Assignment 2 Submission

1. Relational algebra

problem : Write the following queries in the relational algebra using relational schema:

student(id, name)
enrolledIn(id, code)
subject(code, lecturer)

Assuming : here we assumed that the enrolledIn.id join to student.id and enrolledIn.code join to subject.code.

1) What are the names of students enrolled in 1dv513?

   ∏student.name σenrolledIn.code = '2dv513' (student ⋈ student.id = enrolledIn.id (enrolledIn))

2) What are the names of students in both 1dv513 and 2dv513?

   ∏student.name σenrolledIn.code = '1dv513' (student ⋈ student.id = enrolledIn.id (enrolledIn)) ∩ ∏student.name σenrolledIn.code = '2dv513' (student ⋈ student.id = enrolledIn.id (enrolledIn))

3) Who teaches 2dv610?

   ∏lecturer σsubject.code = '2dv610' (subject)

4) Who teaches 1dv513 and 2dv513?

   ∏lecturer σsubject.code = '1dv513' (subject) ∩ ∏lecturer σsubject.code = '2dv513' (subject)

5) What are the names of students who are taking a subject not taught by Ilir?

   ∏student.name σsubject.lecturer ≠ 'Ilir' ((subject ⋈ subject.code = enrolledIn.code (enrolledIn)) ⋈ enrolledIn.id = student.id (student))

2. FDs and Normalization

problem: Several managers perform interviews with job applicants.

Data schema: Interviews (manager, applicant, day, time, room).

1) Find functional dependencies.

FD 1: *manager*, *day* -> *room*

(if we know the manager and the day, we can know which room the manager is in.)

FD 2: *manager*, *day, time* -> *applicant*

(Similarly, if we know the manager, date and time, we can know which applicant interviewed with the manager)

FD 3: *applicant*, *day* -> *manager*, *time*, *room*

(If we know the applicant and the day , we can know the manager, the time and room.)

FD 4: *day, time, room* -> *manager*, *applicant*

(If we know the day, time and room, we can know which manager and applicant are interviewing)

FD 5: *manager*, *applicant* -> *day, time, room*

(Conversely, once we know managers and applicants, then the date, time, and room they interviewed can be known.)

2) Find the keys of the relation.

Except FD5 the attribute day contains in left side equation, thus the superkey is: (manager, applicant, time, room) Because we can get this by checking the other FDs in which the LHS is a subset of the superkey, and that on its RHS contains some other attribute of the superkey. Then we can remove the attributes according condition that uniquely identifies a tuple.

In conclusion we could now be able to determine that the (*manager*, *day, time), (applicant*, *day) and (day, time, room)* is keys of relation.

3) Show that the relation is in 3NF but not in BCNF.

In a 3NF (third normal form) relationship, a non-primary key attribute must not transitively depend on the primary key.

FD1: Left-hand side of FD is not a super key, but right hand side is a prime attribute.
FD2: Left-hand side of FD is a super key.
FD3: Left-hand side of FD is a super key.
FD4: Left-hand side of FD is a super key.
FD5: Right hand side is a prime attribute.
Thus, Interviews relation should be in 3NF.

For all of FD1~ FD5, right hand side should be a super key and the interviews table is 3NF, this is the condition for BCNF table.
Thus, Interviews relation should be not in BCNF.

4) Decompose the relation in relations that are in BCNF.

To decompose the relation into XA & R – {A}, that is a relation in BCNF, we should find the FD that violates the condition. In our case FD1 and FD5 is the target. So, we would have:

- **R1:** (manager, day, room) -> a manager books one room for a day for some interviews
- **R2:** (manager, day, applicant, time) -> a manager schedules an interview with an applicant, at a specific day and time.

In the above relations, manager, day and room is key for R1 and (manager, time and day) and (applicant, day) would be relation key for R2 in BCNF.

5) Draw an E/R diagram that describes the system. Try to incorporate all dependencies.

I think that this system could be described to introduce three tables.

table 1 would store information about ***managers***.

table 2 is for ***applicants*** and table 3 is for ***interview*** information.

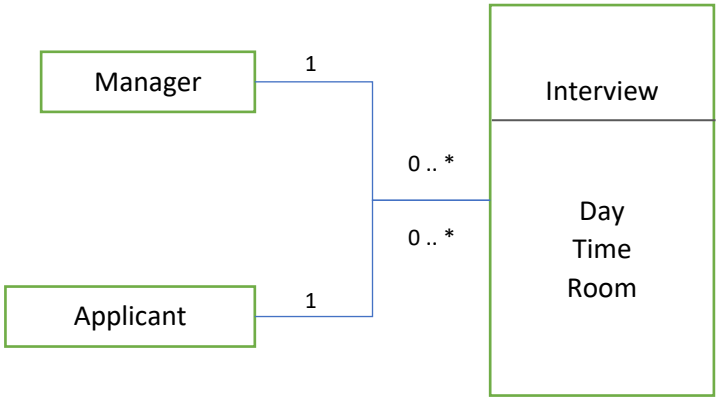

Figure 1: Task 2.5, E/R Diagram

3. Setting up the Reddit database

The database schema is shown as following:

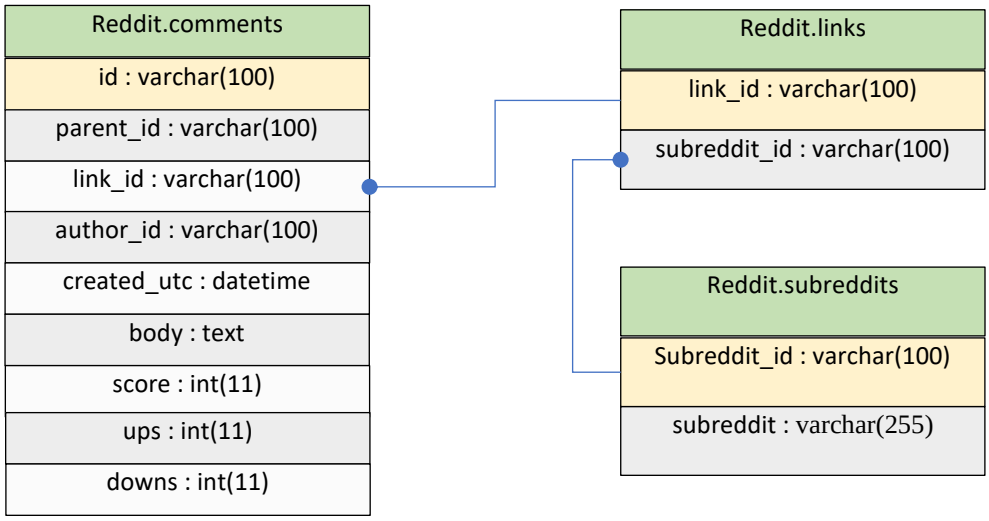

Figure 2: Reddit Database schema with constraints.

The schema of the Reddit database is made the database more understandable and readable and it could reduce data redundancy.

The database included most important attributes, not all attributes.

Table schemas with data types:

- The first table ***subreddits***, contains only two columns:

Field name	type	description
subreddit_id	varchar (100)	The id of the subreddit in base36
subreddit	varchar (255)	The name of the subreddit

- The ***links*** table contains its id and a reference to ***subreddits*** table:

Field name	Type	description
link_id	varchar (100)	The id of the link this comment is a reply to
subreddit_id	varchar (100)	a reference to ***subreddits*** table

- ***comments*** table schema:

Field name	Type	description
id	varchar (100)	The id of a comment, an integer value encoded using base36
parent_id	varchar (100)	The id of the thing this comment is a reply to
link_id	varchar (100)	The id of the link this comment is a reply to
author_id	varchar (100)	The posters name
created_utc	varchar (100)	When the comment was posted
body	text	The comment's contents
score	int (11)	The combination of up and down votes
ups	int (11)	The count of notes that up
downs	int (11)	The count of notes that down

4. Importing the data

I wrote and used a simple php script to import json data into the database.

Uploadeds the .bz2 file in the browser, decompress it, parse the json and insert it into the database.

For the 2007 data file, I inserted data into each table, processing every record one by one.
However, in the case of 2011 data file, as the number of records increased, the computer continued to process the file and append to the table did not work normally. To improve performance, the script has been modified to batch process records in groups of 1000 records.

All the batches ran correctly with no errors, and after the data insertion was completed, the elapsed time value was output.

1) Report measured times and discuss why you think the constraints affected the times.

The result of measuring the time to import records by performing batch insert processing is as follows:

	Record counts	**No Constraints Schema**	**With Constraints Schema**
2007 bz2 data file	subreddits : 32 links : 24370 comments :150429	46 s	62 s
2011 bz2 data file	subreddits : 8,293 links : 727701 comments : 727701	1578 s	1829 s

When dealing with small record sets, the difference was very small. However, as the count of records increases, the time difference between importing becomes larger.

Thus, we can find that importing from a constrained schema takes more time than an unconstrained schema.

The reason is that each time a row is inserted into a constrained schema, the number of checks to see if the inserted row violates the constraint increases.

When a NOT NULL constraint is added, other checks are also performed, such as when the value is not null.

This is a more important consideration as more and more additional time is required as more rows are added.

2) How do these constraints affect the import time?

I have tested for the smallest bz2 file (2007 file), the record count of ***links*** table is 24370.

For 24370 records, execution time cost measurement results is following.

case	batch	constraints	Execution time	Records count
1	no	yes	About 12 mins	24370
2	no	yes	Over than 15 mins	24370
3	yes	yes	2.2 seconds	24370
4	yes	no	1.2 seconds	24370

If I turn on the constraints (not null, unique, primary key for the ***id*** *field of the* ***links*** table), the execution time is 1.2 seconds and if turn of it, the time is 2.2 seconds.

i.e., without the constraints overall execution time cost is small than with the constraints.

I think that because I have inserted into the table only in one direction without to query such as search or checking duplicate, it seems to be unaffected by constraints.

If I join three tables and query with a select, the search speed is much faster when there is a constraint.

3) Would it be reasonable to import and turn on constraints after? When?

I don't think it is reasonable turn on the constraint after to import.

This is because error prevention and performance improvement functions, such as duplicate key prevention and indexing, work only when data is imported with the constraint turned on.

If the constraint is on, there may be time loss in data import, but for the stability of the database structure, the constraints must be set in advance in the table schema.

5. SQL queries

Assumption: In the following queries, ` param ` is the specified condition.

1) How many comments have a specific user posted?

```
SELECT count(*) AS usercommenttotal
FROM comments
WHERE author_id = 'param'
```

2) How many comments does a specific subreddit get per day?

```
SELECT date(created_utc) as createddaily, count(*) AS daycommenttotal
FROM comments
NATURAL JOIN links
WHERE links.subreddit_id = 'param'
GROUP BY date(created_utc)
```

3) How many comments include the word 'lol'?

```
SELECT count(*) as wordtotal
FROM comments
WHERE body LIKE CONCAT('%', 'lol', '%')
```

4) Users that commented on a specific link has also posted to which subreddits?

```
SELECT DISTINCT subreddit
FROM subreddits
NATURAL JOIN links
NATURAL JOIN comments
WHERE comments.author_id IN
(
    SELECT author_id
    FROM comments
    WHERE link_id = 'param'
)
```

5) Which users have the highest and lowest combined scores? (combined as the sum of all scores)

```
SELECT max_author.*
FROM (
  SELECT author_id as author,
  sum(score) AS totalScore
  FROM comments
  WHERE author_id <> '[deleted]'
  GROUP BY author_id
  ORDER BY sum(score) DESC LIMIT 1
) AS max_author
UNION
SELECT min_author.*
FROM (
  SELECT author_id,
  sum(score) AS totalScore
  FROM comments
  WHERE author_id <> '[deleted]'
  GROUP BY author_id
  ORDER BY sum(score) LIMIT 1
) AS min_author
```

6) Which subreddits have the highest and lowest scored comments?

```
SELECT max_subreddit.*
FROM (
  SELECT subreddits.subreddit,
  comments.score
  FROM comments
  NATURAL JOIN links
  NATURAL JOIN subreddits
  GROUP BY subreddits.subreddit
  ORDER BY score DESC LIMIT 1
) AS max_subreddit
UNION
SELECT min_subreddit.*
FROM (
  SELECT subreddits.subreddit,
  comments.score
  FROM comments
  NATURAL JOIN links
  NATURAL JOIN subreddits
  GROUP BY subreddits.subreddit
  ORDER BY score LIMIT 1
) AS min_subreddit
```

7) Given a specific user, list all the users he or she has potentially interacted with (i.e., everyone who has commented on a link that the specific user has commented on).

```
SELECT author_id
FROM comments
WHERE link_id IN (
  SELECT link_id
  FROM comments
  WHERE author_id = 'param'
)
AND author_id NOT IN ('[deleted]', 'param')
```

8) Which users have only posted to a single subreddit?

```
SELECT author_id
FROM comments
NATURAL JOIN links
WHERE author_id NOT IN ('[deleted]')
GROUP BY comments.author_id, links.subreddit_id
HAVING COUNT(*) = 1
```

Conclusion :

the query was first tested on a schema with the default constraint (PRIMARY UNIQUE) and then tested by adding an indexing constraint to the comments table.
Test results show that constraints specially improve performance when there are constraints in the database schema, especially when two or more tables are joined.
In the case of a small number of records, the presence or absence of a constraint was not significantly affected, but on a database with a lot of data, the constraint made the query executed much faster.
For example, in the case mentioned above, the results of measuring the time the second query that include join was executed are shown in the table below.

	Add index on link_id	Without index
2007 database	11.030 s	1.570 s
2011 database	38.175s	176.356 s

When link_id is joined, the time cost of the query execution has increased from several seconds to several minutes when there is no constraint than there is constraints.
The reason is that database indexes, by default, significantly improve performance by reducing the number of disk accesses performed when processing queries.
The index constraints added for the join field are as follows.

```
ALTER TABLE `reddit`.`comments` ADD INDEX (`link_id`), ADD FOREIGN KEY
  (`link_id`) REFERENCES `reddit`.`links`(`link_id`);
```