

CS4710/6710 Database Systems (Fall 2017)
Instructor: Liyang Yu
Assignment #4: Index and SQL Lab, Due: October 24th, 2017

Solution

Name: Sandhya Ramadasan , Panther Id: #002351924

Consider the “index calculation” problem we have studied in the class (see lecture note), where we have a relation of 8,000,000 tuples. Again, make the following assumptions:

- the size of disk block (also called disk page): it is usually 4K or 8K bytes, to make our calculation easier, we will use **4000** bytes as a block size
 - the size of the search key value: say we index the relation by using an attribute that is part of the primary key, and it is **40** byte string
 - the size of physical page addresses: this is needed since every node in the tree will store search key values as well as references to disk pages, i.e., the address of the disk page being referenced. For ease of calculation, we use a **10**-byte page address
 - each swap takes about 10 milliseconds
1. For this relation, calculate how long it will take to find a particular record if we use dense index, as described on lecture note, page 13? [16 points]

Solution:

Given:

Size of disk block (disk Page) = 4000 bytes

Size of each Search Key value = 40 bytes

Size of physical page address = 10 bytes

Time per Swap = 10 milliseconds

Total tuples = 8,000,000 tuples

Calculation:

Size of each index = $40+10 = 50$ bytes

Number of Records per block(Index Entry per page) = $(4000)/50 = 80$ records

Total Number of blocks for 8000000 tuples = $8000000/80 = 100000$ blocks

To search inside the index, Number of Swaps = $\log_2(100000) = 17$ swaps

Using Dense index, Swaps needed = $17+1 = 18$ Swaps.

Time taken to search a particular record (Dense Index)= $18*10 = \mathbf{180 \text{ milliseconds}}$

=> 180 milliseconds

2. Consider again the situation in question 1, with the same dense index, but add one extra level of sparse index, as described by lecture note page 16-17. Now, how long it will take to find one particular record? [16 points]

Solution:

Given:

Size of disk block (disk Page) = 4000 bytes

Size of each Search Key value = 40 bytes

Size of physical page address = 10 bytes

Time per Swap = 10 milliseconds

Total tuples = 8,000,000 tuples

Calculation:

Size of each index = $40+10 = 50$ bytes

Number of Records per block(Index Entry per page) = $(4000)/50 = 80$ records

Total Number of blocks for 8000000 tuples = $8000000/80 = 100000$ blocks

Adding one Extra level of Sparse Index :

For 100000 blocks of dense Index= $100000/80 = 1250$

Number of Swaps = $\log_2 1250 = 11$ swaps

Total Number of Swaps = $11 + 1 + 1 = 13$ swaps

Time taken to search a particular record (with One extra level of Sparse Index)=
 $= 13*10 = \mathbf{130 \text{ milliseconds}}$

=> 130 milliseconds

The following are exercises you need to finish and *submit screen copies*:

1. create a table named PERSON (4 points)

This is very a simple table, which has the following attributes:

id: integer, also the primary key

name: varchar(20)

age: integer

Use the right SQL statements to do this, submit a screen copy showing the statements you have used and the executing results.

Solution

The screenshot shows the SQL Fiddle web application. At the top, the browser menu includes Chrome, File, Edit, View, History, Bookmarks, People, Window, Help, and a user profile for Sandhya. The address bar shows the URL sqlfiddle.com/#19/c6cafa. The main area has tabs for SQL Fiddle (selected), MySQL 5.6, View Sample Fiddle, Clear, and Text to DDL. A sidebar on the right is titled "Query Panel" with the sub-instruction: "Use this panel to try to solve the problem with other SQL statements (SELECTs, etc...). Results will be displayed below. Share your queries by copying and pasting the URL that is generated after each run." Below the tabs are buttons for Build Schema, Edit Fullscreen, Browser, and schema status indicators. The bottom navigation bar includes Run SQL, Edit Fullscreen, Format Code, and schema status indicators.

```

1 CREATE TABLE PERSON (
2   ID INTEGER PRIMARY KEY,
3   Name VARCHAR(20),
4   Age INTEGER
5 );
6
7
8

```



2. insert a few records into PERSON table (6 points)

Use correct SQL statements to insert the following person into the PERSON table:

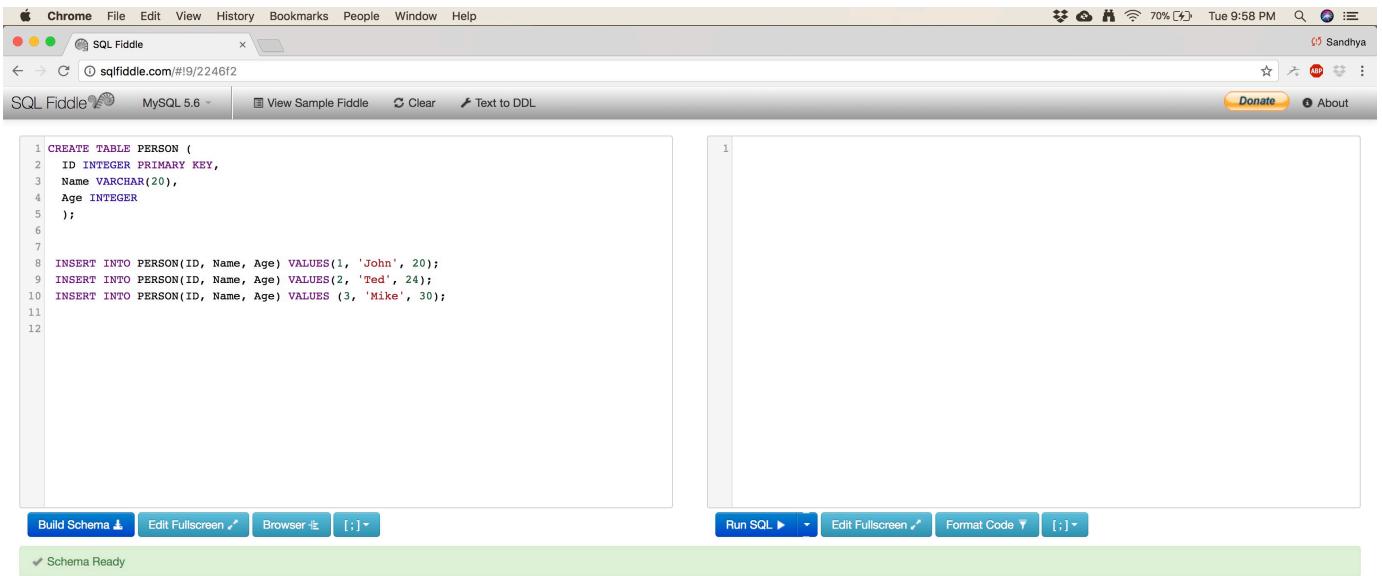
John, id is 1, age is 20

Ted, id is 2, age is 24

Mike, id is 3, age is 30

Submit a screen copy showing the statements you have used and the results you see.

Solution:



A screenshot of a Mac OS X desktop. At the top is the Dock with various application icons. In the center is a window titled "SQL Fiddle" from sqlfiddle.com. The window has a toolbar with "Build Schema", "Edit Fullscreen", "Browser", and "Schema Ready". The main area shows SQL code for creating a "PERSON" table and inserting three records. To the right is a results pane showing a single row of data.

```
1 CREATE TABLE PERSON (
2     ID INTEGER PRIMARY KEY,
3     Name VARCHAR(20),
4     Age INTEGER
5 );
6
7
8 INSERT INTO PERSON(ID, Name, Age) VALUES(1, 'John', 20);
9 INSERT INTO PERSON(ID, Name, Age) VALUES(2, 'Ted', 24);
10 INSERT INTO PERSON(ID, Name, Age) VALUES (3, 'Mike', 30);
11
12
```

3. use basic SELECT statement to show all the records in PERSON table (4 points)

Use the correct SQL statement to show all the records you have inserted into the PERSON table. Submit a screen copy showing what statement you have used and what is the result of the statement.

Solution:

The screenshot shows a Chrome browser window with the SQL Fiddle interface. The top navigation bar includes 'File', 'Edit', 'View', 'History', 'Bookmarks', 'People', 'Window', and 'Help'. The title bar says 'SQL Fiddle' and the address bar shows 'sqlfiddle.com/#/9/2246f2/1'. Below the header are tabs for 'MySQL 5.6', 'View Sample Fiddle', 'Clear', and 'Text to DDL'. On the right, there are links for 'Donate' and 'About'.

The left panel contains the DDL code:

```

1 CREATE TABLE PERSON (
2     ID INTEGER PRIMARY KEY,
3     Name VARCHAR(20),
4     Age INTEGER
5 );
6
7
8 INSERT INTO PERSON(ID, Name, Age) VALUES(1, 'John', 20);
9 INSERT INTO PERSON(ID, Name, Age) VALUES(2, 'Ted', 24);
10 INSERT INTO PERSON(ID, Name, Age) VALUES (3, 'Mike', 30);
11
12

```

The right panel shows the results of the query `SELECT * FROM PERSON;`:

```

1 SELECT * FROM PERSON;

```

Below the results are buttons for 'Run SQL', 'Edit Fullscreen', 'Format Code', and a copy icon. A table displays the data:

ID	Name	Age
1	John	20
2	Ted	24
3	Mike	30

At the bottom, a green bar indicates 'Record Count: 3; Execution Time: 13ms' and provides a 'View Execution Plan' link.

4. update one row. (2 points)

Use the correct SQL statement to update Mike's age, changing his age from 30 to 28. Submit a screen copy showing the statement you have used and the result of the update statement.

Solution:

A screenshot of a Mac desktop. At the top is a standard OS X menu bar with Apple, Chrome, File, Edit, View, History, Bookmarks, People, Window, Help, and a battery icon showing 71% power. Below the menu bar is a toolbar with a red, yellow, and green window icon, a 'SQL Fiddle' tab, a back/forward button, a search bar containing 'sqlfiddle.com/#19/d73413', and user information for 'Sandhya'. The main content area is a web browser window for 'SQL Fiddle MySQL 5.6'. It contains two panes: a left pane with SQL code and a right pane with a query result. The SQL code includes creating a 'PERSON' table, inserting three rows of data, and updating the 'Age' column for the row where 'Name' is 'Mike'. The right pane shows the result of the 'SELECT * FROM PERSON;' query, which returns three rows: ID 1 (Name: John, Age: 20), ID 2 (Name: Ted, Age: 24), and ID 3 (Name: Mike, Age: 30). Below the browser window is a Dock with icons for various Mac applications like Mail, Safari, and Finder. A green status bar at the bottom indicates 'Schema Ready'.

```
1 CREATE TABLE PERSON (
2   ID INTEGER PRIMARY KEY,
3   Name VARCHAR(20),
4   Age INTEGER
5 );
6
7
8 INSERT INTO PERSON(ID, Name, Age) VALUES(1, 'John', 20);
9 INSERT INTO PERSON(ID, Name, Age) VALUES(2, 'Ted', 24);
10 INSERT INTO PERSON(ID, Name, Age) VALUES (3, 'Mike', 30);
11
12 UPDATE PERSON
13 SET Age=28
14 WHERE Name='Mike';
```

```
1 SELECT * FROM PERSON;
```

A screenshot of a Mac desktop, identical to the one above but with a different timestamp in the top right corner: 'Tue 10:01 PM'. The browser window for 'SQL Fiddle MySQL 5.6' shows the same SQL code and results as the previous screenshot. Below the browser window is a table displaying the data from the 'PERSON' table. The table has columns 'ID', 'Name', and 'Age'. The data is as follows:

ID	Name	Age
1	John	20
2	Ted	24
3	Mike	28

At the bottom of the browser window, a green status bar shows 'Record Count: 3; Execution Time: 3ms' and links for 'View Execution Plan' and 'link'. A small note at the bottom of the page says 'Did this query solve the problem? If so, consider donating \$5 to help make sure SQL Fiddle will be here next time you need help with a database problem. Thanks!'.

```
1 CREATE TABLE PERSON (
2   ID INTEGER PRIMARY KEY,
3   Name VARCHAR(20),
4   Age INTEGER
5 );
6
7
8 INSERT INTO PERSON(ID, Name, Age) VALUES(1, 'John', 20);
9 INSERT INTO PERSON(ID, Name, Age) VALUES(2, 'Ted', 24);
10 INSERT INTO PERSON(ID, Name, Age) VALUES (3, 'Mike', 30);
11
12 UPDATE PERSON
13 SET Age=28
14 WHERE Name='Mike';
```

```
1 SELECT * FROM PERSON;
```

5. add another table (6 points)

Use the correct SQL statement to add another table, called EMAILS, which has the following attributes:

id: integer

email: varchar(50)

id + email: primary key, where id is a foreign key referencing the id attribute in PERSON table

Submit a screen copy showing the statements you have used and the result of executing these statements.

Solution:

The screenshot shows the SQL Fiddle interface in a Chrome browser window. The left panel contains the SQL code for creating the PERSON table and inserting three rows of data. The right panel shows the results of a SELECT query on the PERSON table. At the bottom, there is a toolbar with various icons and a status bar indicating 'Schema Ready'.

```
1 CREATE TABLE PERSON (
2     ID INTEGER PRIMARY KEY,
3     Name VARCHAR(20),
4     Age INTEGER
5 );
6
7
8 INSERT INTO PERSON(ID, Name, Age) VALUES(1, 'John', 20);
9 INSERT INTO PERSON(ID, Name, Age) VALUES(2, 'Ted', 24);
10 INSERT INTO PERSON(ID, Name, Age) VALUES (3, 'Mike', 30);
11
12 UPDATE PERSON
13 SET Age=28
14 WHERE Name='Mike';
15
16
17 CREATE TABLE EMAILS (
18     Id INTEGER,
19     Email VARCHAR(50),
20     CONSTRAINT EMAILS_pk PRIMARY KEY(Id,Email),
21     CONSTRAINT EMAILS_fk FOREIGN KEY(Id) REFERENCES PERSON(ID)
22 );
23
24
```

1 SELECT * FROM PERSON;

Run SQL ▶ Edit Fullscreen ▶ Format Code ▶ [:] ▶

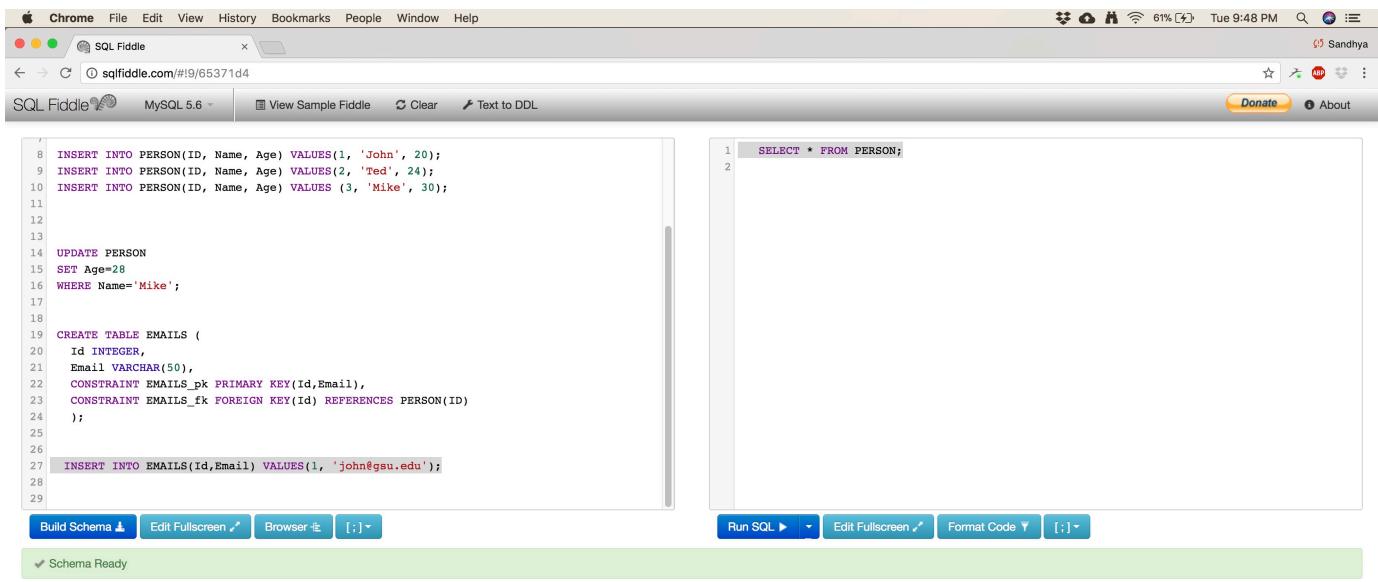
Build Schema ▾ Edit Fullscreen ▾ Browser ▾ [:] ▾

✓ Schema Ready



6. insert a record into EMAILS table (2 points)

Use the correct SQL statement to store John's email, john@gsu.edu. Submit a screen copy showing the statements you have used and the result of executing the statement.



The screenshot shows a Mac desktop with a Chrome browser window titled "SQL Fiddle". The browser window displays an SQL script and a results pane. The script includes several INSERT statements into a "PERSON" table and a CREATE TABLE statement for an "EMAILS" table. The results pane shows the output of a SELECT * FROM PERSON query. At the bottom of the browser window, there is a green bar with the text "Schema Ready". Below the browser window, the Mac OS X Dock is visible, showing various application icons.

```
8 INSERT INTO PERSON(ID, Name, Age) VALUES(1, 'John', 20);
9 INSERT INTO PERSON(ID, Name, Age) VALUES(2, 'Ted', 24);
10 INSERT INTO PERSON(ID, Name, Age) VALUES (3, 'Mike', 30);
11
12
13
14 UPDATE PERSON
15 SET Age=28
16 WHERE Name='Mike';
17
18
19 CREATE TABLE EMAILS (
20   Id INTEGER,
21   Email VARCHAR(50),
22   CONSTRAINT EMAILS_pk PRIMARY KEY(Id,Email),
23   CONSTRAINT EMAILS_fk FOREIGN KEY(Id) REFERENCES PERSON(ID)
24 );
25
26
27 INSERT INTO EMAILS(Id,Email) VALUES(1, 'john@gsu.edu');
28
29
```

```
1 SELECT * FROM PERSON;
2
```

Build Schema ▾ Edit Fullscreen ▾ Browser ▾ [:] ▾ Run SQL ▾ Edit Fullscreen ▾ Format Code ▾ [:] ▾

✓ Schema Ready

7. finally, use basic SELECT statement to show what is inside EMAIL table (4 points)

Use the correct SQL statement to show what is contained by EMAIL table, submit a screen copy showing the result.

Note: full score for this assignment is 64.

The screenshot shows the SQL Fiddle interface. On the left, there is a code editor with the following SQL script:

```
8 INSERT INTO PERSON(ID, Name, Age) VALUES(1, 'John', 20);
9 INSERT INTO PERSON(ID, Name, Age) VALUES(2, 'Ted', 24);
10 INSERT INTO PERSON(ID, Name, Age) VALUES (3, 'Mike', 30);
11
12
13
14 UPDATE PERSON
15 SET Age=28
16 WHERE Name='Mike';
17
18
19 CREATE TABLE EMAILS (
20   Id INTEGER,
21   Email VARCHAR(50),
22   CONSTRAINT EMAILS_pk PRIMARY KEY(Id,Email),
23   CONSTRAINT EMAILS_fk FOREIGN KEY(Id) REFERENCES PERSON(ID)
24 );
25
26
27 INSERT INTO EMAILS(Id,Email) VALUES(1, 'john@gsu.edu');
28
29
```

On the right, there is another code editor with the following SQL query:

```
1 SELECT * FROM PERSON;
2
3 SELECT * FROM EMAILS;
```

Below the code editors are several buttons: Build Schema, Edit Fullscreen, Browser, [;], Run SQL, Edit Fullscreen, Format Code, and [;].

Underneath the buttons, there is a table with one row of data:

Id	Email
1	john@gsu.edu

At the bottom, there is a green bar with the text "Record Count: 1; Execution Time: 4ms" and links for "View Execution Plan" and "link". A note at the bottom says "Did this query solve the problem? If so, consider donating \$5 to help make sure SQL Fiddle will be here next time you need help with a database problem. Thanks!"



The Mac OS X Dock at the bottom of the screen displays various application icons, including Finder, Mail, Safari, Google Chrome, Calendar, Reminders, Notes, Stocks, System Preferences, App Store, iTunes, iBooks, iPhoto, iMovie, iDVD, iCal, and iWork.