Rhea Lee Salonoy
BSCS 2nd Year - Block 1

Advanced Database System:
Activity - Denormalization

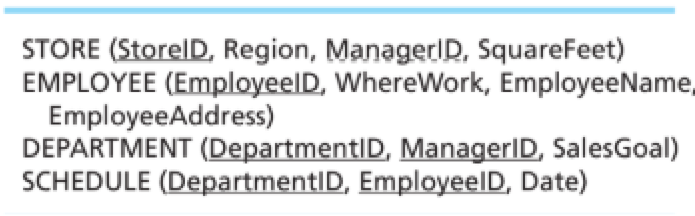
8-48 In a normalized database, all customer information is stored in a Customer table, invoices are stored in an Invoice table, and related account manager information in an Employee table. Suppose a customer changes their address and then demands old invoices with manager information. Will denormalization be more beneficial? How?

Answer: Yes, Denormalization would be more beneficial for this. In the normalized database, updating a customer's address would cause all historical invoices to incorrectly display the new address rather than keeping the original billing information. If we denormalize the invoice table to store the address and manager details directly at the time of creation preserves the original information, ensuring old invoices always reflect what was correct at the time.

8-49 When students fill out forms for admission to various courses or to write their exams, they leave many missing values. this may lead to issues while compiling data. can this be handled at the data capture stage? what are the alternate approaches to handling such missing data?

Answer: Yes, this is best handled at the data capture stage especially right when the student is filling out the form. designing the form to force answers to important questions by making them required, using drop-down menus, and showing clear error messages if something is missed. Alternate approaches when data is already missing is marking is as "Missing" or fill using average answers from other students however this would make the entry invalid.

8-50 Consider the following normalized relations from a data- base in a large retail chain:

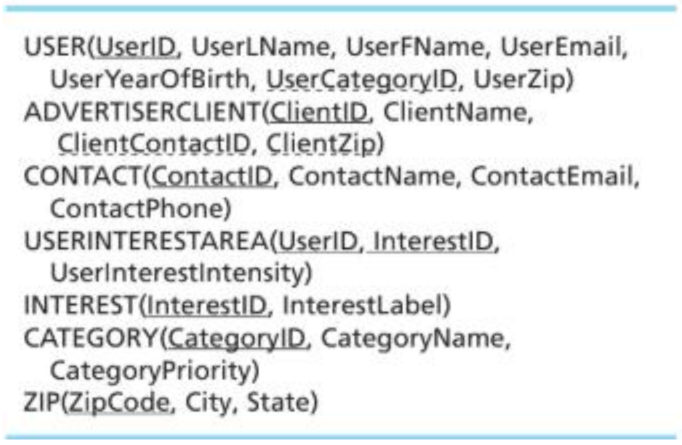


STORE (StoreID, Region, ManagerID, SquareFeet)
EMPLOYEE (EmployeeID, WhereWork, EmployeeName, EmployeeAddress)
DEPARTMENT (DepartmentID, ManagerID, SalesGoal)
SCHEDULE (DepartmentID, EmployeeID, Date)

What opportunities might exist for denormalizing these relations when defining the physical records for this database? Under what circumstances would you consider creating such denormalized records?

Answer: The STORE and EMPLOYEE tables could be partially merged by adding frequently accessed store information (like region) to the EMPLOYEE table, reducing joins for employee location queries. I would implement such denormalization when certain information is searched together very often as this would improve speed performance of reading data.

8-51 Consider the following set of normalized relations from a database used by a mobile service provide to keep track of its users and advertiser customers:


USER(UserID, UserLName, UserFName, UserEmail, UserYearOfBirth, UserCategoryID, UserZip)
ADVERTISERCLIENT(ClientID, ClientName, ClientContactID, ClientZip)
CONTACT(ContactID, ContactName, ContactEmail, ContactPhone)
USERINTERESTAREA(UserID, InterestID, UserInterestIntensity)
INTEREST(InterestID, InterestLabel)
CATEGORY(CategoryID, CategoryName, CategoryPriority)
ZIP(ZipCode, City, State)

Assume that the mobile service provider has frequent need for the following information:

- List of users sorted by zip code.
- Access to a specific client with the client's contact person's name, e-mail address, and phone number.
- List of users sorted by interest area and within each interest area user's estimated intensity of interest.
- List of users within a specific age range sorted by their category and within the category by zip code.
- Access to a specific user based on their e-mail address.

Based on these needs, specify the types of indexes you would recommend for this situation. Justify your decisions based on the list of information needs above.

Answer: Creating indexes like Zip Code and Email in the User table to quickly find users by location or email. Interest Area and Interest Intensity to quickly sort users by their hobbies. Year of Birth and Category to quickly find users in a specific age group and category.
This would make the frequent searches listed in the question run much faster.

8-52 Consider the relations in Problem and Exercise 8-51. Identify possible opportunities for denormalizing these relations as part of the physical design of the database. Which ones would you be most likely to implement?

Answer: The best one to do first is to add the City and State directly to the User and Advertiser tables. This is because "sort by zip code" is a very common need. By doing this, the system won't have to constantly combine the User and Zip tables just to show a user's city and state, which will speed up many queries. Since city and state for a zip code don't change often, this is a safe and effective change.