Notes

CSC230 Database Technologies for Analytics

21 October 2021

- let's design and build more complicated databases!
  - we need some design rules
  - let's retain power, simplicity of relational model
  - let's not add opportunities for errors to creep into our database
  - avoid redundancy
  - divide data among multiple tables
  - link data in several tables with shared keys
- one value in each cell of our tables
- do not make the programmers who will use your database write long and complicated queries!
- specifically, do not make it necessary for programmers to parse the contents of cells
- parsing means, for example, retrieving an address from a single cell and then splitting that single string into a part that contains the city, a part that contains the state, and a part that contains the zip code
- designer of a database must think about how clients will use the database
  - what kind of information will be important to clients?
  - what kinds of questions will clients ask?
  - will there be different kinds of clients?
  - are clients known? could the number and kinds of clients change in the future?
- our databases are relational
  - what is the relationship among elements of a record?
  - does each table have a well-defined purpose?

  - is there a word (or a few words) that say what kind of thing each row in the table describes?

- the first letter in ACID stands for "Atomic"
  - atomic means indivisible
  - the meaning of indivisible depends upon context
  - "New Jersey" is atomic in a program that records the state in which people live
  - 'sweet potato' is atomic in a database that records the kinds of vegetables in the produce section of a supermarket
  - 'green tea' might not be atomic in a coffee shop's database—the shop sells several kinds of tea, several kinds of coffee, and so on—some clients will ask "What are all the kinds of tea that you have?"
- non-atomic data may also mean mixing types of data—e.g., a number and a string in the same cell
  - mixing types makes it harder for the software to catch errors in data entry
  - if cell contains city, state, and zip code (and type of column is VARCHAR), then software cannot easily confirm that zipcode is all digits
- 2 rules for atomicity
  - only one item in each of table's cells
  - only one column with each kind of data
    * you can have several columns that each hold VARCHAR(40)
    * you cannot have several columns that each hold names of students enrolled in the class
- 2 rules for First Normal Form (1NF)
  - data is atomic
  - each record contains a primary key
- primary keys in a table
  - unique
  - non-null
  - concise (and nothing superfluous—just enough information to uniquely identify a record)
  - immutable (cannot be changed)
  - assigned at the time of a record's creation
- database of movies

  - a column for the screenwriters
  - some screenplays have multiple authors
  - crowding several authors in a single cell...
    * makes queries and updates more complicated
    * forces more work on queries and updates even in the case of cells that contain only a single screenwriter—programmer cannot know which cells might contain more than one screenwriter
  - table name, column name, and key value should specify a datum

- database of lectures
  - 3 lectures in same hall
  - repeated info (e.g., address of hall)
  - hazard with updates—did we update all instances?
  - extra cost of delete—remove event, and lose info about hall
  - 2NF—1NF plus key that determines all other columnvalues
  - (all columns functionall dependent on whole key)
  - what if ticket price depends on duration of talk? (end_time - start_time)
  - add table for ticket prices to get 3NF
  - 3NF—all keys directly dependent on key
  - 3NF is as far as we'll go!