Lecture 05: Normal Forms

15-445/645 Database Systems (Fall 2017)
Carnegie Mellon University
Prof. Andy Pavlo

Normal Forms

- Now that we know how to derive FDs, we can:
  1. Search for "bad" FDs
  2. If they exist, then decompose them into two tables, repeat for sub-tables
  3. When done, the database schema is normalized
- A **normal form** is a characterization of a decomposition in terms of the properties that satisfied when putting the relations back together
- **Universal relation**: The joining of all tables
- Three properties:
  1. Lossless Joins: Information is not lost or bad information is not created when joining
  2. Dependency Preservation: FDs are not split across relations
  3. Redundancy avoidance: No repeated attributes in tuples
- **History**
  - Ted Codd introduced the concept of normalization and the first normal form
  - Codd Went on to define second and third normal form
  - Codd and Raymond Boyce later defined Boyce-Codd normal form
- The ith normal form is **more restrictive** than the (i-1)th normal form
- **Most common/important ones are the 3rd or Boyce-Codd normal Form**

Types of Normal Forms

1. 1st normal form (1NF): All tables are flat
   - All types must be atomic
   - No repeating groups
2. 2nd Normal form (2NF): "Good enough"
   - Must be in first normal form
   - Any non-key attributes fully depend on the candidate key

3. 3rd Normal form (3NF): Most common
    - Always preserves dependencies (unlike BCNF) but may have some anomalies
4. Boyce-Codd Normal form (BCNF): Most common
    - No redundancies and no lossless join
    - For any FD, if any left hand side attributes are not a super key, the relations are not in BCNF
    - **Some BCNF decompositions may lose dependencies when decomposed relations are joined back together**
5. 4th and 5th Normal Forms: See textbooks
6. 6th Normal Form: Most (normal) people never need this

NoSQL

- The normal forms is usually not how people design databases
- Instead, people usually think in terms of object-oriented programming
- Key tenants of the NoSQL movements
    1. Prior to early 2000s, few people needed high-performance DBMS. In modern day speed is very important
    2. Joins are slow, so we will denormalize tables
    3. Transactions are slow

Conclusion

- You should know about normal forms, they exist
- There is no magic formula for determining the right amount of normalization for an application