CS2312 Laboratory Exercise

Introduction

The CS2312 course covers one particular aspect of computer applications: information management with databases. The lectures cover some theory in this area and the CS2312 laboratory exercises will build a complete working system over the duration of the course. The exercises involve the design and construction of a database. The exercises will be conducted using partially completed database implementations devised around a case study. The case study is incremental, so that later exercises build on earlier ones. Some of the exercises are to be done individually and others in pairs of your choosing.

Relationship to CS2342 and CS2352

The CS2312 course is free standing, but has a strong relationship to two other Computer Science courses, CS2341 and CS2352, in that all three share a case study. Consequently, the case study material should be familiar to you and is partially repeated here, but with an emphasis on the Sweeney Tours holiday operator's database rather than the Travel-In-Style's travel shop application.

We have the challenge of relating your work in CS2341 using UML and designing an application, with your work in CS2312 using EER and designing a database to be used by that application, where some database objects also appear as application objects. Moreover, those taking CS2352 will be encouraged to link their Java application developed using Together-J with their database, through the use of Java DataBase Connectivity classes.

The relationship of EER to UML will be addressed in two lectures in CS2352, with some example class support in CS2312. An introduction to database connectivity and JDBC will be given in CS2312, together with some examples and a guide, but the lab using this will belong to CS2352. I hope that is all clear ☺.

The Case Study

Sweeney Tours is a holiday tour operator offering package holidays to a number of resorts in the Mediterranean. The holidays fly from a number of airports in the UK and have a fairly complicated pricing structure. There is a great deal of information in the brochures that Sweeney Tours produce along with an accompanying price guide. Some of the information is on computer but this is somewhat cumbersome as it is in separate systems that are incompatible, incomplete and difficult to use. It is particularly difficult to find holidays that match a customer's specific requests unless the brochure is browsed carefully.

Sweeney Tours have decided to computerise the information in their brochure and to include booking information to make it easier to:

- Gather specific information from customers and answer questions about the holidays
- Check availability of specific packages
- Give costs of packages
- Take bookings

For simplicity, you may assume that the system is stand-alone and does not have to integrate with any other. Note that many other simplifying assumptions will be made (e.g., that flight information is fixed for the season, and that all packages are weekly, et.). You should be sure to solve the problems as specified not as you imagine they would be "in the real world".

The Exercises

Exercise One (1 session): Part of the database already exists as a relational database implemented in the Oracle Database Management System. The exercise is to learn to access and manipulate this database in order to familiarise yourself with the query language and database environment. This exercise is undertaken *individually*.

Exercise Two (1 session): The database used in exercise one was designed using the Extended Entity Relationship (EER) data model. In this exercise you will be given the design, asked to explain the meaning of the design in words and explain how the relational database is a true representation of the design. This is a pen and paper exercise and *does not require the use of a computer*. This exercise is undertaken *individually*.

Exercise Three (1 session): You will be required to extend the EER diagram you have been given. You will be asked to identify a number of integrity constraints. You will be asked to map your EER design extensions to a relational database design and check the design for ambiguities and anomalies by using normalisation techniques. You may optimise you database design. This is a pen and paper exercise and *does not require the use of a computer*. This exercise is undertaken in *pairs*.

Exercise Four (2 sessions): You may then create the tables that you have designed. You will be asked to populate your relational design. You may then test your database model by executing a given query. You will be required to discuss and review the quality of your design. Test data will be given in electronic form. This exercise is undertaken in *pairs*.

Note that for the exercises done in pairs (3 and 4), **things will run much more smoothly if you choose a partner from the same lab group as yourself**.

Reports

You will have to hand in written reports for each exercise (jointly with your partner in the case of exercises 3 and 4). In each exercise, one mark will be awarded for "good report style". This means neat, legible, complete (every section fulfilled) and with clear explanations in English (where relevant). **You should include a cover sheet with each exercise** that states your full name (in the form Family_name, Given_name), student number and lab group (for both partners in the case of exercises 3 and 4), as well as the exercise number and submission date. E.g.:

```
Horrocks, Ian      S/N: 1234    Group: A
Goble, Carole      S/N: 5678    Group: A

Exercise:   3
Submitted:  1/4/2002
```

This will facilitate the marking process (remember that we have a large number of reports to deal with) and help us to get the marked reports back to you as quickly as possible.

Example Classes

Example classes run alternately with the scheduled laboratory sessions. As exercises two and three are design based, and are reasonably challenging to complete in lab time, several example classes will be used as lab clinics to give you extra time and help. **Please take advantage of this**.

Oracle Documentation

There are three forms of documentation:

1. Essential material is embedded in the text of the exercises, in particular a tutorial in Exercise 1 and a guide to loading data and controlling permissions in Exercise 4;
2. The *Oracle 8i User Guide* and *Oracle 8i Integrity Constraints Guide* will be made available to you all;
3. An *Oracle 8i User Guide to JDBC* will be made available to those students taking CS2352;
4. The full Oracle manuals are online and can be accessed from http://aardvark.cs.man.ac.uk:8899/816_docs/DOC/index.htm. Go to the Oracle8i manual. For the SQL language guide you need to select SQL Server manual pages. The quick reference guide is at http://aardvark.cs.man.ac.uk:8899/816_docs/DOC/sqlplus.816/a75665/toc.htm and the "Users Guide and Reference" is at http://aardvark.cs.man.ac.uk:8899/816_docs/DOC/s qlplus.816/a75664/toc.htm.

Pointers to all the above resources can be found at
http://www.cs.man.ac.uk/~horrocks/Teaching/cs2312/

Exercise One (1 session)

Marks: 10 + 2 bonus marks available
Equipment required: Windows NT PCs using the Oracle DBMS. It should also be possible to use Linux PCs, but some of the following instructions may not apply in this case (e.g., references to the menu bar), and some script files (e.g., to repair damaged databases) may not work on Linux machines.

To be done INDIVIDUALLY.

Purpose

1. To become familiar with Oracle
2. To understand and experience a data dictionary
3. To become familiar with the data definition and manipulation language SQL.
4. To understand referential integrity
5. To explore the existing database for the Sweeney Tours system.

By convention table names are written in uppercase, but Oracle is not case sensitive.

What you need to do this exercise

You will need to understand SQL, Joins, Views, Referential integrity.
You will also need the *Oracle 8 Guide*.

Problem

This exercise has two parts; a tutorial followed by an exercise.

You are the owner of some tables for Sweeney Tours that have been created for you already. They are:

COUNTRY(country, language, timezone, currency)
REGION(region, landtype, country, scenery, page)
RESORT(resort, region, transfertime, beach, beachnum, page)
HOTELS(hotelid, hotelname, sunbeam, ya, rating, stdbasis, page, resort, resortloc, roomtotal)
FACILITIES(facid, description, category)
FACINRESORT(resort, facid)
FACINHOTEL(hotelid, facid, numof)

Each hotel has one standard meal basis on offer (*stdbasis*), always one of Half Board (hb), Bed&Breakfast (bb), Full Board (fb) or Apartment Only (ao). Some hotels are designated "Young and Active" (meaning ideal for 18-30 year olds) (*ya*). A few resorts cater particularly well for children by running a 'SunBeam Club' (*sunbeam*) where parents can leave their children. These clubs are associated with particular hotels in the resort. Each hotel has a distance from the centre of the resort included (*resortloc*). The transfer time from the airport to the resort is included (*transfertime*).

Resort and/or Hotels have facilities such as swimming pools, discos, horse riding, waiter service, kids playgrounds, telephones in bedrooms etc. Facilities fall into a number of different categories:

accommodation	a	entertainment	e
sport	s	meals	m
children	c	bedroom	b

Tutorial

Starting SQL*Plus

Launch the program SQL Plus 8.0. On Windows machines:

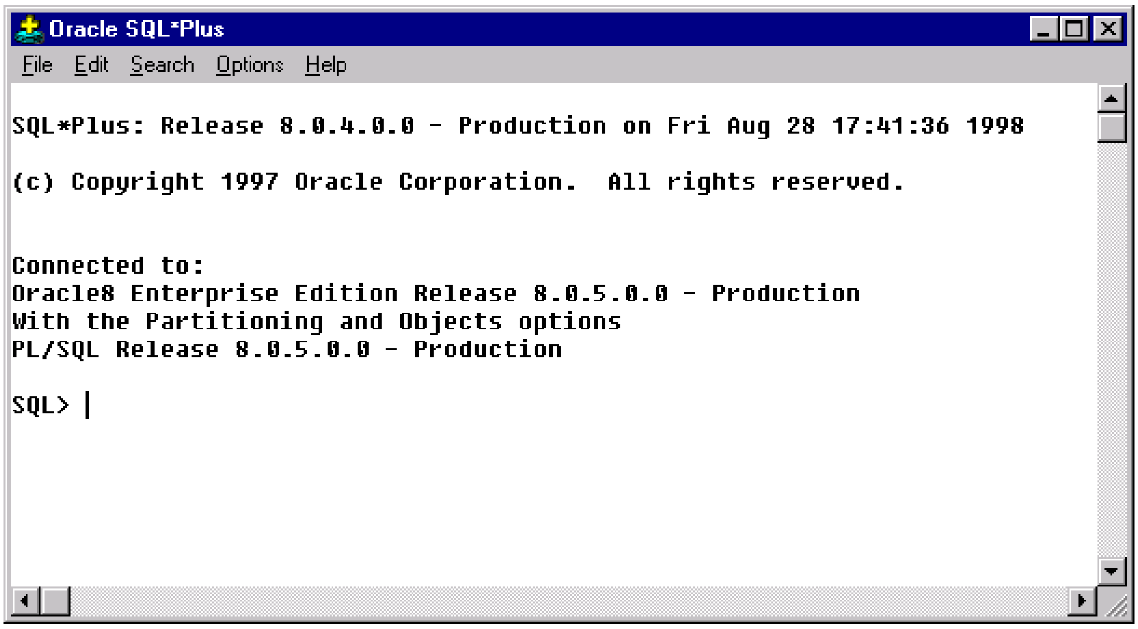
```
Start→Programs→Oracle for Windows NT→SQL Plus 8.0
```

and on Linux machines:

```
/home/oracle/sbin/sqlplus
```

You will be prompted for your user name, Oracle password and host machine. The password is NOT the same as your UNIX one, and will have been allocated to you. The "Host String" (Windows machines) is **`teach`**

Success should lead to the following window:



```
SQL*Plus: Release 8.0.4.0.0 - Production on Fri Aug 28 17:41:36 1998

(c) Copyright 1997 Oracle Corporation.  All rights reserved.


Connected to:
Oracle8 Enterprise Edition Release 8.0.5.0.0 - Production
With the Partitioning and Objects options
PL/SQL Release 8.0.5.0.0 - Production

SQL> |
```

If there is any problem or error message, call a demonstrator. The SQL prompt is:

```
SQL>
```

When you see the SQL prompt simply type in your SQL command, or paste commands in from your favourite text editor (**highly recommended**). Separate words by spaces or tabs. Commands can appear in upper or lower case. There are also special SQL*Plus commands for formatting, setting options and editing and storing SQL commands (see the Oracle 8 Guide).

Listing a Table Definition

One of the tables you own is REGION. Use the SQL*Plus command `describe` to find out the definition of REGION, which is held in the Data Dictionary.

```
SQL> describe region
```

Produces output:

```
Name                             Null?    Type
-------------------------------- -------- ----
REGION                           NOT NULL CHAR(13
LANDTYPE                                  CHAR(4)
PAGE                                      NUMBER(3)
SCENERY                                   CHAR(9)
COUNTRY                          NOT NULL CHAR(8)
```

Getting started with SQL

Selecting the entire relation

To select all columns and all rows in the REGION table, use:

```
SQL> select * from region;
```

Ending a SQL command

You can end a SQL command in three ways but in general you will use a semicolon (;) at the end of the SQL command. This means that you want to run the command. Press *return* to activate the command. If you mistakenly press *return* before typing a semicolon, just enter it on the next line.

Halting a Command

Use `<ctrl-c> or menu bar option File | Cancel`

SQL commands over more than one line

Enter the first line of the command to list the region name and landtype for Greek regions:

```
SQL> select region, landtype
```

If you make a mistake use *delete* to erase it and re-enter. Press *return* to move to the next line. (You can type the whole query on one line and it will wrap but you won't be able to edit characters on the previous line).

SQL*Plus will display a '2', the prompt for the second line. Enter the second line of the command.

```
2  from region
```

SQL*Plus will display a '3', the prompt for the 3rd line. Enter the third line of the command.

```
3  where country = 'greece';
```

Notice the single quotes (') around the string literal `greece`. The semicolon (;) means that this is the end of the command. Notice that `greece` is in lower case because the data in the tables is expressed as lower case. Press *return* (note that if you are using a text editor you can enter the whole multi-line command with a single “paste” operation). SQL*Plus processes the command and displays the results on the screen.

```
    REGION        LAND
    ------------- ----
    rhodes        isle
    crete         isle

    2 rows selected.

SQL>
```

Notice that the column name (attribute) LANDTYPE has been truncated in the output to LAND to match up to the length of the data in the columns.

After displaying the results and the number of records retrieved, SQL*Plus displays the command prompt again. SQL*Plus has syntax rules for commands. If you break a syntax rule you will get an error and the command will not execute. You can divide your SQL command into separate lines at any point you wish, so long as the individual words are not split across lines. Don't leave any blank lines in your command because two returns is also counted as meaning 'end of command'.

Editing SQL commands

Typically you will enter an SQL query on several lines, execute it and then want to alter it and try again. The SQL command is in the command buffer, which can be edited (via the menu bar), but it is **highly recommended** that you edit your commands by using a text processing/editor application (e.g. Word, WordPad or NotePad). Type your commands into the text editor and then just cut-and-paste. The line numbers may flash up and make the query look a bit of a mess, but they don't affect its execution. This way you can also maintain a history of different queries you have tried and edit/retry complex queries until you get the syntax right.

Remember to include the semicolon and *return* to complete the command!

Saving your queries for reports

An easy option is to cut and paste the queries into a Word document as you go along. Alternatively you can use menu bar option File | Save or Save As.

Saving the results of your queries for reports

Again, an easy option is to cut and paste the answers into a Word document as you go along. Alternatively you can use menu bar option File | Spool | Spool File to spool the query results to a file. Results will be spooled to the file until you switch spooling off using menu bar option File | Spool | Spool Off.

Sorting results

To sort the rows in a table in cascading ascending order (i.e. sort first by X and then within that sort by Y, and then within that sort by Z) add the clause ORDER BY X, Y, Z. at the end of the select statement. e.g.

```
SQL> select *
     from region
```

```
      order by country, region;
```

This is described in your SQL notes.

Interpreting Error Messages

An example: if you misspell the name of a table while entering a command, an error message will tell you that the table or view does not exist:

```
SQL> describe rigion;

Object does not exist
```

If you need an explanation other than the message, do the following:

- If an error is a numbered error for the SQL*Plus, look up the message in Appendix A of the SQL*Plus Guide.
- If the error is unnumbered, look up the syntax for the command that generated the error in Chapter 6 of the SQL*Plus Guide, or SQL Language Manual.
- If the error is a numbered error beginning with the letters "ORA", this will have to be looked up in the *ORACLE ERROR Messages and Code Manual* or the *ORACLE Installation and user's manuals for Unix*. You can also use an Oracle program to look up information about errors. 'oerr' will give some details about a message. To use it type:

  `oerr` *`facility number`*

  where *facility* is the three letter code at the start of the error message. Eg: ora, and *number* is the error number. Typing `oerr` on its own will give some instructions. In other words, its bad news.

Getting the results to fit on the screen

When the results get large two commands can be used to make the results easier to read.

```
SQL> set pagesize 25
SQL> set linesize 150
SQL> column <column_name> format a20
```

Sensible use of the above (possibly with different numeric values and column names) can help to make your reports look neater. There are many such SQL*Plus commands that control the environment (as opposed to the SQL commands that are the query language).
Choosing from menu bar Options | Environment, gets you to a whole bunch of environment variables that you can alter through the dialogue box.

The Oracle Catalog and Data Dictionary

Oracle has a data dictionary containing all the management information on the objects (tables, views, indexes etc) in the database. You cannot alter this dictionary directly, but you can look at it: the data is held as tables so the usual SQL commands work. The ***Oracle8 Server Administrator's Guide*** has a list of all the Data Dictionary Views and the Oracle reserved words.

Listing the tables you own

The data dictionary table (well actually its a view) USER_TABLES contains information on all the tables the user (i.e. you) own. As its just a table itself you can access the information using SQL statements. E.g.:

```
SQL> select table_name from user_tables;
```

Notice that the data strings in the dictionary are in upper case, so any 'where' clauses for matching values will have to be aware of this. Have a look at USER_TABLES and see just how much metadata (data about data) a table has.

Listing the tables you have access to

You can grant permission for other users to access your tables and vice versa. (Full details are in the ***Oracle 8 Guide*** and in Exercise 4.) You have been granted permission to select-only from Carole's tables, some of which are the same as the Sweeney Tours database you have, and some of which are the University Database case study used in the lectures. The dictionary table ALL_TABLES lists all the tables you can access.

```
SQL> select table_name from all_tables
     where owner = 'CAROLE';
```

Don't worry about the some of the table names being the same as yours—yours will always be accessed in preference.

Creating tables with integrity constraints

The SQL create statements used to create the Sweeney Tours tables are in a file **$L:\courses\CS2312\ex1\create-tables.sql** that you can look at (**/opt/info/courses/CS2312/ex1/create-tables.sql** for Linux users). An excerpt is below:

```
create table resort
   (resort char(15) constraint resort_pk primary key,
   page number(3),
   region char(13) not null,
   beach char(7),
   beachnum number(2),
   transfertime number(3),
   constraint region_fk foreign key (region) references
          region(region) on delete cascade);

create table hotels
   (hotelid number(5) constraint hotels_pk primary key,
   hotelname char(20) not null,
   resort char(15) not null,
   page number(3),
   stdbasis char(2) check (stdbasis in ('hb', 'bb', 'fb')),
   roomtotal number(3),
   rating number(1) check (rating between 1 and 5),
   sunbeam char(1) check (sunbeam in in ('y', 'n')),
   ya char(1) check (ya in ('y', 'n')),
   resortloc number(2,1),
   constraint resort_fk foreign key (resort) references
          resort(resort) on delete cascade);
```

Integrity Constraints in the Data Dictionary

`Describe` accesses information in the ORACLE data dictionary. Notice that this doesn't tell us anything about keys or any integrity constraints defined on the table. We have to SELECT from the data dictionary to find out such information. To check out the integrity constraints on the REGION table we need to look at two dictionary views:

USER_CONSTRAINTS	constraints on the tables
USER_CONS_COLUMNS	constraints on columns in the tables

(There are equivalent views for tables not necessarily owned by you called ALL_CONSTRAINTS and ALL_CONS_COLUMNS.)

These views are pretty confusing to look at.

```
select constraint_name, constraint_type,
       table_name, r_constraint_name, delete_rule, status
from   user_constraints
where  table_name = 'HOTELS' or table_name = 'RESORT';
```

```
CONSTRAINT_NAME     C TABLE_NAME   R_CONSTRAINT_NAME     DELETE_RU  STATUS
------------------------------ - ------------------------------ --------------
-----
HOTELS_PK           P HOTELS                                        ENABLED
REGION_FK           R RESORT       REGION_PK             CASCADE    ENABLED
RESORT_FK           R HOTELS       RESORT_PK             CASCADE    ENABLED
RESORT_PK           P RESORT                                        ENABLED
SYS_C001289         C RESORT                                        ENABLED
SYS_C001292         C HOTELS                                        ENABLED
SYS_C001293         C HOTELS                                        ENABLED

7 rows selected.
```

Notice that the literal values in the dictionary table are all in uppercase, so when they are queried for they must be in uppercase in the select statement. You might get a few more than those listed here...don't worry about it.

The constraint_type can be:

P	Primary key
R	Foreign Key (Referential integrity)
C	Constraint, (SQL expression such as 'not null' or 'check in ('cis','cs')')

Rows 1 and 4 state that the HOTELS and RESORT have a primary key defined and that it is enabled. The constraint has been named by the owner with a constraint_name.

Rows 2 and 3 state that RESORT and HOTELS have foreign keys targetted on a primary key. So row 3 says that HOTELS has a foreign key trained on the primary key constraint RESORT_PK which we know is define on resort table because it says so in row 4.

- delete_rule for referential integrity can be no action (which means restrict) or cascade.
- status can be enabled or disabled.

The constraints named SYS_*xxxxx* are created by Oracle for unnamed constraints.

```
SQL> select constraint_name, constraint_type,
     table_name, search_condition
from user_constraints
where constraint_type = 'C';

   CONSTRAINT_NAME           C TABLE_NAME      SEARCH_CONDITION
   ------------------------------------------------------------
   SYS_C001235               C REGION          COUNTRY IS NOT NULL
   SYS_C001238               C RESORT          REGION IS NOT NULL
   SYS_C001241               C HOTELS          HOTELNAME IS NOT NULL
   SYS_C001242               C HOTELS          RESORT IS NOT NULL
```

These tuples all have a constraint type = 'C'. This represents a check constraint - in this case to check that the attributes are NOT NULL.

USER_CONSTRAINTS just tells us that there are constraints on the tables, but not the columns that the constraints act upon. To see this we have to look at view USER_CONS_COLUMNS.

```
SQL>  select constraint_name, table_name, column_name, position
      from user_cons_columns
      where table_name = 'RESORT' or
            table_name = 'HOTELS';

   CONSTRAINT_NAME      TABLE_NAME      COLUMN_NAME  POSITION
   --------------------------- ----------------------------
   HOTELS_PK            HOTELS          HOTELID      1
   REGION_FK            RESORT          REGION       1
   RESORT_FK            HOTELS          RESORT       1
   RESORT_PK            RESORT          RESORT       1
   SYS_C001289          RESORT          REGION
   SYS_C001292          HOTELS          HOTELNAME
   SYS_C001293          HOTELS          RESORT

   7 rows selected.
```

This tells us that the constraint RESORT_PK refers to column RESORT in the table RESORT. The name suggests that this is a primary key constraint. It also tells us that RESORT_FK in the HOTELS table refers to column RESORT, suggesting that this is a foreign key constraint referring to the RESORT table's primary key (identified by RESORT_PK). We can infer much of this information from the constraint names—that is why it is a good idea to give constraints sensible names. (Note that POSITION refers to the position of a column in a composite key.)

Joins

There follows two queries that link together two tables. What is the difference between them?

```
SQL> select *
     from country, region;

SQL> select *
     from country, region
     where country.country = region.country;
```

Ambiguous Names

Try the following statement:

```
SQL> select country
     from country, region
     where country.country = region.country;
```

What is the problem and how should you fix it?

Views

Views are 'virtual relations' (see lecture notes) that are created dynamically by a query but are treated as if they were a base table. To create a view that selects hotels with less than 50 beds:

```
SQL> create view smallhotels as
     (select hotelid, hotelname from hotels
     where roomtotal < 50);
```

and now we can treat SMALLHOTELS as if it were a table, e.g.

```
SQL> select * from smallhotels;
```

The data dictionary view USER_VIEWS shows you how the view is stored-select from it and take a look after you have defined your view.

Deleting Tables and Views

To delete a view use

```
SQL> DROP VIEW <view_name>;
```

To delete table use

```
SQL> DROP TABLE <table_name> cascade constraint;
```

You have to use the cascade constraint clause because of all the integrity constraints attached to tables, otherwise you have to drop tables in a strict order. If you haven't any referential integrity constraints attached then forget the clause.

In fact to get rid of any Oracle object use

```
SQL> DROP (oracle object) objectname;
```

Oracle objects include TABLE, SYNONYM, VIEW, INDEX etc

Deleting, Updating and Inserting Tuples

The statements are described in the ***Oracle 8 Guide*** and SQL Manual as well as your text book and lecture notes. Briefly (and not necessarily realistically!):

```
SQL> insert into country
     (country, language, timezone, currency)
     values ('france', 'french', 1, 'franc');

SQL> delete from country where language = 'french';
SQL> update country set timezone = timezone + 1
     where language = 'spanish';
```

Reloading the Tables when you've screwed up

If you've managed to delete or otherwise damage the data in the tables for the Sweeney Exercise. The best thing to do is to drop all the Sweeney Tours tables and start again. An sql command file has been created for you for this very purpose, found on drive L. The procedure is as follows. First, **exit from SQL*Plus**. Then, open and run **L:\courses\CS2312\scripts\repair-tables.bat** (for Linux users, execute the command **/opt/perl/bin/perl /opt/info/courses/CS2312/scripts/repair-tables.pl**).

Big Hint

The main problems for new SQL programmers concern syntax errors that are difficult to identify from the error messages. Therefore:

1) Break your queries into separate lines, and build queries up slowly, bit by bit.
2) Use cut and paste from a text editor so you can edit complex commands.

Exercise

1. Work your way through the tutorial. Make sure you know the tables that you own, the data in them and their definitions.

   a) List the primary, the foreign keys and their target primary keys for each Sweeney Tours table.

2. Using SQL solve the following queries:

   a) List the names of the Spanish island resorts and their brochure pages;

   b) List for each hotel the hotel name, its region, its resort and its number of rooms sorted by region, resort and number of rooms;

   c) There are 3 resorts that actually don't have any hotels in the database. List their names.

3. Define a view for query 2a and use it to list the names of Spanish island resorts and the names of their hotels for the hotels that have a 'swimming pool' or a 'disco' or both.

4. Hotel Splendide has *hotelid* 99999. It is in the resort of Antalya in the region of Atheka in Turkey.

   a) If Hotel Splendide was inserted what would be the consequences for the tables? Which ones would you have to alter and why?

   b) In what order would you have to insert the rows?

5. Delete 'portugal' from the COUNTRY table. What is the consequence to the database and why?

Bonus question for extra marks.

6. If the foreign key constraint from resort to region was altered from DELETE CASCADE to the default delete restrict (which appears in the dictionary as NO ACTION) what would the effect of question 5 have been?

Report

You must hand in a report containing the following:

1. The primary and foreign keys of the Sweeney Database (1 mark).
2. The SQL for each of the 3 queries and the results (3 marks).
3. The view definition and query using it (1,1 marks).
4. Explanation of the consequences of adding Hotel Splendide to the database (1 mark).
5. Explanation of the effects of removing the tuple for 'portugal' from the COUNTRY table (2 marks).

Optionally include:

6. The effects of NO ACTION instead of DELETE CASCADE (2 bonus marks)

One mark will be awarded for good report style. Good report style means:

- neat, without curry stains and not chewed by the dog;
- complete with every section fulfilled and clear explanations in English

Exercise Two (1 session)

Marks: 10 + 2 bonus marks
Equipment required: Pen and paper.
To be done INDIVIDUALLY.

Purpose

1. To understand the Extended Entity Relationship schema for the Sweeney Tours system.
2. To understand the mapping of the EER model to a relational model, and explore the consequences to the integrity constraints.

What you need to do this exercise

You will need to understand the EER model, the relational model, referential integrity, keys.

Problem

Below is an EER schema which models the brochure information implemented in the relational schema in Exercise 1.

1. Understanding the EER schema for the Sweeney Tours system.

To demonstrate that you understand the information represented by the schema, explain using REGION, RESORT, HOTEL and FACILITIES as examples:

i) attributes, entities and relationships
ii) cardinality & participation constraints on relationships
iii) category relationships: supertypes and subtypes.

Don't write more than one page to answer question 1.

2. Understanding the mapping of the EER schema to the relational schema of exercise one

Explain how:
i) REGION & RESORT maps to the relational schema, clearly identifying the candidate, primary and foreign keys.

ii) the cardinality and participation constraints on relationship REGIONLOC are, or could be, implemented in the relational model.

iii) HOTEL and FACILITIES, OTHER FACILITIES and ACCOMMODATION FACILITIES maps to the relational schema, clearly identifying the candidate, primary and foreign keys.

iv) the cardinality and participation constraints on relationship FACINHOTEL are, or could be, implemented in the relational model.

3. Extending the schema for Airports

Each holiday region is served by exactly one airport. A holiday airport can serve more than one holiday region. All holiday makers fly to their holiday destination airport from an UK airport. Each airport has a unique name and a unique 3 letter code. The duration of a flight between the UK airport and holiday airport is required.

a) extend the EER schema to include this, and
b) map your EER schema to a relational schema, clearly identifying primary, candidate and foreign keys.

4. Bonus question.

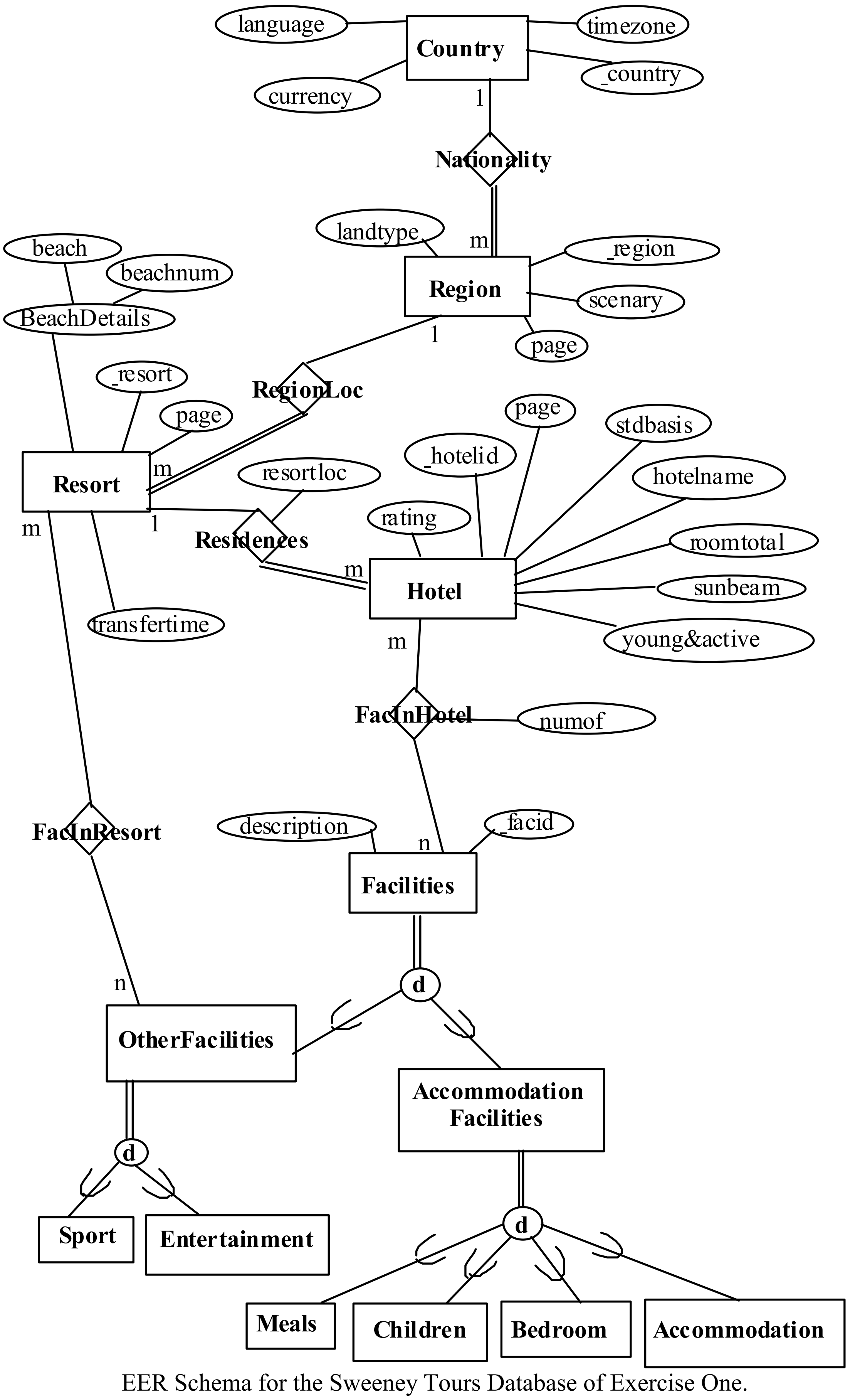
The EER schema is much more expressive than the relational schema, so that some semantic information has been lost in the mapping and will have to be replaced by tests. This is the case with FACINRESORT and OTHERFACILITIES. What has been lost and how could the semantics be recovered?

Report

You must hand in the following:
1. The required explanation of the EER schema (2.5 marks).
2. The required explanation of how the EER schema maps to the relational schema with the candidate, primary and foreign keys identified (3.5 marks).
3. The new part of the EER schema for Airports (2 marks) and an explanation of its mapping to a relational schema. (1 mark)

Optionally include:
4. An explanation of what has been lost in the mapping from EER to relational schema, and of how the semantics could be recovered via integrity constraints for the facilities category relationships in the EER schema (2 marks).

1 mark will be given for good report style.



EER Schema for the Sweeney Tours Database of Exercise One.
Explicit Constraints: Hotels have only twin-bedded rooms.

Exercise Three (1 session)

Marks: 20
Equipment required: Pen and paper.
To be done in **PAIRS**.

Purpose

1. To practice data modelling.
2. To extend the data model for the Sweeney Tours system.
3. To identify integrity constraints in the model.
4. To map the abstract model to a relational model and explore the consequences of the integrity constraints.

What you need to do this exercise

You will need to understand EER schemas, the relational model, functional dependencies, Boyce-Codd Normal Form and explicit integrity constraints.

Problem

The Sweeney Tours database of the previous exercises only covers brochure information. It must now be extended with flight, package, booking and timetabling information given below. Concrete examples are given to illustrate specific points and should be read carefully.

Note that the specification given below makes many simplifying assumptions, and would not necessarily be appropriate for a "real" travel agent; your design should reflect the specification, and not your idea of what would be necessary "in the real world".

What is a package holiday for Sweeney Tours?

A package holiday combines three things into one package
1. a flight on a specific week
2. a hotel
3. a predefined number of nights

which have been costed per person into one predefined package. Often a party of people book a package together. Sweeney Tours only deals with package holidays and does not deal with independent travellers, or with booking flights only or hotel bookings only.

Information can be divided into:
- Generic information about flights and packages which is static and fixed for the season, and hence doesn't change every week;
- Specific information and flights and packages which are the actual specific weekly instances with dates and availability figures of the generic flights and packages.

Generic Fixed Flight Information

Fixed flight information is ***static and is fixed for the whole season***. A flight flies between one UK airport and one foreign airport. Each flight has a unique flight code, UK departure day and time, and UK arrival day and time (the day the flight arrives back in the UK). All times are local times and are based on the 24 hour clock. Each flight has a maximum number of seats that have been reserved by Sweeney Tours.

A UK airport obviously has more than one flight from it (otherwise it would be very underused). A UK airport might have several flights flying to the same foreign airport at different times (if its a popular holiday destination). A UK airport might have several flights to different foreign airports. A foreign airport may have more than one flight from it going to different UK airports.

Example:
Flight PL2S always flies from Gatwick (UK) to Palma Airport (Majorca) every Tuesday at 21:00 hours and always returns on the following Wednesday (the next day), arriving at Gatwick at 06:00.

Flight PL2E flies from Gatwick to Palma every Tuesday at 16:30 and returns on the same Tuesday, arriving at Gatwick at 22:50.

Flight PM6S flies from Manchester to Palma every Saturday at 23:59 and returns the following Sunday (the next day) at 07:30.

Generic Packages

The holiday package is made up of the outward and homeward flight and the hotel (we don't worry about duration as all holidays are for one week). Each hotel is associated with more than one flight and a flight is associated with more than one hotel. On a flight a fixed block of seats are reserved for a hotel throughout the season. In a hotel a fixed block of rooms are reserved for a flight throughout the season. (By the way, the number of rooms and the number of seats don't necessarily match as sometimes rooms are under-occupied).

Example,
Residents of Hotel 33 come in and go out on flight PL2S. The reason for this is that you can always be sure of the change-over day. The room changes occupancy on a Tuesday, when the new group arrives on the flight's outward leg and the previous group returns on the same flight's homeward leg. Consequently, the holiday-makers in Hotel 33 always fly from Gatwick. However, flight PL2S also carries passengers for Nova Hotel.

Specific Weekly Flight Information

Weekly flight information is ***dynamic and changes each week***. Every package flight has one and only one actual weekly flight for every week of the season. Each weekly flight has a departure date for the outward leg and the UK arrival date for the homeward leg.

Each weekly flight flies to its holiday destination with new holiday makers, turns around and takes holiday makers who came in on that flight, 1 week previously, back home. So, when a flight is booked a holiday-maker travels out on the weekly flight's outward leg on one date, has the holiday, and returns on the following weeks flight's homeward leg.

Example:
A particular instance of flight PL2S is on 16th July. The outward leg of the flight is on Tuesday 16th July at 21:00, the plane returns to the UK and arrives on Wednesday 17th July at 06:00 (it being a night flight).

The following week, the outward leg on Tuesday 23rd July turns round and arrives back in the UK on Wednesday 24th July. Holiday makers who have booked a 7 night holiday who flew outward on the previous Tuesday 16th July PL2S flight, will get the PL2S Tuesday 23rd July homeward flight, arriving in the UK on Wednesday 24th July at 06:00.

Weekly Package Availability

One of the most important aspects of the system is checking whether a package is available or not. A package's availability for the week depends on the:

1. availability of seats on the outward and homeward bound weekly flights;
2. availability of rooms in the hotel for the weekly flight.

Another way of thinking about it is "how many places do I have free on a package'.

Holiday Durations & Package Costs

Sweeney tours only deals with packages of 7 nights duration (to make the exercise simpler!). Each week has a start date and an end date. Some weeks are more expensive than others. The basic cost per person of a package is based on the week of departure and the hotel. There is no difference in cost for adults or children.

Example:
MAJORCA

Hotel	Hotel 33 BB	Hotel Nova BB
Hotel id	21007	21005
No of Nights	7	7
....		
19 Jul -25 Jul	322	284
26 Jul- 1 Aug	329	299
1 Aug-7 Aug	329	299

Booking

A booking has a unique booking identifier and is made in the name of one person as the contact. A booking date is also required along with the cost of the whole party's package. Sweeney Tours needs to know the total number in the party travelling. As rooms are double, you must record how many rooms are being booked in case some will have single occupancy.

When a party books a flight Sweeney Tours must make sure that the availability of seats and rooms is good enough, and that availability will be adjusted after the booking.

Volumes, Update and Query Profiles

A database has to cater for update transactions and queries otherwise it is not terrible useful. Other kinds of information has to be taken into consideration when you make database design decisions that may lead to compromises on a pure design that just supports the functionality described above. In particular performance, efficiency and ease of use.

Update profile

Once a year all the flight, hotel and package details, including dates and schedules, are created or altered for the next season. Once this is complete most of this information is expected to remain unchanged, like the flight dates, hotel names etc, until the next season. Once the season commences agents will query for holiday packages and make bookings, which means that the availability figures for flights, hotels and packages will constantly change. Costs of packages may also change.

Query and transaction profile

Obviously you would expect that many potential packages will be browsed and queried before a booking is actually made; the usual ratio is 10:1 (i.e. 10 investigations for every 1 booking). Many queries will never result in a booking. Its important to understand how the data in the database will be used. The case where the holiday-maker knows all the details of their specific holiday and just to go ahead and book is *very rare*.

- They usually know the rough interval of dates that they can travel between, the kind of hotel they want and a rough cost;
- They might know the hotel they want to book and need to know when its available or if its available between certain dates;
- They might not know the hotel but they know the resort they want to book and need to know when its available or if its available between certain dates;
- They might not know the hotel or resort, just the dates they want to and they need to know what is available between those dates;
- They might just know the country they want to book;
- They might be restricted on the airport they can fly from, or that might be flexible.

The most common requirement for a family is "somewhere hot and good for the kids".

In addition the travel agency has be able to ask questions such as:

- How may bookings has Hotel X, or Resort R or Country C had?
- How many seats are available on flight F?
- How many places are available on package P?
- How much money have we made?

Remember that bookings are usually made in an interactive and exploratory way (see below). The user of your system is the Tour Operator not the Holiday-Maker.

Scenario

Here is a typical transactional scenario.

Holiday-maker: I want to book 7 nights for 2 adults at Hotel 33 in Palma Nova, Majorca on about Wed 12th June flying from Gatwick.

Tour operator: There are no Wednesday flights, the nearest is Tuesday 11th June at 21:00 and 16:30.

Holiday-maker: I'd prefer the day-time flight.

Tour operator: There are no seats available on the 16:30 for Hotel 33, so you can either take the night flight which does have seats available or see if there is any other accommodation in Palma Nova that hasn't been booked up for the daytime flight.

Holiday-maker: I'd rather stay with Hotel 33. When is it available to my preferred date? (i.e. when can I get a flight?)

Tour operator: Saturday 15th June at 08:00 is available.

Holiday-maker: Is there a resort near Palma Nova with accommodation available for 11th June?

Tour operator: Yes, Magaluf is nearby and has vacancies but the hotels aren't really like Hotel 33 and they are more expensive.

Holiday-maker: Are there any daytime flights from anywhere in the UK near Wed 12th June that have places for Hotel 33?

Tour operator: The best choice is flight PB2A on Tuesday 11th at 16:00 from Birmingham which has 2 places.

Holiday-maker: I'll book that on 'option'.

Of course the Tour Operator would suggest a number of alternatives rather than be interrogated by a client, but you get the idea.

Volumes

There are 8 countries with 23 regions and 101 resorts. There are 188 hotels. The average number of hotels per resort is 2, the minimum is 0, the maximum is 9. The holiday season covers 27 weeks. There are 21 holiday airports and 8 UK airports. There are 124 different flights and during the season 3348 actual flights. Each airport has between 1 and 6 flights per week spread across the week. There are 16,384 basic package costs. About 100 people take a flight, about 334800 holiday-makers travel during the season and there are around 83,700 bookings, averaging parties of 4.

Okay, now its your turn.

Some hints: We advise you to do 1, 2 and 3 all together. Sometimes working out what an attribute would functionally depend on is easier than trying to do the EER diagram, and then you can 'reverse engineer' the EER schema.

Often when you have done your EER schema and you do the functional dependencies you realise that you have missed something in your diagram.

The trick is to use both the top-down and bottom-up techniques together to get the best solution. It doesn't much matter which order you do things in.

1. Extend the EER schema for the Sweeney Tours database.

   The EER model is a conceptual model of the data and relationships so try to put as much semantic information in as possible. Do not be afraid to invent codes and ids as keys—beware, dates as keys are notoriously difficult to manage.
2. Derive a relational schema from the EER, and make any refinements or compromises you feel necessary. Clearly identify all candidate, primary and foreign keys.
3. List all the functional dependencies for the database schema. Demonstrate that each relation is in Boyce Codd Normal Form (BCNF).
4. Identify TWO new explicit integrity constraints for the EER schema (not domain constraints).

Report

The pair should hand in a joint report containing the following, taking care to first **PHOTOCOPY** it for the next exercise. The marks awarded for the report will be given to both students.

- The extended EER schema for the Sweeney Tours database. (8 marks)

- The derived relational schema from the EER. Discussion of the derivation and refinements in full. Clearly identified all candidate, primary and foreign keys. (4 marks)

- The functional dependencies for the database schema. A demonstration that each relation is in Boyce Codd Normal Form (BCNF). (4 marks)

- Explicit integrity constraint identified for the EER schema. (2 marks)

Two marks will be awarded for good report style.

Exercise Four (2 sessions)

Marks: 20
Equipment required: Windows NT PCs using the Oracle DBMS. It may also be possible to use Linux PCs, but this is not recommended as paths and script files (e.g., to repair damaged databases) may not work on Linux machines.
To be done in PAIRS.

Purpose

1. To map the conceptual EER data model to a relational model and explore the consequences of the integrity constraints.
2. To implement and populate the database for the Sweeney Tours tour operator system.
3. To gain an insight into the cost of data take-on in the database development cycle.
4. To encourage the critical assessment of the database design.
5. To check that the design will satisfy a typical query.
6. To understand permissions in Oracle 8.

What you need to know to do this exercise

You will need to understand SQL CREATE and INSERT, granting permissions in SQL and creating synonyms.You will also need to be good at editing, and good at organising your pair efficiently.

Problem

This exercise falls into two parts:
1. creating and populating your relational database, and executing a query over it;
2. evaluating your database design in the light of using it and building it.

To reflect these two parts the exercise has two deliverables that are separately recorded on the mark sheets.

Part 1. Creating, populating and querying your database

Create your relational database

Using the CREATE TABLE command. Make sure that you define date attributes with the datatype DATE, so that later you can do date comparisons. **Note that you do not need to create the `BOOKING` table as it will not be populated with any data**.

Creating and populating your tables in the right order

When you use referential integrity definitions in your create table commands, make sure that you create your tables in the right order. You wouldn't be allowed to create a table A that references another table B unless you have already created B. Sometimes this can get tricky, so you may have to add a constraint onto the table later.

You must also take care when you are populating the database. If you enter a tuple A that has a referential integrity constraint to another tuple B, and that tuple B doesn't exist yet, you will not be allowed to enter tuple A. So you must organise the order that you create the data in very carefully. If you end up without anyway of starting because everything is so interconnected, then you will have to temporarily disable a referential integrity constraint to get you going. **See section 2 of the notes *Using Integrity Constraints* in the *Oracle 8 Guide*.**

Granting permissions to your partner

The best approach is that one of you creates the tables and one fixes up the test data. Any table, view or synonym is initially only accessible by the user who created it. Oracle allows the owner of an object to grant privileges on this object to other users. The SQL command for this is GRANT.

```
Grant privilege, ...
     ON object
     TO user, ...;
```

The privileges that may be granted are:

ALTER	allows the grantee to change the table definition with the ALTER TABLE command.
DELETE	allows the grantee to remove rows from the table or view with the DELETE command.
INDEX	allows the grantee to create an index on the table with the CREATE INDEX command.
INSERT	allows the grantee to add new rows to the table or view with the INSERT command.
REFERENCES	allows the grantee to create a constraint that refers to the table.
SELECT	allows the grantee to query the table or view with the SELECT command
UPDATE	allows the grantee to change data in the table or view with the UPDATE command.
ALL PRIVILEGES	all operations

Say you are Fred Bloggs and you want to grant Joe Soap (soapj) the right to select and update your student table:

```
GRANT SELECT, UPDATE
     ON student
     TO soapj;
```

The command to reverse such assignation is REVOKE

```
REVOKE right, ...
     ON object
     FROM user, ...;
```

A privilege may only be withdrawn by the user who granted it or by the overall database administrator. There is a special user called PUBLIC which means all users.

Synonyms

So now you have the privilege to access but all is not yet complete. To access that users table or view, you must include their username as part of the table name. Eg:

```
SELECT studentno
     FROM bloggsf.student;
```

This is going to get really tedious. Instead you can associate a synonym, an additional name, to a table. The table still retains its original name.

```
CREATE SYNONYM synonym_name FOR table;
```
e.g.:
```
CREATE SYNONYM mystudent FOR bloggsf.student;
```

delete a synonym with the DROP command, e.g.:

```
DROP SYNONYM mystudent;
```

It would be a good idea for you both to settle on the same names for the tables or synonyms. Synonyms can also be given to views.

2. Populate your relational database schema.

The test data is available in electronic form as a series of SQL insert statements. A directory containing a **.sql** file for each relation that you will need to load should be copied to your file store. You will be given the directory once you have handed in Exercise 3.

We have tried to think of all the ways that you may have designed your relations and provided the appropriate collection of statements. You will need to look at the definitions of all the different possible insert files to find the ones suitable for your definitions. A file called README has a list of the files and their definitions. We doubt whether we will have thought of every possible way of designing the relations. You may well have to edit the statements to suit your relations and you will certainly need to alter the names of relations and attributes. Talk to a demonstrator if you have such a radical design that you just can’t edit what you have been given.

Using SQL Command files

We present this by way of example. The procedure is as follows:

1. Copy a file that matches (or nearly matches) your table definition to your file store.

2. Edit the file to match your table definition, including column and table names. So the file to load up the data into the `airport` table could be called `airport.sql`. If we list this file it looks like this:

```
INSERT INTO airport(apcode,apname,category)
     VALUES ('ali','alicante','hl') ;
INSERT INTO airport VALUES ('plm', 'palma', 'hl');
INSERT INTO airport VALUES ('gtk', 'gatwick', 'uk') ;
INSERT INTO airport VALUES ('brm', 'birmingham', 'uk') ;
INSERT INTO airport VALUES ('man', 'manchester', 'uk');
INSERT INTO airport VALUES ('ibz', 'ibiza', 'hl');
INSERT INTO airport VALUES ('fvt', 'fuerteventura', 'hl');
INSERT INTO airport VALUES ('fro', 'faro', 'hl');
INSERT INTO airport VALUES ('mon', 'monastir', 'hl');
```

```
        INSERT INTO airport VALUES ('rho', 'rhodes', 'hl');
        INSERT INTO airport VALUES ('hra', 'heraklion', 'hl');
```

3. To run the SQL command file (and load the data into your database) use the SQL*Plus `START` command, e.g.:

```
        SQL> START file_name
```

where file_name is the name of your SQL command file. You need to type in the full path (e.g., `C:\…`); you do not need to type in the `.sql` file extension.

Common mistakes

- The data is of the wrong type
- The tuple does not contain enough attributes
- A foreign key defined in the table cannot find a primary key in another table being referenced. An attribute in the current tuple being loaded does not have a value that matches the primary key in the parent table. I.e. you are populating your tables in the wrong order.
- The primary key is violated. i.e. a key value occurs more than once in the load data.

3. Query your relational database schema.

Your database design should be a faithful representation of the mini-world of tour operators and holidays. Consequently you should be able to answer any *ad hoc* query that can be put to the database; some will be more difficult than others depending on your data model. Feel free to use views where helpful, but remember that views often make your queries run slower. In particular, your database must be capable of answering the following query (the SQL query and results will form part of your report).

Scenario

Freda Bloggs wishes to book a holiday in Majorca for 1 week, leaving some time between 29th July '91 and 11th August '91. Four people will be going requiring two rooms. Freda doesn't care which UK airport she flies from. They would all like to stay in the same hotel.

Question: What is the availability of a suitable package?

Select the resort, hotel, airport, flight, UK departure date and time of flight, number of seats available for the package, number of rooms available and the cost per person.

You shouldn't include those hotels and flights that don't have enough rooms or seats (outgoing or returning).

Part 2. Evaluating your database

4. Designing for Change

If Sweeney Tours introduced 14 day holidays and the Tamarell Hotel introduced single rooms, how would these changes affect your database schema?

5. Update Transaction

Freda Bloggs wishes to book one of the flights and an appropriate number of rooms to accommodate her party. Which relations and attributes would you need to update (particularly to ensure the integrity of your database schema).

6. Views

Suggest and justify one view that in your opinion would be of great benefit to the operator of the Sweeney Tours database system.

7. Hindsight

Having designed and built your database are there any differences between your proposed relational schema of exercise 3 and your final implementation? What are the strengths and weaknesses of your design? Is there any redundant data in your relations because of your design? What lessons have you learnt and what compromises have you made?

Report

The pair should JOINTLY hand in the following:

Part A deliverable:

- For each table, the database schema definition immediately followed by the data for that relation (4 marks)
- A listing of the SQL and results for the query (5 marks)

Part B deliverable:

- A discussion of how the schema would alter, or not, to incorporate the changes. (4 marks)
- A discussion of the update consequences of making a booking (2 marks)
- The SQL definition of the view and its justification. (1 mark)
- A thorough discussion of any differences between your originally proposed relational schema and your final implementation and a critique of your database design. (2 marks)

One mark will be given for good report style.