Inf1B Data and Analysis
Tutorial 3 (week 5)

Hutchins-Korte, Simpson

29 January 2008

- Please answer all questions on this worksheet in advance of the tutorial, and bring with you all work, including printouts of code and other results. Tutorials cannot function properly unless you do the work in advance.
- Data & Analysis tutorial exercises are not assessed, but are a compulsory and important part of the course. If you do not do the exercises then you are unlikely to pass the exam.
- Attendance at tutorials is obligatory; please let your tutor know if you cannot attend.
- *Recommended Reading:* Chapter 3 (The Relational Model), 4 (Relational Calculus), and 5 (SQL) of 'Database Management Systems' (Raghu Ramakrishnan and Johannes Gehrke, 2003). Chapter 3 can be found on Google Books (`http://books.google.com/`); copies of Chapter 4 were handed out during an earlier lecture, but spare copies can be obtained on AT level 5; copies of Chapter 5 will be handed out during this week's Thursday lecture (Jan 31st).

1 Introduction

In the previous two tutorials you (a) designed an ER diagram, (b) mapped in to a relational schema, and (c) formulated some queries in relational algebra on the resulting model; all in the context of a gliding competition scenario. This week, we will be using the gliding scenario to specify statements in SQL and queries in relational calculus and SQL. Some of these queries will in fact be the same ones that you formulated in relational algebra last week. For this reason, and possibly to remind yourself of some details of the gliding scenario, you may wish to refer back to the first two tutorial sheets. N.B. Questions 3–6 can be attempted even without completing Questions 1–2.

1.1 OpenOffice Base

In this tutorial, you will be using *OpenOffice Base*. To set up the program on a DICE machine, you need to follow the following steps:

(a) Download `Comp.odb` from the course webpage:
`http://www.inf.ed.ac.uk/teaching/courses/inf1/da/`

(b) From the desktop, choose `Applications -> Office -> Database Development`
(alternatively, you can open a terminal window and type `oobase &`)

(c) When the *Database Wizard* window appears, choose *'Open an existing database file'*.

(d) Click `<Open>`.

(e) Locate `Comp.odb` and click `<Open>`.

The data of both gliding competitions are now loaded and you are ready to start the tutorial. Please note that *OpenOffice Base* may sometimes respond somewhat slowly and may even crash on occasion, so make sure you regularly save any data you wish to keep!

1.2 SQL Trivia

- SQL is *not* case sensitive. So keywords, table names, column names etc., can be written upper-case, lower-case or mixed upper/lower-case without affecting meaning.
- Nevertheless, often keywords are written in upper case, and it is good practice to maintain a consistent approach to the cases used in table and field names.
- In contrast to the above, strings *are* case sensitive.
- Note that strings are written using single quotation marks (apostrophes) as delimiters, *viz:* `'this is a string'`.

2 Queries: Relational Calculus and SQL

The tables of Competition A contain the following data:

Person_A

personId	name	experienceLevel
1	John Jones	pre-solo
2	Fraser McEwan	cross-country
3	Jane Smith	instructor
4	John Jones	cross-country
5	Ann Brown	cross-country
99	Ponder Stibbons	instructor

Team_A

university	feePaid
Edinburgh University	Yes
Heriot-Watt University	No
Unseen University	Yes

MemberOf_A

personId	university
1	Heriot-Watt University
2	Edinburgh University
3	Heriot-Watt University
4	Edinburgh University
5	Edinburgh University
99	Unseen University

Glider_A

callsign	type	numSeats
MF	K8	2
P19	Pirat	1
FNS	DG300	2
CPG	Discus	1

Task_A

compDayNo	startTime	startHeight
1	30/03/07 9:30	5000
2	01/04/07 8:00	3000

Flies_A

personId	callsign	compDayNo	crewCapacity
1	MF	1	P2
1	MF	2	P1
3	MF	1	P1
2	P19	1	P1
2	P19	2	P1
4	FNS	1	P1
4	FNS	2	P1
5	CPG	1	P1
5	CPG	2	P1

The tables of Competition B contain the following data:

Person_B

personId	name	experienceLevel
1	John Jones	pre-solo
2	Fraser McEwan	cross-country
3	Jane Smith	instructor
6	Fiona Stewart	pre-solo
7	Tom Woods	instructor

Team_B

university	feePaid
Edinburgh University	Yes
Heriot-Watt University	No
Napier University	Yes

MemberOf_B

personId	university
1	Heriot-Watt University
2	Edinburgh University
3	Heriot-Watt University
6	Napier University
7	Napier University

Glider_B

callsign	type	numSeats
MF	K8	2
P19	Pirat	1
FNS	DG300	2

Task_B

compDayNo	startTime	startHeight
1	12/05/07 9:00	5000
2	13/05/07 7:00	10000

Flies_B

personId	callsign	taskNo	crewCapacity
1	MF	1	P2
1	MF	2	P1
3	MF	1	P1
2	P19	1	P1
2	P19	2	P1
6	FNS	1	P2
6	FNS	2	P2
7	FNS	1	P1
7	FNS	2	P1

You may assume that both competitions used the same values for personId.

Question 1 - Competition A

(a1) Specify a query in *tuple-relational calculus (TRC)* which retrieves all fields in the `Person_A`-table for people of pre-solo standard.

(a2) Do the same in SQL and run your query on the `Comp`-database. To do this, you need to follow the following steps:

- Click on `Queries` in the left column on the `Comp`-window.
- In the `Tasks`-view, choose *'Create Query in SQL View'*.
- In the window that pops up, click once on the little SQL-button on the right.
- Now enter your query in the white bit.
- To run your query, choose: `Edit -> Run Query`

(b1) Specify a query in *TRC* which retrieves the `personId` for people of at least cross-country standard.

(b2) Do the same in SQL and run your query on the `Comp`-database.

Question 2 - Competition A and B

(a1) Specify a query in *TRC* which retrieves all fields for all entries that appear in the `Flies`-table of at least one of the two competitions.

(a2) Do the same in SQL and run your query on the `Comp`-database.

(b1) Specify a query in *TRC* which retrieves the call signs and types of all gliders who were registered for Competition A, but not for Competition B.

(b2) Do the same in SQL and run your query on the `Comp`-database.

(c1) Specify a query in *TRC* which retrieves the call signs and types of all gliders who were registered for both Competition A and Competition B.

(c2) Do the same in SQL and run your query on the `Comp`-database.

(d1) To prevent unfair advantages, the organizers of Competition B want to prevent setting a task that some of the participants have already flown in Competition A. Specify a query in *TRC* which retrieves the `compDayNos` of all the tasks which have been flown by students participating in Competition B.

(d2) Do the same in SQL and run your query on the `Comp`-database. Ensure that you do not retrieve duplicate entries.

(e1) A gliding inspector wants to check that nobody of pre-solo standard flew illegally (either solo or as P1 in a 2-seater) in Competition A. Specify a query in *TRC* which retrieves the `personId` for all participants of pre-solo standard who flew illegally in Competition A.

(e2) Do the same in SQL and run your query on the `Comp`-database.

(f1) Specify a query in *TRC* which retrieves all possible pairs of pilots (i.e. their personIds) who registered for Competition A and are allowed to fly a two-seater together. Return this as a set of tuples containing P1 and P2. A tuple should contain the `personId` of the pilot as its first element, and the `personId` of the copilot as its second element.

(f2) Do the same in SQL, but return a table with two columns. The first column should contain the `personIds` of P1; the second should contain the `personIds` of P2. Run your query on the `Comp`-database.

3 Commands: Inserting/Deleting Data using SQL

The `Comp`-database was created using two sets of `CREATE`-statements; the set of statements used for the tables of Competition A are shown below. The tables of Competition B were created in the same way.

```
create table Person_A (
      personId        integer,
      name            char(30),
      experienceLevel char(15),
      primary key (personId)
      )

create table Team_A (
      university      char(50),
      feePaid         bool,
      primary key (university)
      )

create table MemberOf_A (
      personId        integer,
      university      char(50),
      primary key (personId),
      foreign key (personId) references Person_A,
      foreign key (university) references Team_A on delete cascade
      )


create table Glider_A (
      callsign        char(5),
      type            char(10),
      numSeats        integer,
      primary key (callsign)
      )

create table Task_A (
      compDayNo       integer,
      startTime       timestamp,
      startHeight     integer,
      primary key (compDayNo)
      )

create table Flies_A (
      personId        integer,
      callsign        char(5),
      compDayNo       integer,
      crewCapacity    char(2),
      primary key (personId, callsign, compDayNo),
      foreign key (personId) references Person_A,
      foreign key (callsign) references Glider_A,
      foreign key (compDayNo) references Task_A
      )

create table TurningPoint_A (
      trigraph        char(3),
      latitude        char(10),
      longitude       char(10),
      primary key (trigraph)
      )

create table TaskTP_A (
      compDayNo       integer,
      trigraph        char(3),
      primary key (compDayNo, trigraph),
      foreign key (compDayNo) references Task_A,
      foreign key (trigraph) references TurningPoint_A
      )
```

Question 3 - Inserting

(a) Specify an SQL command for inserting the following entry into the `Flies_A`-table: (3,MF,2,P2).

(b) Execute the command on the `Comp`-database and describe the information you just added. To execute a command, you need to follow the following steps:

- Choose `Tools -> SQL` in the `Comp`-window.
- Type your command in the top-most white box of the window that just popped up.
- Click `<Execute>`.
- If there are any error messages, they will show up the bottom-most white box.

Question 4 - Problems with Inserting

(a) Specify an SQL command for inserting the following new entry into the `Flies_A`-table: (1,MF,3,P1).

(b) Execute the command on the `Comp`-database. What happens? Why?

(c) What would you need to do to be able to insert the entry intro the `Flies_A`-table?

(d) Specify SQL queries to do this and execute them on the `Comp`-database. *Hint: To insert a value of the datatype* TIMESTAMP*, it has to be in the format 'yyyy-mm-dd hh:mm:ss'. For example, April 2nd 2007 at 7am would be '2007-04-02 07:00:00'.*

Question 5 - Deleting

(a) Specify an SQL command for deleting the Unseen University entry from the `TeamA`-table.

(b) Execute the command on the `Comp`-database. Which other entry from which table also gets deleted as a result of this command? Why?

Question 6 - Problems with Deleting

(a) Specify an SQL command for deleting the entry for John Jones (`personId` = 1) from the `Person_A`-table.

(b) Execute the command on the `Comp`-database. What happens? Why?

(c) What would you need to do to be able to delete the entry from the `Person_A`-table?