COMP9120 Relational Database Systems

Tutorial Week 12: Revision

Q1: SQL

Download the University schema from and use pgAdmin to connect to PostgreSQL server and run the SQL file to create all the tables and populate them with data. You might need to refresh your client after running the script to see the created tables (eg: right click Tables node and click on Refresh). Then try writing queries to answer the following questions based on this university schema:

a) Which lecturers (by id and name) have taught both 'INFO2120' and 'INFO3404'? Write a SQL query to answer this question using a SET operator.

Solution:

```
SELECT id, name
FROM AcademicStaff JOIN UoSOffering ON id=instructorId
WHERE uosCode = 'INFO2120'
INTERSECT
SELECT id, name
FROM AcademicStaff JOIN UoSOffering ON id=instructorId
WHERE uosCode = 'INFO3404';
```

b) Which lecturers (by id and name) have taught both 'INFO2120' and 'INFO3404'? Answer this using a sub-query without SET operators. Make sure your result doesn't include duplicates.

Solution:

```
SELECT DISTINCT id, name
FROM AcademicStaff JOIN UoSOffering ON id=instructorId
WHERE uosCode = 'INFO2120'
AND id IN ( SELECT instructorId
 FROM UoSOffering
 WHERE uosCode = 'INFO3404' );
```

Q2: Relational Algebra

Consider the following schema:

Book (isbn, title, publisher, publicationYear)
Author (aname, birthdate)
Publisher (pname, address)

Wrote (isbn, aname) // which author wrote which book.

For the same schema as above, use relational algebra to express the following queries:

a) Find all book titles published by Acme Publishers

$$\pi_{title}(\sigma_{publisher='Acme'}(Book))$$

b) Find all authors (just by name) of the book with ISBN 0444455551

$$\pi_{aname}(\sigma_{isbn=0444455551}(Wrote))$$

Q3: Serializability and Update Anomalies

Given the following relation

Offerings(uosCode, year, semester, lecturerId)

with this example instance:

uosCode	**Year**	**semester**	**lecturerId**
COMP5138	2012	S1	4711
INFO2120	2011	S2	4711

Consider the following hypothetical interleaved execution of two transactions T1 and T2 in a DBMS where concurrency control (such as locking) is not done; that is, each statement is executed as it is submitted, using the most up-to-date values of the database contents.

T1	**SELECT** * **FROM** Offerings **WHERE** lecturerId = 4711
T2	**SELECT** year **INTO** :yr **FROM** Offerings **WHERE** uosCode = 'COMP5138'
T1	**UPDATE** Offerings **SET** year=year+1 **WHERE** lecturerId = 4711 **AND** uosCode = 'COMP5138'
T2	**UPDATE** Offerings **SET** year=:yr+2 **WHERE** uosCode = 'COMP5138'
T1	**COMMIT**
T2	**COMMIT**

Indicate:

a) the values returned by the **SELECT** statements and the final value of the database;

T1: sees both tuples of initial table content: (COMP5138, 2012, S1, 4711) and (INFO2120,2011,S2,4711)

T2: sees the original year of COMP5138: 2012

Final database state: (COMP5138, 2014, S1, 4711), second tuple unmodified

b) whether the execution is conflict serializable or not.

non-conflict serializable: the end-effect of this is neither the same as T1 before T2 or T2 before T1

Q4: Calculating space and time

Suppose we have a table Rel1(A, B, C). Each A field occupies 4 bytes, each B field occupies 12 bytes, each C field occupies 8 bytes. Rel1 contains 100,000 records. There are 100 different values for A represented in the database, 1000 different values for B, and 50,000 different values for C. Rel1 is stored with a primary B+ tree index on the composite key consisting of the pair of attributes (A,B); assume this index has 2 levels, including the root page and excluding the leaf (record) pages. Assume that in this database, each page is 4K bytes, of which 250 bytes are taken for header information. Record locations within the index use a 4-byte *rowid*. Assume that reading a page into memory takes 150 msec, and that the time needed for any query can be approximated by the time spent doing disk I/O.

1a) Calculate the space needed for the records/data of Rel1.
Hint: How much space is used by a single record? How many records per page? How many pages for the table? Then convert this back to space. [Note that each record is not split across pages, and each page has a header.]

Solution:
Each page has 4096 – 250 = 3846 bytes available to store records. As each record needs 4+12+8 = 24 bytes, we can fit 3846/24 = 160 records in a page (note that we round down to an integer, as one doesn't store just part of a record in a page). Since Rel1 has 100,000 records, we need 100,000/160=625 pages to store the table; measured in bytes it is approximately 2.4 MB. (Note that our simplifying assumption here is 100% occupancy and you should always state your occupancy assumption).

1b) Calculate the time taken to perform a table scan (i.e., linear scan) through the table Rel1.
Hint: How many pages are needed to make up the space occupied by the table? How many pages will be read from disk during a table scan?

Solution:
To do a table scan we must fetch each of the 625 pages of the table; measured in seconds, this takes 625*150 = 93750 ms = 93.75 s or approximately 1.5 minutes.