DATABASES

ADIT—TDDD37

Lab compendium - Normalization and Project

Institutionen för datavetenskap (IDA), Linköpings universitet

Table of Contents

General Information

For all labs you have to hand in a report.

Please print your lab report on paper and check that it is easy to read and understand. Do *not* e-mail reports unless explicitly told so by your lab assistant.

Refer to the course web page for further information and documentation needed for the labs.

Normalisation

Objectives

The purpose of this lab is to arrive at an understanding of the normal forms 1NF, 2NF, 3NF and BCNF.

Background Reading

Lecture notes and book material on normalisation.

The Lab

1) Given a relation R(A, B, C, D)

   a) Supply a set of functional dependencies and a primary key such that R is in 1NF but *not in 2NF*.

   b) Supply a set of functional dependencies and a primary key such that R is in 2NF but *not in 3NF*.

   c) Supply a set of functional dependencies and a primary key such that R is in 3NF but *not in BCNF*.

2) Given the universal relation R={A, B, C, D, E, F} and the set of functional dependencies F={{AB $\rightarrow$ C}; {A $\rightarrow$ D}; {D $\rightarrow$ AE}; {E $\rightarrow$ F}}

   a) What is a primary key for R? Show how you arrive at your solution (applying the inference rules for functional dependencies).

   b) Decompose R in 2NF. Watch out for candidate keys.

   c) Decompose R in 3NF. Watch out for candidate keys.

   d) Decompose R in BCNF.

3) Consider the following relation for published books:
   BOOK(<u>Title</u>#, Title, <u>Author</u>#, BookType, Price, AuthorName, Publisher)
   with the following additional dependencies:
   Title# $\rightarrow$ Title, BookType, Publisher
   Author# $\rightarrow$ AuthorName
   BookType $\rightarrow$ Price

   a) What is the normal form of BOOK?

   b) Decompose it stepwise into BCNF.

Handing in

- Answers to the lab questions.

Project: BrianAir Database

Objectives

This lab combines what you have learnt in the previous labs in one larger project.

Background Reading

For this lab, you need to be familiar with EER-modelling, translation of EER into relational tables, SQL queries, stored procedures, normalisation, and transactions, that is, almost the whole course.

The Lab

The low price airline BrianAir asks you to assist them with designing their flight and booking database. In addition, you have to implement the backend for their web interface for booking flight, i.e. you provide stored procedures and SQL queries that will be embedded in the web middleware.

1) **(at home)** Draw an EER-diagram for BrianAir's database. However, BrianAir's representatives do not have much time to supply you with a thorough requirement specification. They only give you the information below and refer you to their competitors' websites for checking how online-booking is done.

   a. BrianAir uses only one type of airplane that takes 60 passengers. You need not model different airplane types.

   b. BrianAir currently only flies between Lillby and Smallville (and returns) but they will soon expand.

   c. BrianAir operates on a strict weekly schedule. There are no exceptions for holidays. The weekly schedule is valid for one year and may be changed on every January 1st. Of course, there can be several flights per day.

   d. The flight pricing depends on

      - the start and stop destination which has a base price,
      - the day of the week. BrianAir has the same weekday pricing factor for all flights regardless of destination, e.g. factor 4.7 on Fridays and Sundays, factor 1 on Tuesdays, etc.
      - the number of already confirmed/booked passengers on the flight. The more passengers are booked the more expensive becomes the flight. Passenger pricing factor determines the extent of price increase.
      - The price is thus calculated as:

      $\text{baseprice}_{\text{to, from}} \cdot \text{weekdayfactor}_{\text{day}} \cdot \max(1, \text{pass\#}_{\text{flight}})/60 \cdot \text{passengerfactor}$

      Pricing factors and base prices can change when the schedule changes, once per year!

   e. **(simplifying assumption)** BrianAir only issues tickets to passengers older than 18 years. (**optional complication as in the real world**) Passengers can be adults (older than 18 years of age), children, and infants (younger than 2 years of age). A child is only allowed to travel alone when s/he is at least 14 years of age. Infants travel at no cost and do not receive a seat (sit in parent's lap).

f. Customers cannot book more than one year in advance.

g. It is possible to book several people in one booking. One passenger of a booking is the *contact* and must supply phone number and e-mail address.

h. It is a normal case that the one who pays for the flight (the credit card holder) does not necessarily participate in the flight.

i. The payment of the flight is confirmed by issuing a unique ticket number that the passenger needs to bring to the airport instead of a paper ticket.

j. Customers must not end up in a deadlock situation, and double payments are not allowed. However, BrianAir is afraid of denial-of-service attacks to its reservation system and does not want that unpaid and pending reservations block paying customers. The airline demands a reservation strategy that fails in favour of BrianAir: It is possible to reserve seats on a flight that already has many reserved seats; but it is not possible to reserve seats on a flight where all seats are already paid for. At payment time, it must be checked that the reserved seats are really available. If they are not available, the whole booking is aborted and the customer needs to restart from scratch.

k. (**simplifying assumption if you want**) The actual price for a booking is calculated at payment time (at step *h*) and may thus be different (higher) than when the booking started.

l. (**simplifying assumption if you want**) Each person in one booking has the same seat price.

   Other requirements are up to your assumptions. State them in the EER-diagram. Also read *3)* and *4)*.

2) Translate the EER-diagram into a functioning, normalised database.

   a. **(at home)** Translate the EER-diagram from 1) into tables.

   b. **(at home)** Check the tables for their normal forms. Normalise to a suitable normal form. If you decide not to achieve BCNF, motivate this.

   c. **(partially at home)** What additional integrity constraints do you have to implement? Check constraints? Triggers? Plan them !

Before proceeding,
hand in your EER-diagram and translations.

Reading the description of the rest of the exercises in the lab may help you to draw your EER-diagram, though.

Only projects with a previously approved EER-diagram and translations are considered when lab assistants check your final project report.

3) It is now time to fill your database with data. The goal is to arrive at a system that allows the booking of flights. Make sure that you have a simple but working system so you can solve 5). Improve later.
   Create INSERT-statements or STORED PROCEDURES for

   a. setting up new destinations (INSERT),
   b. *For your own convenience, create a script that removes all BrianAir tables from your account and installs them again with suitable data. You need this for testing, reinstallation and in case of having to demonstrate your project. Keep this script updated as you proceed.*
   c. setting up a weekly flight schedule for this and next year (INSERT) and creating the actual flight details from this schedule for one year and one day in advance (STORED PROCEDURE).[1]
   d. setting up a pricing policy for this and next year (INSERT),
   e. reserving seats for a given number of passengers on a specified flight taking into account the strategy described in *1.j* (STORED PROCEDURE). Seat reservation results in a session number that can (and must) be referred in *f-h*. Session number and ticket number are not the same!
   f. adding passenger details (mr/ms, first name, surname, possibly adult/child/infant and at least one contact info per booking) to a booking (STORED PROCEDURE). (**advanced**) Implement a check so that one cannot add more passengers—except infants if you modelled them—to a booking than there are reserved seats in the booking created in *e*. Implement a check so that you have at least one contact per booking.
   g. adding payment details such as amount, name as on credit card, credit card type, expiry month, expiry year, credit card number. *How can you protect the credit card information in the database from hackers? You need not implement this protection.* (STORED PROCEDURE),
   h. confirming previously reserved seats, calculating the actual price of the booking and returning a unique ticket number (the flight is now paid for). The actual payment transaction takes place in the database of the credit card company and need not be considered here. Take into account that the seat reservation may have been invalidated due to other paid bookings and that you may have to abort the whole booking (see *1.j* for the strategy description) (STORED PROCEDURE). (**advanced**) Implement a check that you cannot confirm a booking without payment details.

4) Given a destination, number of prospective passengers and a date, show all flights for this date with their price and the number of available seats according to the reservation strategy (see. *1.j*). The result of this query is used by customers to find available flights. You may create views to solve this query.

[1] Depending on your modelling, this may be very simple or the most difficult procedure of this lab. Do not get stuck on this exercise. You may want to fill the flight table with only a few flights first to be able to proceed. Or you may want to implement a simplified version first, e.g. only create Sunday flights. This procedure may need date functions. Check them out in the lab material at the end of the compendium.

5) Open a new MySQL session in a new window. We call the original session A and the new session B. Do not forget to **set autocommit = 0** in both sessions.
    a. In session A, add a new booking (step e in exercise 3).
    b. Is this booking visible in session B ? Why ?
    c. What happens if you try to modify the booking from A in B ? Explain.
6) Let two customers book seats on the same flight at the same time using two different mysql sessions with **set autocommit = 0** in both sessions. Make sure that there are not enough seats available for both customers (only one customer can succeed). Use your stored procedures from *3)* and experiment with COMMIT, LOCK TABLES, ROLLBACK, SAVEPOINT, SELECT...FOR UPDATE and START TRANSACTION until the strategy of *j* (no deadlocks, no double bookings) is satisfied.
    Do NOT modify your procedures *but put transactional commands* ***around*** *your procedure calls.*
    Deadlocks in MySQL usually time out within a minute.

Handing in

Step 1: At the beginning of the project (before or directly after the first lab session for the project)

- EER-diagram,
- Sketch of table translations (or even the actual SQL code as required for the final report).

Step 2: At the end of the project.

- ***Approved* EER-diagram,**
- Normalisation discussion,
- SQL-code for creating the tables and constraints on them,
- Code for all stored procedures, triggers, functions etc.
- *Example runs* for
    - filling the database with flights,
    - booking a flight.
- Code for *4)*.
- Written answers to *5)*.
- Demonstrate *6)* by showing how the two transactions interleave.
- Show how you have addressed *1.d, 1.e,* and *1.f,* and state how you would address *3.g*.

The lab assistant may contact you for a question and answer session and a possible demonstration of your project. Therefore, do not delete any database objects before your lab is approved in the lab system!