DD1334 Databasteknik

Laboration 1: SQL Basics

Andreas Gustafsson, Hedvig Kjellström, Michael Minock and John Folkesson

The purpose of Laboration 1 is to learn how to retrieve
rmation stored in relational databases. You will learn 1) how to formulate SQL queries and understand how they apply to the schema and how they are executed; 2) how constraints effect insertions and deletions; 3) about the basics of view, transactions, indices and triggers.

The recommended reading for Laboration 1 is that of Lectures 1-7 , particularly Chap 6-8.

Laboration ***Come prepared to the review session!*** *Only one try is allowed – if you fail, you are not allowed to try again until the next lab session. The review will take 10 minutes or less, so have all papers in order. To pass you should have*

Completed Task 1 with at least 9 of the 10 queries right, completed Task 2 and able to explain why certain actions give errors and other do not, also 8 of the 9 steps should be documented showing you executed them correctly, Task 3 the output file should show that the trigger works as it should.
The grade is A if passed when the review when due. See the Lab Grading page on the course web for the due dates for the labs and the grading of late assignments.

Laboration 1 is intended to take 30h to complete.

Computing Environment

In this assignment you will use Nestor 2.0. Nestor is KTH's logic engine (computer) dedicated to hold the databases used in this and other similar courses. Nestor is aptly named after Nestor the Gerenian horseman from Homer's Illiad. Nestor was, by the time of the Trojan war, an old man allegedly one hundred and ten years old. His advanced age had imparted great wisdom and knowledge and he often gave sage counsel to the kings of the Achaeans. Nestor was an Argonaut, one of the famous boatmen from Argos who explored Hellas and fought exotic creatures such as centaurs and sirens. Our Nestor is not that old, he was actually born (created) quite recently to succeed his predecessor who had held the office for almost twenty years. Nestor may not be wise, but he holds much knowledge in the form of databases, and just as Nestor of old did, we will depart on an exploratory journey into the world of databases.

An alternative to Nestor 2.0 is to use your own database system. We recommend that you use PostgreSQL, which can be downloaded to your own computer free of charge.

Logging In

To access your database on Nestor 2.0, you have to go through the remote host access of the database system PostgreSQL. To be able to log in to Nestor you must first find your password (which is not your CSC password), which is located in your private folder on your CSC account. To access your CSC account you must either use one of the computers at CSC or SSH to one of them.

All of the following steps assume you use Ubuntu with the default window manager.

SSH to a CSC Computer

Skip this step if you are using a CSC computer.

1. Open a termial window (Applications->Accessories->Terminal).
2. Type `ssh <username>@u-shell.csc.kth.se` where <username> is your CSC username.

Finding Your Password to Nestor 2.0

1. Open a terminal window (Applications->Accessories->Terminal) if you are on a CSC computer, or SSH to a CSC computer if you use your own computer.
2. Run the command `cat Private/.psqlpw-nestor2.csc.kth.se`.
3. Take notes of the output.

Logging in to Nestor 2.0

1. Run the command `psql -h nestor2.csc.kth.se -U <username>` where <username> is your CSC username.
2. Input the Nestor password (see above).
3. You are now connected to your own slice of database, called `<username>` where <username> is your CSC username. If not, ask an assistant.
4. When you are done, quit using the command `\q`.

Running SQL scritps from a terminal

Actually working logged in to nestor2 is not the only way. When you have written some queries or other SQL commands that you would like to execute you can use a text editor to create a file with them in it. To run a file, for example, named ‘queries.sql, you may (log out from nestor2 and) type from your u-shell directory:

psql –h nestor2.csc.kth.se < queries.sql –U <username>

To save the output in answers.txt:

psql –h nestor2.csc.kth.se < queries.sql > answers.txt

The files of course will reside on your u-shell.csc.kth.se account, not your laptop and not on nestor2.

Booktown[1] Database Structure

books((book_id), title, *author_id*, *subject_id*)
publishers((publisher_id), name, address)
authors((author_id), last_name, first_name)
stock((isbn), cost, retail_price, stock)
shipments((shipment_id), *customer_id*, *isbn*, ship_date)
customers((customer_id), last_name, first_name)
editions((isbn), *book_id*, edition, *publisher_id*, publication_date)
subjects((subject_id), subject, location)

(Attributes in parantheses are key attributes, attributes in *italics* are foreign keys.)

A read-only version of Booktown is already installed on Nestor 2.0, under the name `booktown`. To go from your own database to the read-only Booktown instance when logged in to Nestor 2.0, use the command `\c booktown`. To see information on the database you can type for example:

```
SELECT table_name FROM INFORMATION_SCHEMA.TABLES
WHERE table_schema='public';

SELECT table_name, column_name, is_nullable, data_type FROM
INFORMATION_SCHEMA.COLUMNS
WHERE table_schema='public';
```

Notice that you need to scroll through the output and when you get to (END) you can hit q to return to the command prompt. To return to your own database `<username>`, use the command `\c <username>`. You can execute SQL instructions from a file by the command `\i filename.sql`.

To be able to make changes to the database, you need a private, editable Booktown copy. Such a copy is created by first returning to your home database (see above) and then issuing the command `\i /info/DD1334/dbtek14/labbar/lab1/create.sql`. This can be repeated to return the tables to their original state. You should however undo explicitly any other modifications such as creating views, constraints, or triggers.

If you are using your own database system, Booktown can be created in the same manner using the SQL script create.sql, found in the course directory /info/DD1334/dbtek14/labbar/lab1/ in the CSC tree.

Task 1: SQL

Formulate SQL queries that correspond to the following questions. If you get it right, you should get the same answers as noted below. Note however that your queries are not necessarily correct just because the output matches, you should consider special cases. We suggest that you pose the queries to the read-only database `booktown` (see above), since you thereby avoid mistakes due to unintentional changes of the database.

Be prepared to suggest indices on certain attributes that would make the execution more efficient, and explain what happens to the efficiency if, for example, the number of tuples in the relation change.

If you are interested (ie. you don't have to) you can try to make the queries as efficient as possible, one could go deeper into how the DBMS perform such query optimization *(extra reading/browsing needed)*. You can use the commands `analyze` and `explain` to see how the queries are optimized internally.

Before the review: Create a SQL file that runs your queries. Then, go to the read-only database booktown (see above), pose all queries to the system and save the output, e.g. as a text file. Count % correct queries and make note of this.

An SQL file is an ordinary acii text file with the SQL code as in the book. Each statement needs a terminating ';'. This file can be created by for example emacs and has the .sql extension. See the previous section on running script files and saving results.

At the review: Show the SQL query file as well as the answers (with % correct noted) to the assistant. Both group members should be prepared to answer questions about the queries.

1. Who wrote "The Shining"?

   Answer: *King, Stephen*

2. Which titles are written by Paulette Bourgeois?

   Answer: *Franklin in the Dark*

3. Who bought books about "Horror"?

   Answer: *Morrill, Royce*
   Jackson, Annie
   Holloway, Adam
   Black, Jean
   King, Jenny
   Anderson, Jonathan
   Gould, Ed
   Becker, Owen
   Brown, Chuck

4. Which book has the largest stock?

Answer: *Dune*

5. How much money has Booktown collected for the books about Science Fiction? They collect the retail price of each book shiped.

   Answer: *137.80*

6. Which books have been sold to only two people?
   Note that some people buy more than one copy and some books appear as several editions.
   Answer: *Dune*
   Little Women
   The Velveteen Rabbit

   2001: A Space Odyssey

7. Which publisher has sold the most to Booktown?
   Note that all shipped books were sold at 'cost' to as well as all the books in the stock.

   Answer: *Ace Books, 4566.00*

8. How much money has Booktown earned (so far)? (Explain to the teacher how you reason about the incomes and costs of Booktown)

   Answer: *136.15 or -12789.85, depending on how the stock is treated*

9. Which customers have bought books about at least three different subjects?
   Answer: *Jackson, Annie*

10. Which subjects have not sold any books?
    Answer: *Mystery*
    Business
    Religion
    Cooking
    Poetry
    History
    Romance
    Entertainment
    Science

Task 2: View, insert, delete, update, constraint.

Now create your own private copy of Booktown as described above. It is very important that you do not continue to work with the public copy used in Task 1. It might help to study the create.sql and understand both the schema and the database contents. You can rerun this file to wipe your changes and start fresh if needed.

Again write your SQL code in a file and present it along with the output in another file.

1. Create a view that contains isbn and title of all the books in the database. Then query it to list all the titles and isbns. Then delete (drop) the new view. Why might this view be nice to have?
2. Try to insert into editions a new tuple ('5555', 12345, 1, 59, '2012-12-02'). Explain what happened.
3. Try to insert into editions a new tuple only setting its isbn='5555'. Explain what happened.
4. Try to first insert a book with (book_id, title) of (12345, 'How I Insert') then
   One into editions as in 2. Show that this worked by making an appropriate query of the database. Why do we not need an author or subject?
5. Update the new book by setting the subject to 'Mystery'.
6. Try to delete the new tuple from books. Explain what happens.
7. Delete both new tuples from step 4 and query the database to confirm.
8. Now insert a book with (book_id, title, subject_id) of (12345, 'How I Insert', 3443). Explain what happened.
9. Create a constraint, called 'hasSubject' that forces the subject_id to not be NULL and to match one in the subjects table. (HINT you might want to look at chap. 6.1.6 on testing NULL). Show that you can still insert an book with no author_id but not without a subject_id. Now remove the new constraint and any added books.

At the review: Show the SQL file as well as its output to the assistant. Both group members should be prepared to answer questions about the code and results.

Task 3: Triggers.

Continue working from your own private copy of Booktown as described above. It is very important that you do not work with the public copy used in Task 1.

Again write your SQL code in a file and present it along with the output in another file.

Your task here is to write a trigger that functions so that whenever a new shipment is recorded the stock is automatically decreased to reflect the shipment. If the current stock=0 then the insertion should not happen and an 'EXCEPTION' raised with some message such as 'There is no stock to ship'.

Notice that postgresql has a slightly different way than the book of 'handling' triggers with a user defined function (procedure) in the language plpgsql. You can find information on the web if you get stuck, but the following hints might be enough to show you how:

```
DROP FUNCTION decstock() CASCADE;

CREATE FUNCTION decstock() RETURNS trigger AS $pname$
      BEGIN
            … BLAH BLAH … SQL STATEMENTS;
            RETURN NEW;
      END;

$pname$ LANGUAGE plpgsql;

CREATE TRIGGER
      … (state the triggering condition as in described in
book 7.5.1)
      FOR EACH ROW
            EXECUTE PROCEDURE decstock();
```

Also notice that you can refer to the new tuple as NEW in the function as in 'NEW.isbn'. So you do not need the REFERENCING lines shown in the book . In the function above between BEGIN and END you can write most SQL statements and have them inside 'if' clauses with syntax for example:

```
IF something=SOME SQL QUERY
         THEN RAISE EXCEPTION 'message to show'
ELSE
         …
END IF;
```

Demonstrate that you trigger works by showing the stock, then inserting a shipment and then showing the change to the stock.

Have these two shipments in your .sql file after you create the trigger. :

SELECT * FROM stock;

Output: the whole table before making shipments.

```
INSERT INTO shipments
VALUES(2000, 860, '0394900014', '2012-12-07');
```

Since this book is not in stock, this should give:

Output: ERROR: There is no stock to ship

```
INSERT INTO shipments
VALUES(2001, 860, '044100590X', '2012-12-07');
```

This should give

Output: INSERT 0 1

Then test by:

```
SELECT * FROM shipments WHERE shipment_id > 1999;
```

Which should show that only the second shipment was actually inserted:

Output: 2001 | 860 | 044100590X | 2012-12-07 00:00:00+01

SELECT * FROM stock;

Output: the stock table with 044100590X decremented.

Remember to then restore the database to its original state (in the .sql file):

```
DELETE FROM shipments WHERE shipment_id > 1999;
```

Output: DELETE 1

```
UPDATE stock SET stock = 89 WHERE isbn = '044100590X';
```

Output: UPDATE 1

I would also end my sql file with

```
DROP FUNCTION decstock() CASCADE;
```

Which will remove the trigger and function from the database.