COSC 4820

Designing Schema (3.3)

Kim Buckner

University of Wyoming

Jan. 30, 2023

Designing Schema

- Most of us just sit down and write.
- Lets be honest, we put little planning into the project.
- With database design this results in building in problems.
- These have to do with errors in data consistency because of the design.

The Anomalies

- Data redundancy — unnecessarily repeated data.
- Update anomalies — fail to update related items.
- Deletion anomalies — lose information as a side-effect.
- These must be resolved/eliminated.

Decomposition

- Decomposition is the accepted solution to anomaly resolution.
- Split a relation into 'smaller' relations.
  - Change R into S and T such that all the attributes of R are also the union of the attributes of S and T.
  - $S = \pi_{B_1 B_2 \cdots B_n}(R)$
  - $T = \pi_{C_1 C_2 \cdots C_k}(R)$

Decomposing and BCNF

- This does not say that intersection of the attributes of S and T is empty.
- There may be data repeated in the tables that result from decomposition. But this data is **not** unnecessary.
- Text example on page 87-88, Movies3.
- We will rely on Boyce-Codd Normal Form (BCNF) to eliminate the anomalies

Normal form

- A term you will hear quite a bit in conjunction with database design
- This means that the schema have been "normalized"
- We apply some specific, usually small, set of rules to the schema.
- These rules allow us to guarantee something about the schema.

BCNF (1)

- We start with BCNF because of what it gives us.

BCNF (1)

- We start with BCNF because of what it gives us.
- The claim is that if a database is in BCNF then anomalies cannot exist
- A relation, R, is in BCNF iff
  - Whenever there is a **non-trivial** FD $A_1A_2\cdots A_n \rightarrow B_1B_2\cdots B_m$ for R
  - $A_1, A_2, \ldots, A_n$ is a *superkey*

more BCNF

- Another way: the left side must contain a key for every non-trivial FD
- It can be show that any two-attribute relation is in BCNF
- Four cases to the proof
  - No non-trivial FD's
  - $A \rightarrow B$ holds but $B \rightarrow A$ does not.
  - The symmetric case
  - Both $A \rightarrow B$ and $B \rightarrow A$ hold

Decomposition to BCNF

- Break relation into subsets such that
  - subsets are schemas of relations in BCNF
  - data from original is faithfully represented in the decomposition.
  - means we can reconstruct the original.

BCNF Algorithm (1)

- INPUT: a relation R_0 with a set of FD's S_0
- OUTPUT: a set of relations all in BCNF, from which the original relation could be accurately reconstructed
- Recursively apply the following, starting with $R = R_0$ and $S = S_0$.
    1. If R is in BCNF nothing further needs to be done, return $\{R\}$.

The Algorithm (1)

- Method continued
  2. Let $X \to Y$ be a BCNF violation. Compute X^+. $R_1 = X^+$ and $R_2 = X$ plus attributes of R not in X^+.
  3. Compute the sets of FD's for R_1 and R_2. These will be S_1 and S_2 respectively.
  4. Recursively decompose R_1 and R_2. Return the union of these decompositions.

Example (1)

- Suppose that we have a relation, and of course we'll call it R, that has the following set of attributes:*class, nguns, displ, launch, name, battle, result, date.*
- And it has this set of FD's:

$$
\begin{aligned}
class &\rightarrow nguns, displ \\
name &\rightarrow launch \\
battle &\rightarrow date \\
battle, name &\rightarrow result
\end{aligned}
$$

Example (2)

- First question is "What are the keys to the relation?"
- Why do we need to know this?
- Then we can apply our fancy new algorithm