

# CSCI143 Final, Spring 2021

## **Collaboration policy:**

You may not:

1. discuss the exam with any human other than Mike; this includes:
  - (a) asking your friend for clarification about what a problem is asking
  - (b) asking your friend if they've completed the exam

You may:

1. take as much time as needed
2. use any written notes / electronic resources you would like
3. ask Mike to clarify questions via email

Name:

---

# 1 True/False Questions

For each question below, circle either True or False. Each correct answer will result in +1 point, each incorrect answer will result in -1 point, and each blank answer in 0 points.

1. True    False    A table that takes up 432KB on disk has 54 pages.
2. True    False    Postgres automatically compresses large TEXT values.
3. True    False    You should disable autovacuum to improve the performance of your database.
4. True    False    Decreasing the `fillfactor` for a table from the default value of 100 will make HOT tuple updates more likely.
5. True    False    A btree index created on an INTEGER column will have higher fanout than the same index created on a BIGINT column.
6. True    False    Postgres's btree indexes contain XID metainfo in order to determine tuple visibility for index only scans.
7. True    False    For very small tables, the postgres query planner is likely to choose a bitmap scan instead of a sequential scan.
8. True    False    A database stored using HDDs should have a higher value for the `random_page_cost` system parameter than a database stored using SSDs.
9. True    False    A denormalized representation of data tends to take up less disk space than a normalized representation.
10. True    False    The nested loop join strategy can be used to join tables on an equality constraint.
11. True    False    The hash join strategy can be used for self joins.
12. True    False    A hash index can be used to speed up a nested loop join.
13. True    False    A btree index can be used to speed up a CHECK constraint.
14. True    False    One advantage of the RUM index over the GIN index is that the former supports index scans and the latter does not. This implies that the RUM index can be used to speed up queries using the LIMIT clause, but the GIN index cannot.
15. True    False    If postgres crashes while a DELETE/INSERT/UPDATE statement is modifying a RUM index, the index becomes corrupted and must be regenerated from scratch.
16. True    False    The ANALYZE command collects statistics on the values in the table which the query planner uses when selecting which scan algorithm to use for a query.

- |     |      |       |  |
|-----|------|-------|--|
| 17. | True | False | Given the string “César Chávez”, an NFD-normalized UTF-8 encoding will require more bytes than a NFKC-normalized UTF-8 encoding.                         |
| 18. | True | False | The UTF-16 encoding cannot represent NFKD-normalized text.   |
| 19. | True | False | The TSVECTOR type can be created on NFC normalized TEXT fields, but cannot be created on NFD normalized TEXT fields.                                     |
| 20. | True | False | Given any string in NFKC form, normalizing to NFC and back to NFKC is guaranteed to be an idempotent operation (i.e. you will get the same string back.) |

## 2 Integrated Questions

The questions below relate to the following simplified normalized twitter schema. Each question (or sub-question) is worth 2 points, for a total of 20 points in this section.

```
CREATE TABLE users (  
    id_users BIGINT PRIMARY KEY,  
    created_at TIMESTAMPTZ,  
    username TEXT  
);  
  
CREATE TABLE tweets (  
    id_tweets BIGINT PRIMARY KEY,  
    id_users BIGINT REFERENCES users(id_users),  
    in_reply_to_user_id BIGINT REFERENCES users(id_users),  
    created_at TIMESTAMPTZ,  
    text TEXT  
);  
  
CREATE TABLE tweet_tags (  
    id_tweets BIGINT REFERENCES tweets(id_tweets),  
    tag TEXT,  
    PRIMARY KEY(id_tweets, tag)  
);
```

1. List all the tables/columns that have indexes created on them.

2. List the scan methods applicable for the following SQL query.

```
SELECT count(*) FROM tweet_tags WHERE id_tweets=:id_tweets;
```

3. Create index(es) so that the following query can use an index only scan.

Do not create any unneeded indexes; if no new indexes are needed, say so.

```
SELECT count(*)  
FROM users  
WHERE lower(username)=:username;
```

4. Create index(es) so that the following query can use an index only scan, avoid an explicit sort, and take advantage of the LIMIT clause for faster processing.

Do not create any unneeded indexes; if no new indexes are needed, say so.

```
SELECT id_users
FROM users
WHERE created_at <=:created_at
ORDER BY created_at DESC
LIMIT 10;
```

5. Construct index(es) so that the following query will run as efficiently as possible.

Do not create any unneeded indexes; if no new indexes are needed, say so.

```
SELECT id_users ,count(*)
FROM users
JOIN tweets USING (id_users)
JOIN tweet_tags ON (id_tweets)
WHERE tag = :tag
GROUP BY id_users;
```

6. Create index(es) so that the following query can use an index scan, avoid an explicit sort, and take advantage of the LIMIT clause for faster processing.

Do not create any unneeded indexes; if no new indexes are needed, say so.

```
SELECT id_tweets
FROM tweets
WHERE to_tsvector(text) @@ to_tsquery(:tsquery)
ORDER BY created_at <=> '2020-01-01'
LIMIT 10;
```

7. Consider the following SQL query.

```
SELECT *
FROM (
    SELECT
        id_tweets ,
        unnest(tsvector_to_array(to_tsvector(text))) as lexeme
    FROM tweets
) t
WHERE lexeme = :lexeme;
```

a) The query above cannot be sped up using an index. Why?

b) Rewrite the query from the previous question into an equivalent query that can be sped up using an index. Also provide the index that would speed up the query.



8. Consider the following SQL query that uses a cross join.

```
SELECT id_tweets
FROM users, tweets
WHERE to_tsvector(text) @@ to_tsquery(username)
      AND id_users = :id_users;
```

- a) Which join methods can be used to implement this query?
- b) If it is possible to construct an index that will speed up this query, do so. Otherwise, state that it is impossible and explain why.

### 3 Extra Credit

For each video listed below, you can earn up to 1 point of extra credit by: watching the complete video, and listing 5 facts from the video that you learned. These questions are entirely optional, and you will not be penalized should you choose not to complete them.

1. Scaling Instagram Infrastructure

<https://www.youtube.com/watch?v=hnpzNAPiCOE>

2. The Evolution of Reddit.com's Architecture

<https://www.youtube.com/watch?v=nUc07n4hek4>

3. PostgreSQL at 20TB and Beyond: Analytics at a Massive Scale (AdTech use of postgres)

<https://www.youtube.com/watch?v=BgcJnurVFag>

4. Large Databases, Lots of Servers, on Premises, in the Cloud - Get Them All! (AdTech use of postgres)

[https://www.youtube.com/watch?v=4GB7EDxGr\\_c](https://www.youtube.com/watch?v=4GB7EDxGr_c)

5. Breaking Postgres at Scale (how to configure postgres for scaling from 1GB up to many TB)

<https://www.youtube.com/watch?v=eZhSUXxfEu0>