STAT:5400
Computing in Statistics

Intro to relational database concepts

Lecture 15
Oct. 10, 2018

Kate Cowles
374 SH, 335-0727
kate-cowles@uiowa.edu

Introduction to relational database concepts

- database: a system for storing data

- *relational* database model has become the de-facto standard for the design of databases both large and small

- storage of data for use in statistical analysis ideally should follow this model

- today's lecture deals with two related topics
  - efficient storage of data (applies to setting up datafiles for use by SAS or any other analysis system)
  - some aspects of relational database software (such as Microsoft Access)

What is a relational database?

- relational database stores all its data in "tables"

- table is a set of rows and columns
  - set has no predefined sort order for its elements
  - "record" is database terminology for a row or observation
  - "field" or "attribute" is database terminology for a column or variable

Basic concepts

- Primary and Foreign Keys
- Queries
- Referential Integrity
- Normalization

Flat files (how not to store complex data)

- simplest model for a database

- a single table which includes fields for each element you need to store

- you have probably worked with flat file databases, at least in the form of spreadsheets

- waste storage space and are problematic to maintain

Example: customer order entry system

- You're managing the data for a company with a number of customers, each of which will be placing multiple orders.

- Each order can have one or more items

Data that we wish to record for each component of the application

- Customers
  - Customer Number
  - Company Name
  - Address
  - City, State, ZIP Code
  - Phone Number
- Orders
  - Order Number
  - Order Date
  - PO Number
- Order Line Items
  - Item Number
  - Description
  - Quantity
  - Price

Problems with a flat file for representing this data

- Each time an order is placed, you'll need to repeat the customer information, including the Customer Number, Company Name, etc.

- What's worse is that for each item, you not only need to repeat the order information such as the Order Number and Order Date, but you also need to continue repeating the customer information as well.

- Let's say there's one customer who has placed two orders, each with four line items. To maintain this tiny amount of information, you need to enter the Customer Number and Company Name eight times.

- What if the company should send you a change of address?

- unacceptable aspects of flat file storage
  - effort required to maintain the data
  - likelihood of data entry errors causing inconsistency in customer address between records

Solution: use a relational model for the data

- each order entered is related to a customer record
- each line item is related to an order record
- relational database management system (RDBMS) is a piece of software that manages groups of records which are related to one another

Break flat file into three tables

- Customers
  - CustID
  - CustName
  - CustAddress
  - CustCity
  - CustState
  - CustZIP
  - CustPhone
- Orders
  - OrdID
  - OrdCustID (new field)
  - OrdDate
  - OrdPONumber

- OrderDetails
  - ODID
  - ODOrdID (new field)
  - ODDescription
  - ODQty
  - ODPrice

Keys

- key: a field that can be used to identify a record

- key fields may contain
  - data element you are storing or derived from that data
  - an arbitrary value

- example: for the Customers table
  - could use the company name as a key, but if you ever had two companies with the same name, your system would be broken
  - alternatively could use some derivation of the company name in an effort to preserve enough of the name to make it easy for users to derive the name based on the key, but that often breaks down when the tables become large
  - may be easiest to simply use an arbitrary whole number

Primary and Foreign Keys

- primary key: a field that uniquely identifies a record in a table
  - No two records can have the same value for a primary key.
  - Each value in a primary key will identify one and only one record.

- foreign key: represents the value of primary key for a related table
  - foreign keys are the cornerstone of relational databases
  - example: in Orders table, OrdCustID field would hold the value of the CustID field for the customer who placed the order.
    * By doing this, we can attach the information for the customer record to the order by storing only the one value.

Queries

- so far we've
  - broken down our order entry system into three tables
  - added foreign keys to the Orders and OrderDetails tables

- Now, rather than repeating the Customers table data for each Orders table record, we simply record a customer number in the OrdCustID field.

- By doing this, we can change the information in the Customers table record and have that change be reflected in every order placed by the customer.

- This is accomplished by using *queries* to reassemble the data.

- query: a view of data which represents the data from one or more tables

Reassembling the data for analysis and presentation

- human users of the system will only be able to view data in two dimensions
  - become rows and columns in a table either on the screen or on paper

- to see orders placed by our customers
  - link the Customers and Orders tables using the CustID field from Customers and the OrdCustID field from Orders
    * value of the OrdCustID field represents a related record in the Customers table and is equal to the CustID value from that record
  - by joining together the two tables based on this relationship, we can add fields from both tables and see all orders along with any pertinent customer data

Referential Integrity

- purpose is to maintain validity of data

- example: what would happen if you needed to delete a customer?
  - if the customer has orders, the orders will be orphaned
  - must have a means in place to enforce that for each order, there is a corresponding customer

- two ways that that database management system can enfore "referential integrity"
  - by cascading deletions through the related tables
  - by preventing deletions when related records exist

Normalization

- essentially the process of distilling the structure of the database to remove repeating groups of data into separate tables.

- example: we have normalized customers and orders by creating a separate table for the orders

- sometimes need to sacrifice normalization to practicality
  - in Customers table, not really necessary to include the CustCity and CustState fields since a US ZIP Code uniquely defines a city and state in the US
  - to fully normalize the Customers table, would need to remove the CustCity and CustState fields and create a table, perhaps called ZIPCodes, which included these fields
  - then include only the CustZIP field and join the Customers table to the ZIPCodes table in order to reconstruct the full address
  - problem: adds overhead of an additional join in every query where you need to have the full address available

Example of a poorly designed database

- StudentName
- AdvisorName
- CourseID1
- CourseDescription1
- CourseInstructorName1
- CourseID2
- CourseDescription2
- CourseInstructorName2

Normal forms

- First normal form: no repeating groups

- Second normal form: no nonkey attributes depend on a portion of the primary key.

- Third normal form: no attributes depend on other nonkey attributes.

First normal form

- What we're looking for is repeating groups of columns.

- example: students and courses file
  - columns for course information have been duplicated to allow the student to take two courses
  - problem occurs when the student wants to take three course or more

- set of columns in a table with field names that end in numbers xx1, xx2, xx3, etc., is clear warning signal that you have repeating groups in the table.

- A common exception to this would be a table of street addresses, where you might have AddressLine1, AddressLine2, etc., rather than using a single field for multiple line addresses.

Putting the courses data into first normal form

- first table: Students
  - *StudentID*
  - StudentName
  - AdvisorName
- second table: StudentCourses
  - *SCStudentID*
  - *SCCourseID*
  - SCCourseDescription
  - SCCourseInstructorName

- identify primary keys and foreign keys

Putting the data in second normal form

- No nonkey attributes depend on a portion of the primary key.

- applies only to tables where the primary key is defined by two or more columns.

- if there are columns which can be identified by only part of the primary key, they need to be in their own table.

- example: the StudentCourses table
  - the primary key is the combination of SCStudentID and SCCourse ID.
  - table also contains the SCCourseDescription and the SCCourseInstructorName columns, which depend only on the SCCourseID column

Putting these data into second normal form

- first table: Students
  - *StudentID*
  - StudentName
  - AdvisorName
- second table: StudentCourses
  - *SCStudentID*
  - *SCCourseID*
- third table: Courses
  - *CourseID*
  - CourseDescription
  - CourseInstructorName

Adding more detail for realism

- first table: Students
  - *StudentID*
  - StudentName
  - StudentPhone
  - StudentAddress
  - StudentCity
  - StudentState
  - StudentZIP
  - AdvisorName
  - AdvisorPhone
- second table: StudentCourses
  - *SCStudentID*
  - *SCCourseID*
- third table: Courses
  - *CourseID*
  - CourseDescription
  - CourseInstructorName
  - CourseInstructorPhone

Third normal form

- No attributes depend on other nonkey attributes.
- means that all the columns in the table contain data about the entity that is defined by the primary key
- example: in Students table, we have two data items about the student's advisor: the name and phone number. The balance of the data pertains only to the student and so is appropriate in the Students table.
- same logic applies to the instructor information in the Courses table. The data for the instructor is not dependent on the primary key CourseID.

Completing the normalization

- first table: Students
  - StudentID
  - StudentName
  - StudentPhone
  - StudentAddress
  - StudentCity
  - StudentState
  - StudentZIP
  - StudentAdvisorID
- second table: Advisors
  - AdvisorID
  - AdvisorName
  - AdvisorPhone
- third table: Instructors
  - InstructorID
  - InstructorName

- – InstructorPhone
- fourth table: StudentCourses
  - – SCStudentID
  - – SCCourseID
- fifth table: Courses
  - – CourseID
  - – CourseDescription
  - – CourseInstructorID

Summary

- primary and foreign keys, which are used to define relationships

- referential integrity, which is used to maintain the validity of the data

- normalization, which is used to develop a data structure.