrow \{\text{birth,city}\} \) or \( \text{name} \rightarrow \{\text{birth,city}\} \)
Some dependencies must always hold

- \{\text{birth, date}\} \rightarrow \{\text{birth, date}\}
- \{\text{birth, date}\} \rightarrow \text{date}
- \{\text{birth, date}\} \rightarrow \text{birth}

For any relation schema \( R \) and subsets of attributes \( X \) and \( Y \) in \( R \), if \( Y \subseteq X \), then \( X \rightarrow Y \).
ANOTHER LOOK AT KEYS

- Assume that EMPLOYEE(EmpNo, FirstName, LastName, Department, Email, Phone) has keys:
  1. EmpNo
  2. Email
  3. (FirstName, LastName, Department)

- Some functional dependencies:
  - EmpNo → {EmpNo, FirstName, LastName, Department, Email, Phone}
  - Email → {EmpNo, FirstName, LastName, Department, Email, Phone}
  - {FirstName, LastName, Department} → {EmpNo, FirstName, LastName, Department, Email, Phone}
  - {EmpNo, Email, Phone} → {EmpNo, FirstName, LastName, Department, Email, Phone}

- Given relation scheme R(A₁,A₂,…,Aₙ) and set of attributes X in R. X is a superkey for R if X → {A₁,A₂,…,Aₙ}.
  - Often written as X → R

- To determine that X is a key, need to also show that no proper subset of X determines R
  - ∅ Y such that Y ⊆ X and Y → R
BOYCE-CODD NORMAL FORM

- A relation schema R is in Boyce-Codd Normal Form (BCNF) if whenever a nontrivial functional dependency X → A holds in R, then X is a superkey of R.
  - If X → A and A ≠ X, then X → R
- Relation schemas in BCNF avoid the problems of redundancy
  - We won’t worry about other normal forms in this class.
  - Examples

<table>
<thead>
<tr>
<th>EMP_DEPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ename</td>
</tr>
<tr>
<td>-----------</td>
</tr>
</tbody>
</table>

- Dnumber → {Dname, Dmgr_ssn} but Dnumber ↗ Ename

<table>
<thead>
<tr>
<th>EMP_PROJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ssn</td>
</tr>
<tr>
<td>-------</td>
</tr>
</tbody>
</table>

- Pnumber → {Pname, Plocation} but Dnumber ↗ SSn
- SSn →  Ename but SSn ↗ Pnumber
LECTURE SUMMARY

- Informal guidelines for good design
- Functional dependency
  - Basic tool for analyzing relational schemas
  - Check for Boyce-Codd Normal Form (BCNF) to validate designs