LAB 1 – DB Design Exercise

DT228/4 – Advanced Databases, Dr. Pierpaolo Dondio

In this lab we review DB modelling: ER diagram design, relational model and physical implementation of the model using Oracle.

Exercise 1 (short question on DB modelling)

A database contains a list of toys, all of them identified by a name, a price, a unique ID. However, toys are different and each toy could have specific attributes that only that toys has. For instance, the toy *car* has 2 attributes specific to the *car* only: *engine_size, petrol_or_diesel*. The toy *teddy* has two other specific attributes *material* and *age* (and, of course, *teddy* does not have the *engine_size* and *petrol_or_diesel* attributes, and the toy *car* does not have the *material* and *age* attributes) and so on…

How would you store this information in a database?

Explain and justify your solution. Consider all the potential aspects of the solution (easy to use and maintain, performance, storage efficiency, what if some new attributes are added?). You might also search the web for the problem

Exercise 2 (complete exercise about ER diagram, Relational model and implementation)

A travel agency has a DB for storing holiday offers and sales.

The DB contains information about customers (described by name, surname, address, an ID generated at registration time and the kind of subscription – premium or standard) and all the holiday packages available.

The Travel agency DB has a list of flights and hotels. A flight is a single way flight from city A to city B with a flight code, an airline name and a departure and arrival time.

A hotel is identified by a name, its city and the category (number of stars, from 1 to 5).

A holiday package has a total price (a derived field) and a starting and ending date. It includes one or many hotels with the number of nights in each hotel and the price per night in each hotel. It also includes the required flights, the number of seats booked on each flight and the price of each flight (for all the tickets). There could be some flights or hotels that are not yet into a holiday package, but a holiday package must have at least one hotel and two flights (go & back).

For instance, a travel package could be 2 nights at Hotel Excelsior in Rome (100 euro per night), 3 nights at Hotel Etoille in Paris (120 euro per night), plus flight from Dublin to Rome (90 euros), flight from Rome to Paris (150 euros) and Flight from Paris to Dublin (100 euros). The total price is 900 euro and the period and the holiday is from 1 Jan 2013 to 6 Jan 2013.

The DB contains also the holidays purchased by each customer, the date the package was booked and if the payment has been done.

Each customer has the option to unsubscribe from the system, but only if he has already paid all its holidays packages. In that case, all the information related to the customers (customer details and

packages purchased) must be deleted from the systems. A customer cannot register without buying its first holiday package (i.e. customer has at least a holiday package).

1. Create (on paper) an ER Diagram, identifying relationships, cardinalities, type of entities (weak, strong).
2. Provide a logical design (relational models) showing tables, primary and foreign keys, not null/null values
3. Implement the DB using Oracle SQL Developer. Implement the following constraints:
    a. An hotel category is a integer number from 1 to 5
    b. The total price of an holiday package is less than 100,000 euros (and more than zero)
    c. The kind of customer (premium or standard) is implemented using a char variable with only two allowed values: ‘p’ or ‘s’
4. Write a query to show all the 3-star hotels where a specific customer stayed.

Useful SQL Commands for Oracle

1. Create table with a primary key

```
CREATE TABLE NewTable1(
 FieldA number(2) PRIMARY KEY,
 FieldB varchar2(40) NULL ,
 FieldC number(2) NULL);
```

2. Create table with a composite primary key (fieldA + fieldB)

```
CREATE TABLE NewTable2(
 Field2A number(2) PRIMARY KEY,
 Field2B varchar2(40) NULL ,
 Field2C number(2) NULL,
CONSTRAINT ConstraintName PRIMARY KEY (FieldA, FieldB));
```

3. Create a new table3 linked to table1 with a foreign key (not null in the example)

```
CREATE TABLE NewTable3(
 Field3A number(2) PRIMARY KEY,
 Field3B varchar2(40) NULL ,
 Field3C number(2) NULL,
 FieldFKA number(2)NOT NULL REFERENCES NewTable1);
```

4. Create a new table4 with constraints on some fields. In the example, Field4C must be less than 2000 and Field4B must be either 'Y' or 'N'

```
CREATE TABLE NewTable4(
 Field4A number(2) PRIMARY KEY,
 Field4B varchar2(1) NOT NULL ,
 Field4C number(2) NULL,
 CONSTRAINT check4C CHECK (Field4C < 2000),
 CONSTRAINT check4B CHECK (Field4B IN ('Y','N')),
);
```

5. Weak entity foreign key

```
CREATE TABLE NewTable3(
 Field3A number(2) PRIMARY KEY,
 Field3B varchar2(40) NULL ,
 Field3C number(2) NULL,
 FieldFKA number(2)NOT NULL REFERENCES NewTable1 on cascade delete);
```

6. Table created if there is a many-to-many relationship between tableA and TableB.

```
CREATE TABLE many_to_many_a_b(
 ID_A integer not null, /* key of the first table */
 ID_B integer not null, /* key of the second table */
 CONSTRAINT Pkey PRIMARY KEY (ID_A,ID_B), /*composite primary key */
 CONSTRAINT FK1 FOREIGN KEY (ID_A) REFERENCES TableA, /*f.key to tableA */
 CONSTRAINT FK1 FOREIGN KEY (ID_A) REFERENCES TableB, /*f.key to tableB */
);
```