Advanced Database
Spring 2015
Indexing

Submission Format: Online on D2L Dropbox as a PDF document

The tasks for this deliverable are described below. All work must be done in your database.

1. Copy Database

Download the file, createTablesWOIdx.sql (Create Populated Tables File link), from D2L. In this file is the SQL necessary to create a database complete with tables populated with data. Using SQL Developer, connect to your database. From the SQL Developer menu, select File -> Open and navigate to where you downloaded createTablesWOIdx.sql and open that file. You should see the SQL code in your SQL editor. Click the Run Script button. After this has run through completion, view the log. The only errors you should see in the log are DROP TABLE failures. This can be caused by a couple of things. It is possible you didn't have the table to start with. You also could have created the table and misspelled the table name when you created it. Do not be alarmed if you get these failure messages. You should see where the 10 tables were successfully created as well as the tables altered after adding primary keys and other constraints. At the end of this document is a table diagram. You may find it useful to explore the database before you start. Look at the data types of the columns. Look at some of the rows. You will be able to see the different domains for the columns.

2. Verify Data

Explore the new database and record the number of records for each table.

Order	Table	Records
1	STUDENT	10,369
2	SCHOOL	7,160
3	GUARDIAN	10,514
4	EXAM	3
5	VOLUNTEER_WORK	43,331

Order	Table	Records
6	ACTIVITY	77,726
7	SCHOOL_APPLY	39,488
8	SCHOOL_ATTEND	9,568
9	STUDENT_GUARDIAN	13,600
10	STUDENT_EXAM	23,329

3. Tune the Queries

Attachment 1, table diagram, will help you complete these tasks. The goal of tasks 1 – 6 is for you to tune the database using indexes to achieve the best performance from the DB in order to reduce the cost of running the queries to the lowest cost you can achieve. You should try ONE thing at a time. Otherwise, you won't know the true savings of your attempt. List what you did and the total cost for running the query using the tuning mechanism you selected. For example, if you build an index on a field in the table, list that you created an index, type of index, as well as the table and column and list the total cost. Not every attempt will reduce the cost but still list what you tried even if there was not a cost savings. You will need to implement multiple performance mechanisms, not just one. Ensure you start each task with the base line cost, meaning, you may have to delete indexes created on previous steps as not to falsely affect the baseline result.

Note: The script has three queries for creating indices (IDX_GUARDIAN_NAME, IDX_STUDENT_ENROLL_DATE and IDX_SCHOOL_NAME)

1. Enter the following query into SQL Developer.

```
SELECT COUNT(*), state
        FROM school
        GROUP BY state
        ORDER BY COUNT(*)
;
```

   Base Line Cost: 18
   After creating index for state (IDX_SCHOOL_STATE) the new cost: 6

2.

```
    SELECT count(*), city
    FROM guardian

    GROUP BY city

    ORDER BY city

  ;
```

Base line cost: 33

After creating index for city (IDX_GUARDIAN_CITY) the new cost: 10

3.

```
    SELECT last_name, description

        FROM student

        JOIN activity USING (student_id)

        WHERE gender = 'F' AND

            grade_level = '8' AND

            start_date <= '03-APR-06'

        ;
```

Base line cost: 180

After creating index for last_name (IDX_STUDENT_LAST_NAME) the new cost: 180 (no change)

After creating index for grade_level (IDX_STUDENT_GRADE_LEVEL) the new cost: 180 (no change)

After creating index for student_id (IDX_ACTIVITY_STUDENT_ID) the new cost: 180 (no change)

With three separate indices i.e.; IDX_STUDENT_GENDER (bitmap), IDX_STUDENT_GRADE_LEVEL (Non-Unique), IDX_STUDENT_START_DATE the new cost: 150

4.

```
SELECT  last_name || ', ' || first_name AS name, COUNT(*) AS num_accepted
    FROM student s
    JOIN school_apply sa ON (s.student_id = sa.student_id)
    WHERE decision = 'accepted' AND
        grade_level = 8
    GROUP BY s.last_name, s.first_name
    HAVING count(*) <= 2
    ORDER BY count(*) DESC, last_name, first_name;
```

Base line cost: 87

After creating index for student_id (IDX_SCHOOL_APPLY_STUDENT_ID) the new cost: 87 (no change)

After creating index for first_name, last_name (IDX_STUDENT_FIRST_LAST_NAME) the new cost: 87 (no change)

After creating index for school_id (IDX_SCHOOL_APPLY_SCHOOL_ID) the new cost: 87 (no change)

5.

```
SELECT last_name, first_name, name

    FROM student s JOIN school_attend a USING (student_id)

    JOIN school c USING (school_id)

    WHERE s.end_date >= '01-JAN-06'

    AND s.end_reason = 'Graduated'

    AND c.state = 'GA'

    AND type = 'C'

    AND a.end_date IS NULL

    ORDER BY name, last_name, first_name;
```

Baseline Cost: 82

After creating index for state (IDX_SCHOOL_STATE) the new cost: 82 (no change)

6. Create a Bitmap Index on the Student table on the Gender column. Execute a query that selects all of the female students and displays their last name and first name.
Does this query utilize your bitmap index?
Based on your answer above, explain why you think it does or doesn't.

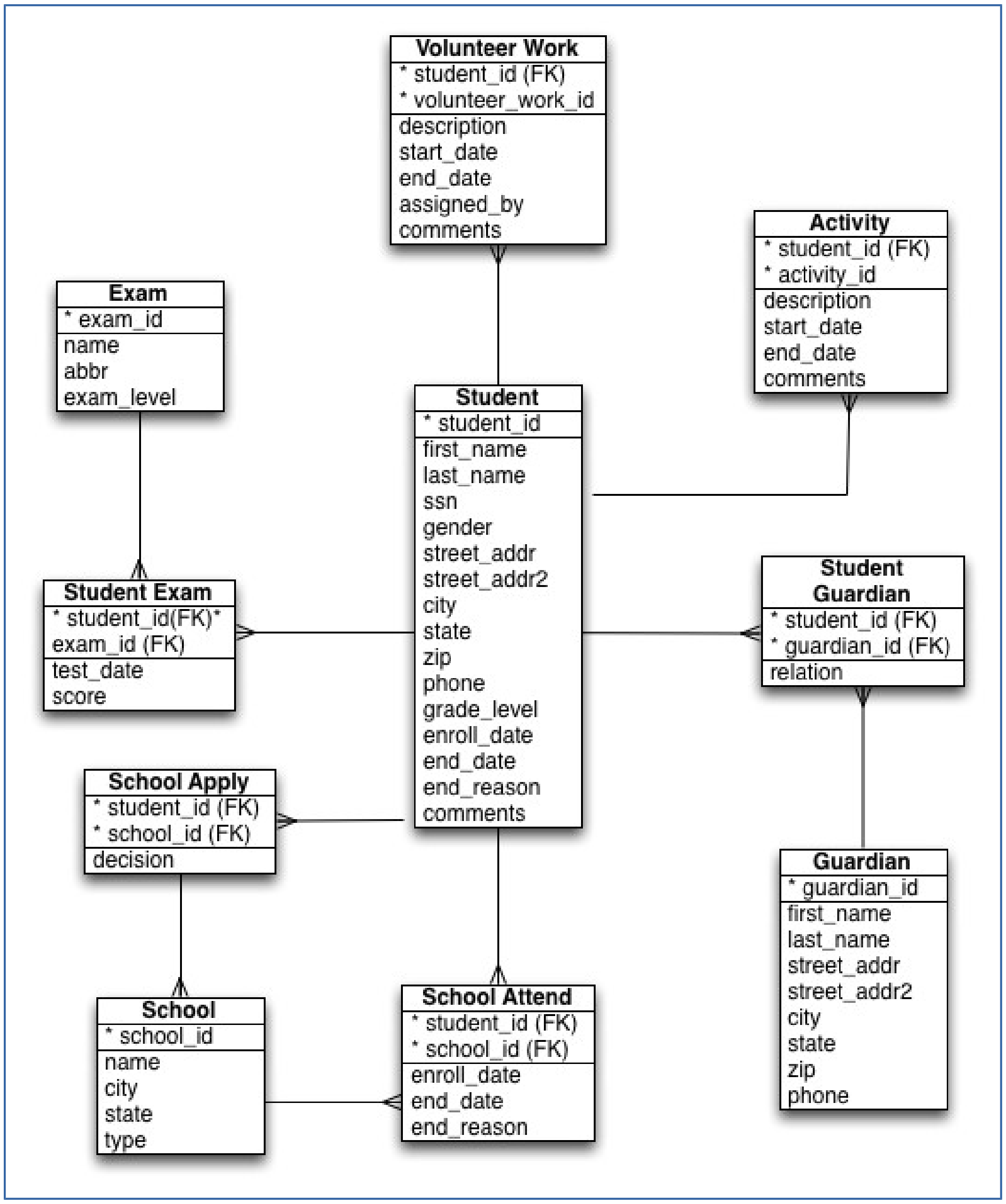
Query:

```
SELECT s.last_name, s.first_name
FROM student s
WHERE s.gender= 'F';
```

Baseline Cost: 48
Cost (After creating bitmap on gender) : 48

It did not utilize the bitmap index, probably because the full table scan was probably faster than using the bitmap.

7. Ensure you have submitted your assignment to D2L. Many times, you have loaded your assignment, however, you have not submitted and I, therefore, do not have access to it. Also, ensure the document you submit to D2L can be downloaded and opened. If for whatever reason, I either cannot download your document or cannot open it, that is the same as not submitting it and you will receive a zero for the assignment.


Volunteer Work
* student_id (FK)
* volunteer_work_id
description
start_date
end_date
assigned_by
comments
Activity
* student_id (FK)
* activity_id
description
start_date
end_date
comments
Exam
* exam_id
name
abbr
exam_level
Student
* student_id
first_name
last_name
ssn
gender
street_addr
street_addr2
city
state
zip
phone
grade_level
enroll_date
end_date
end_reason
comments
Student Exam
* student_id(FK)*
exam_id (FK)
test_date
score
Student Guardian
* student_id (FK)
* guardian_id (FK)
relation
School Apply
* student_id (FK)
* school_id (FK)
decision
Guardian
* guardian_id
first_name
last_name
street_addr
street_addr2
city
state
zip
phone
School
* school_id
name
city
state
type
School Attend
* student_id (FK)
* school_id (FK)
enroll_date
end_date
end_reason