CS411

SQL/Mongo

Trigger:

```
create trigger [trigger_name]
[before | after]
{insert | update | delete}
on [table_name]
REFERENCING
NEW ROW AS NewTuple
[for each row]
[trigger_body]

INSERT INTO Drinks(name)
VALUES(NewTuple.drink);

// Use [NEW / OLD] and [ROW / TABLE] AS <name>.
// INSERT statements imply a new tuple (for row-level)
     or new set of tuples (for statement-tuple)
// DELETE implies an old tuple or set of tuples
// UPDATE implies both (the row before the update, and
     the row after the update)
```

Procedure:

```
CREATE PROCEDURE Result()
BEGIN
    DECLARE done int default 0;
    DECLARE cursor_container VARCHAR(10);
    DECLARE temp VARCHAR(255);
    DECLARE cur CURSOR FOR SELECT DISTINCT netid FROM
         Enrollments;
    DECLARE CONTINUE HANDLER FOR NOT found SET done =
         1;

    DROP TABLE IF EXISTS NewTable;

    CREATE TABLE NewTable(
        NetId VARCHAR(10),
        Course_Load_Status VARCHAR(255),
        PRIMARY KEY (NetId)
    );

    OPEN cur;
    REPEAT
        FETCH cur INTO cc;
        IF (query1) > 12 THEN SET temp = 'Maximum';
        ELSEIF (query2) = 12 THEN SET temp='Full';
        ELSE SET temp='Minimum';
        END IF;
        INSERT IGNORE INTO NewTable
        VALUES (cc, temp);
        UNTIL DONE
    END REPEAT;
    CLOSE cur;
    SELECT  FROM NewTable;END
```

Mep Reduce:

```
let mapper = function() {
    for(i=0; i < this.actors.length; i++) {
        emit(this.actors[i], this.ratings);
    }
};

let reducer = function(stack_id, values_arr) {
    sum = 0;
    for (i = 0; i < values_arr.length; i++) {
        sum += values_arr[i];
    }
    if (values_arr.length) {
        avg = sum/values_arr.length;
        return avg;
    } else {return 0;}
};

let mapReduceOutput = db.Movies.mapReduce(mapper,
     reducer, {out:"times", query: {release_year:{$lt:
     2000}}});
let updateRes = db.times.update({}, {$rename: {"value":
     "avg_ratings"}}, false, true);

db.times.find();
```

Sample Mongo Query

```
db.Movies.aggregate([
    {$match: {$and: [{ratings: {$gte: 9}}, {
         release_year: {$gt: 2000}}]}},
    {$unwind: "$actors"},
    {$group: {_id:"$actors", numMovies:{$sum: 1}}},
    {$sort : {numMovies : -1}},
    {$project: {_id:0, actorId:"$_id", numMovies:1}}
]);
```

ER Models

```
Rectangle means entity set
Diamond represents a relationship set between two
     entity sets
Oval is an attribute name.
A triangle with "isa" points from a subclass to a
     parent class
a → b means there is at most 1 b for an a, but the b
     can have 0 to infty as. For example, a is the
     employee and b is the department. (many-to-one)
a ←→ b means at most one entity is connected to at
     most one entity. For example, a is the employee
     and b is a laptop id. (one-to-one)
a --- b means there are no constraints. Could be
     anything! (many-to-many)
a --) b means that there is exactly one b for an a.
```

Converting Model to DDL

```
# NOTE: The primary key of table a is a_pk
Multiple attributes with the same name: check if all
     those names are duplicate. If so, you can add
     those attributes as 1 attribute to that entity.
     Otherwise create a different entity called attr-
     assignment that includes the primary key of the
     first entity and the attribute value. For example,
      Employee(SSN, E-Name, Office, Office) becomes
     Employee(SSN, E-Name) and Office-Assignment(SSN,
     Office), and make SSN the foreign key.
Many-to-Many relationship. Suppose there is a many-to-
     many relationship from a to b called c. Then
     create a new table called c where c_pk = a_pk,
     b_pk where a_pk and b_pk are foreign keys
Many-to-One relationship. Suppose there is a many-to-
     one relationship from a to b called c. Here a has
     only one b associated with it. Then create a new
     table called c where c_pk = a_pk where a_pk and
     b_pk are foreign keys. You can also simply add b_k
      to a as an attribute. If c has attributes, do it
     the former way.
Weak Entity Set. Suppose you have a weak entity set a
     and an entity set b that is connected. Now, when
     you create b, simply add b_pk as part of the
     primary key of a. Then follow the many-to-X rules.
     Then add FOREIGN KEY(b_pk) REFERENCES b, ON DELETE
      CASCADE
Subclasses: Suppose b and c are subclasses of a. Simply
      create a table for a-c and a-b.
```

Normal Forms

```
# BCNF (lossless)
1. Find every FD (use the process of finding the full
     set of FDs)
2. Check that A of every non-trivial FD in the form A
     →B is a superkey, meaning it uniquely identifies
     every attribute in the table. If so, you are done!
3. If not, compute A+.
4. Decompose it by chopping off everything that the
     left side does not uniquely determine and put it
     in a separate table with the left side attributes.
      This new table has no FDs. So you have table R1 =
      A+, R2 = A union (R - A+).
5. For each table, repeat from 1 to 4.
# 3NF (lossless and dependency preserving)
1. Find the key
2. Get a minimal basis G of given FDs
3. For each FD A→B in the minimal G, use AB as the
     schema of a new relation
4. If none of the schemas from step 2 is a superkey,add
      another relation whose schema is a key for the
     original relation.
```