Introduction to Databases

IT University of Copenhagen

Trial exam, Fall 2003

This exam consists of 6 problems with a total of 12 questions. The weight of each problem is stated. You have 4 hours to answer all 12 questions. If you cannot give a complete answer to a question, try to give a partial answer. You may choose to write your answer in Danish or English. Remember to write the page number, your name, and your CPR-number on each page of your written answer. The complete assignment consists of 5 numbered pages (including this page).

GUW refers to *Database Systems – The Complete Book* by Hector Garcia-Molina, Jeff Ullman, and Jennifer Widom, 2002

All written aids are allowed / Alle skriftlige hjælpemidler er tilladt.

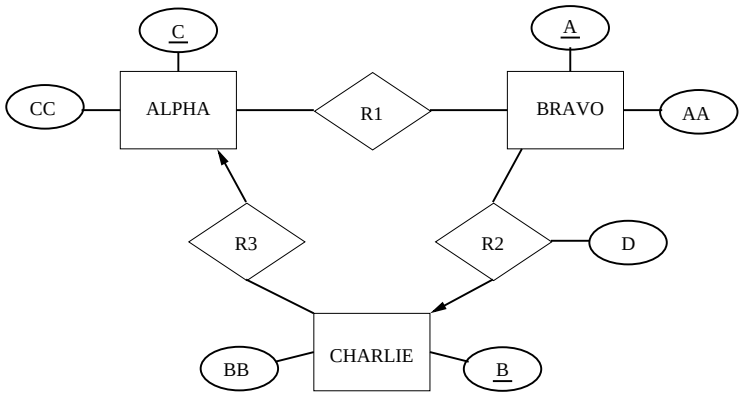
1 Database design (25%)

This problem has two unrelated parts. In the first part we consider the task of designing a database for a book club. The club has a selection of books for sale to its members. There is a "book of the month" every month, which is automatically shipped to all members who have not cancelled the shipment before a certain date. The database should contain information about:

- The selection of books, including information on titles, authors, publishers, and ISBN numbers (each book has a unique ISBN number).
- The members, including information on names, addresses, and payment due.
- The books selected as "book of the month".
- Which books have been shipped to each customer, and in what quantity.
- Which "books of the month" have been cancelled by each customer.

a) Draw an E/R diagram for the database. Emphasis should be put on using the design principles of GUW section 2.2. Remember to include appropriate keys and constraints.

In the second part we consider the following E/R diagram:



b) Perform a conversion of the E/R diagram into relation schemas, using the method described in GUW. You should combine relations when possible. Write SQL statements to create the relations, including key constraints. Assume that all attributes are integers.

2 Normalization (15%)

Consider the following instance of a relation R:

student	course	grade	address
William Gates	Operating systems	6	Microsoft Way 1
Jakob Nielsen	User interfaces	8	Silicon Valley 22
Mads Tofte	IT Management	10	Glentevej 67
Jakob Nielsen	User interfaces	8	Glentevej 38
William Gates	Intro. to databases	7	Microsoft Way 1
Steve Jobs	Intro. to databases	10	Apple Drive 10
Steve Jobs	Operating systems	10	Apple Drive 10

The functional and multivalued dependencies of a relation schema can be determined only from knowledge of the data that the relation is supposed to contain. However, from a given relation instance we may be able to say that a given FD or MVD does **not** hold.

a) Which of the following functional dependencies can be seen to **not** hold for R: `student` $\rightarrow$ `address`, `address` $\rightarrow$ `student`, `course` $\rightarrow$ `grade`.
Argue in each case why a functional dependency is possible or impossible.

b) Which of the following multivalued dependencies can be seen to **not** hold for R: `student` $\twoheadrightarrow$ `address`, `address` $\twoheadrightarrow$ `student`, `course` $\twoheadrightarrow$ `grade`.
Argue in each case why a multivalued dependency is possible or impossible.

3 SQL queries and relational algebra (20%)

Again consider the relation R from Problem 2. The following SQL query is used in the statistics office:

```
SELECT student, avg(grade) AS GPA
FROM R
HAVING count(*) > 1
GROUP BY student;
```

a) What is the result of the query when run on the instance of R from Problem 2? Describe in words what the query computes in general.

b) Rewrite the query to avoid the `HAVING` clause. That is, write an equivalent SQL query that does not contain the keyword `HAVING`. **Hint:** Create a new attribute to contain a count of tuples, and select those tuples where this attribute has value > 1.

c) Write an expression in relational algebra which is equivalent to the above query. **Hint:** Use your answer from b).

4 Authorization (15%)

The user `alice` has just created a relation `R(user,info)`, and issues the following SQL commands:

```
GRANT INSERT ON R TO bob WITH GRANT OPTION;
GRANT SELECT ON R TO bob WITH GRANT OPTION;
GRANT SELECT ON R TO claire;
```

Consider the following SQL commands:

1. `SELECT * FROM alice.R WHERE user='claire';`
2. `INSERT INTO alice.R VALUES ('claire','clairvoyant');`
3. `GRANT SELECT ON alice.R TO dorothy;`

a) State for each of the three users `bob`, `claire`, and `dorothy`, which of the above SQL commands he/she has autorization to execute.

Now assume that `bob` executes the command

```
GRANT INSERT ON R TO claire;
```

and `alice` then executes the command

```
REVOKE INSERT ON R FROM bob CASCADE;
```

b) Again, state for each of the three users `bob`, `claire`, and `dorothy`, which of the above SQL commands he/she has autorization to execute at this point.

5 Transactions (10%)

Consider the following three transactions on the relation `accounts(no,balance,type)`:

Transaction A

```
UPDATE accounts SET balance=balance*1.02 WHERE type='savings';
UPDATE accounts SET balance=balance*1.01 WHERE type='salary' AND balance<0;
UPDATE accounts SET balance=balance*1.07 WHERE type='salary' AND balance>0;
```

Transaction B

```
UPDATE accounts SET type='salary' WHERE no=12345;
```

Transaction C

```
UPDATE accounts SET balance=balance-1000 WHERE no=12345;
```

The purpose of Transaction A is to add interest to the balance of accounts, depending on the type and balance. Transaction B changes the type of a particular account. Transaction C makes a withdrawal from a particular account.

a) Suppose that the transactions are run more or less simultaneously at isolation level `READ COMMITTED`. What results of running the transactions are possible in this case, but not if the transactions had been run at isolation level `SERIALIZABLE`?

6 Database efficiency (15%)

You are appointed the administrator of a new DBMS that is used to register business transactions of a large multinational company, for use by the top management. Every day about 10,000 new business transactions are registered, and there are about 10 queries for old business transactions (identified by a transaction code). Having learnt about indexes on IDB, you consider placing an index on the transaction code to speed up queries. A full table scan processes 10,000 business transactions per second, while an index lookup can be done in 100 ms. The time used to insert is 40 ms without an index, and 100 ms with an index.

a) With 100,000 business transactions registered, what is the daily processing time (registering new + looking up old business transactions) with and without an index?

b) When will it become an advantage (in terms of total processing time) to use an index?