3.1. Functional Dependency

In this lecture we look at...
[Section notes PDF 64Kb]

3.1.01. Introduction

- What is relational design?
  - Notion of attribute distribution
  - Conceptual-level optimisation
- How do we asses the quality of a design?

3.1.02. Value in design

- Allocated arbitrarily by DBD under ER/EER
- Goodness at
  - Internal/storage level (base relations only)
    - Reducing nulls - obvious storage benefits /frequent
    - Reducing redundancy - for efficient storage/anomalies
  - Conceptual level
    - Semantics of the attributes /single entity:relation
    - No spurious tuple generation /no match Attr,-PK/FK

3.1.03. Initial state

- Database design
- Universal relation
  - $R = \{A_1, A_2, \ldots, A_n\}$
  - Set of functional dependencies F
- Decompose R using F to
  - $D = \{R_1, R_2, \ldots, R_n\}$
  - D is a decomposition of R under F

3.1.05. Aims

- Attribute preservation
  - Union of all decomposed relations = original
- Lossless/non-additive join
  - For every extension, total join of $r(R_i)$ yeilds r(R)
  - no spurious/erroneous tuples

3.1.06. Aims (preservation)

- Dependency preservation
  - Constraints on the database
    - X -> Y in F of R, appears directly in R_i
  - Attributes X and Y all contained in R_i
  - Each relation R_i in 3NF
- But what's a dependency?

3.1.07. Functional dependency

- Constraint between two sets of attributes
  - Formal method for grouping attributes
- DB as one single universal relation/-literal
  - $R = \{A_1,A_2,\ldots,A_n\}$
  - Two sets of attributes, X subset R,Y subset R
- Functional dependency (FD or f.d.) X -> Y
- If $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$
  - Values of the Y attribute depend on value of X
  - X functionally determines Y, not reverse necessarily

3.1.08. Dependency derivation

- Rules of inference
- reflexive: if X implies Y then X -> Y
- augment: {X -> Y} then XZ -> YZ
- transitive: {X -> Y,Y -> Z} then X -> Z
- Armstrong demonstrated complete for closures

3.1.09. Functional dependency

- If X is a key (primary and/or candidate)
  - All tuples t_i have a unique value for X
  - No two tuples (t_1,t_2) share a value of X
- Therefore X -> Y
  - For any subset of attributes Y
- Examples
  - SSN -> {Fname, Minit, Lname}
  - {Country of issue, Driving license no} -> SSN
  - Mobile area code -> Mobile network (not anymore)

3.1.10. Process

- Typically start with set of f.d., F
  - determined from semantics of attributes
- Then use IR1,2,3 to infer additional f.d.s
- Determine left hand sides (Xs)
  - Then determine all attributes dependent on X
- For each set of attributes X,
  - determine X+ :the set of attributes f.d'ed by X on F

3.1.11. Algorithm

- Compute the closure of X under F: X+
  - xplus = x;
  - do
    - oldxplus = xplus;
    - for (each f.d. Y -> Z in F)
      - if (xplus implies Y) then
      - xplus = xplus union Z;
  - while (xplus != oldxplus);

3.1.12. Function dependency

- Consider a relation schema R(A,B,C,D) and a set F of functional dependencies
  - Aim to find all keys (minimal superkeys),
  - by determining closures of all possible X subsets of R’s attributes, e.g.
    - A+, B+, C+, D+,
    - AB+, AC+, AD+, BC+, BD+, CD+
    - ABC+, ABD+, BCD+
    - ABCD+

3.1.13. Worked example

- Let R be a relational schema R(A, B, C, D)
- Simple set of f.d.s
- AB -> C, C -> D, D -> A
- Calculate singletons
  - A+, B+, C+, D+,
- Pairs
  - AB+, AC+,…
- Triples
  - and so on

3.1.14. Worked example

- Compute sets of closures

  - AB -> C, C -> D, D -> A
- 1.Singletons
  - A+ -> A
  - B+ -> B
  - C+ -> CDA
  - D+ -> AD
- Question: are any singletons superkeys?

3.1.15. F.d. closure example

- 2.Pairs (note commutative)
  - AB+ -> ABCD
  - AC+ -> ACD
  - AD+ -> AD
  - BC+ -> ABCD
  - BD+ -> ABCD
  - CD+ -> ACD
- Superkeys?

3.1.16. F.d. closure example

- 3.Triples
  - ABC+ -> ABCD
  - ABD+ -> ABCD
  - BCD+ -> ABCD
- Superkeys? Minimal superkeys (keys)?
- 4.Quadruples
  - ABCD+ -> ABCD

3.1.17. F.d. closure summary

- Superkeys:
  - AB, BC, BD, ABC, ABD, BCD, ABCD
- Minimal superkeys (keys)
  - AB, BC, BD