Introduction to Databases

IT University of Copenhagen

January 20, 2004

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%)

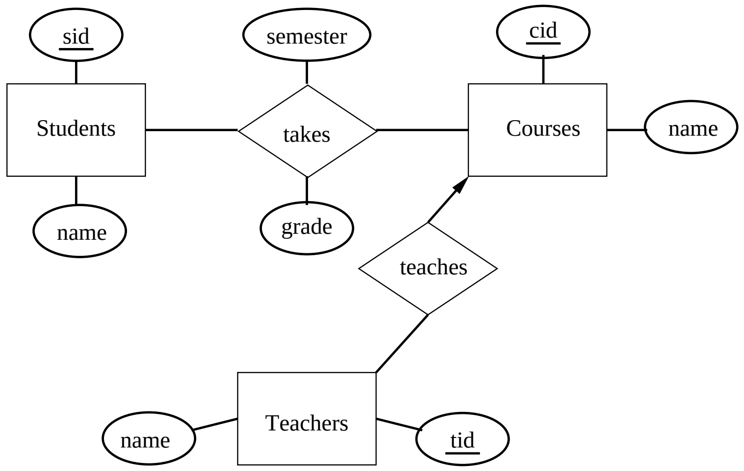
This problem has two unrelated parts. In the first part we consider the task of designing a database for characters in the *Lord of the Rings* books. The Tolkien trilogy contains characters belonging to many different people, e.g., hobbits, elves, dwarves, and men. The database should record:

- The name of each character, and the people he/she belongs to (you may assume that there is exactly one such people for each character).
- The places in the books (volume and page number) featuring this character. (The page number would refer to a specific edition.)
- For every pair of characters, the places in the book where these two characters meet.
- For each people, information on what place(s) it resides in (or has resided in), including the start (and possibly the end) of the period of residence.

For example, it should be recorded that the elves resided in Rivendell since the Second Age, and that Bilbo Baggins and Frodo Baggins are hobbits, and meet on page 3, say, of volume 1.

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

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. The attributes *sid, cid, tid*, and *grade* should be of type integer, and the remaining attributes text strings of length at most 30.

2 SQL queries and relational algebra (25%)

Consider the following relation schemas, used for examples in GUW:

```
Movie(title,year,length,inColor,studioName,producerC#)
StarsIn(movietitle,movieyear,starname)
MovieStar(name,address,gender,birthdate)
```

We assume (as in GUW) that the title and year uniquely identify a movie, and that the name uniquely identifies a movie star. Consider the following SQL queries, `Q1` and `Q2`:

```
Q1:  SELECT DISTINCT title, studioName
     FROM Movie, StarsIn
     WHERE starname='Meryl Streep' AND
           title=movietitle AND
           year=movieyear;

Q2:  SELECT DISTINCT title, studioName
     FROM Movie, StarsIn, (SELECT starname
                           FROM StarsIn
                           HAVING count(*)>10
                           GROUP BY starname) Productive
     WHERE title=movietitle AND
           year=movieyear AND
           Productive.starname=StarsIn.starname;
```

a) Give a description in words of what each of the queries computes. Emphasis should be put on giving a short and clear description.

b) Write a relational algebra expression (using extended operators if needed) corresponding to each of the queries. You may use the aggregation operator `COUNT(*)` to obtain a count of all tuples. **Hint:** First rewrite the subquery to avoid the `HAVING` clause.

The following SQL query computes the title and year of all color movies in the database from before 1939:

```
SELECT title, year
FROM Movie
WHERE year<1939 AND inColor=1;
```

c) Write a sequence of SQL statements that permanently remove from the database information on all color movies from before 1939, and all actors in the database starring *only* in such movies.

3 Indexing (10%)

Consider again query `Q1` of Problem 2.

a) Suggest an index which could speed up `Q1`, and write SQL (using the syntax presented in GUW) to create the index.

Suppose that `Q1` takes 100 ms with an index, and 1100 ms without an index (at that the size of the relations is not changing too much, so this number can be regarded as fixed). Also, assume that updating the index takes 100 ms per update (insertion or deletion).

b) How many updates must there be for each query before the time used for updating the index exceeds the time saved on queries.

4 Normalization (15%)

Consider the following instance of a relation R:

`saleID`	`salesman`	`regNo`	`make`	`office`
42	B. Honest	VY 34718	Opel	City
53	W. Gates	PQ 11112	Ford	Redwood
87	B. Honest	MX 32781	Ford	City
99	L. R. Harald	AB 12345	Porche	City

The functional dependencies of R, not including trivial ones, are:

1. `saleID` → `salesman regNo make office`
2. `salesman` → `office`
3. `regNo` → `make`

a) Decompose the relation into BCNF. For each step of the decomposition procedure, state what functional dependency it is based on, and give the relation schemas after the step has been carried out.

b) State the relation instances in your BCNF schema corresponding to the above instance of R. Give an example of an update anomaly of the original relation schema that has been eliminated in the BCNF schema.

5 Database constraints (15%)

We again consider the relation schemas from Problem 2. In this problem we suppose that the schema for `StarsIn` contains the declarations

```
FOREIGN KEY (movietitle,movieyear) REFERENCES Movie(title,year) ON DELETE CASCADE,
FOREIGN KEY (starname) REFERENCES MovieStar(name) ON DELETE SET NULL
```

and that corresponding `UNIQUE` constraints are set on `Movie` and `MovieStar`.

a) Explain what happens, if anything, to maintain the referential integrity constraints in each of the following cases:

1. A tuple in `Movie` is deleted.
2. A tuple in `MovieStar` is deleted.
3. A tuple in `StarsIn` is deleted.

Suppose we issue the SQL command:

```
INSERT INTO StarsIn VALUES ('Total Recall',1990,'Arnold Schwarzenegger');
```

b) Explain what is the result of the insertion command in each of the following cases:

1. The movie `Total Recall` does not exist in `Movie`.
2. The name `Arnold Schwarzenegger` does not exist in `MovieStar`.

6 Transactions (10%)

Consider the following three transactions on the relation `students(name,grade)`:

Transaction A
```
INSERT INTO students VALUES ('F. Student',5);
INSERT INTO students VALUES ('A. Student',13);
INSERT INTO students VALUES ('C. Student',8);
```

Transaction B
```
UPDATE students SET grade=grade+1 WHERE grade<11 AND grade>3;
```

Transaction C
```
UPDATE students SET grade=3 WHERE grade=5;
```

a) Suppose that the transactions run more or less simultaneously at isolation level `READ COMMITTED`, and that `students` is initially empty. List all 4 possible instances of `students` after all transactions have committed (assuming that no transaction is rolled back).