Metis

RELATIONAL DATABASES

AND THE POWER OF SQL

I. INTRO TO DATABASES

II. RELATIONAL DATABASES

III. FUN WITH SQL

LEARNING GOALS

- What are databases?
- Why are databases needed?
- What are the differences between relational and non-relational databases?
  - When is one preferred to the other?
- How does one interact with relational databases?
- What is the purpose of SQL?

I. INTRO TO DATABASES

What are Databases?

Databases are a **structured** data source optimized for efficient **retrieval and storage**

structured : we will have to define some pre-defined organization strategy

retrieval : the ability to read data out

storage: the ability to write data and save it

Databases are a **structured** data source optimized for efficient **retrieval and persistent storage**

structured : we will have to define some pre-defined organization strategy

retrieval : the ability to read data our

storage: the ability to write data and save it

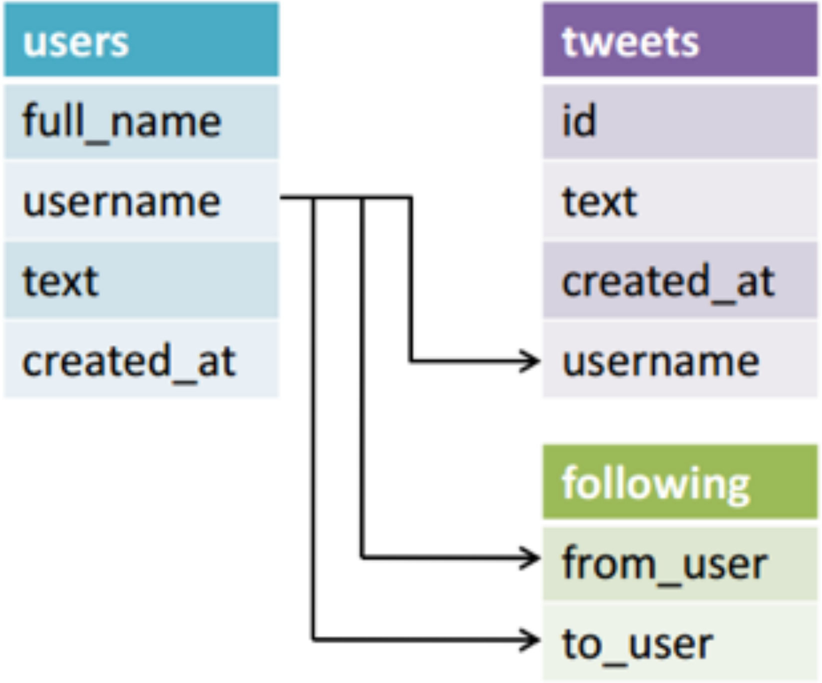
II. RELATIONAL DATABASES

Relational databases are traditionally organized in the following manner:

A database has **tables** *which represent individual entities or objects — "***Relations***"*

Tables have a predefined **schema** *- rules that tell it what columns exist and what they look like*

RELATIONAL DATABASES

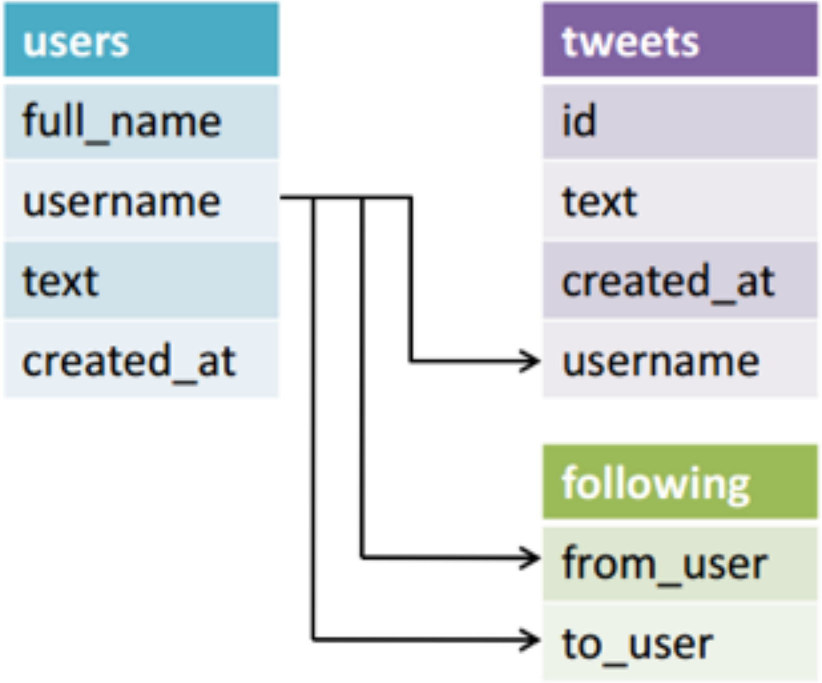


users
full_name
username
text
created_at

tweets
id
text
created_at
username

following
from_user
to_user

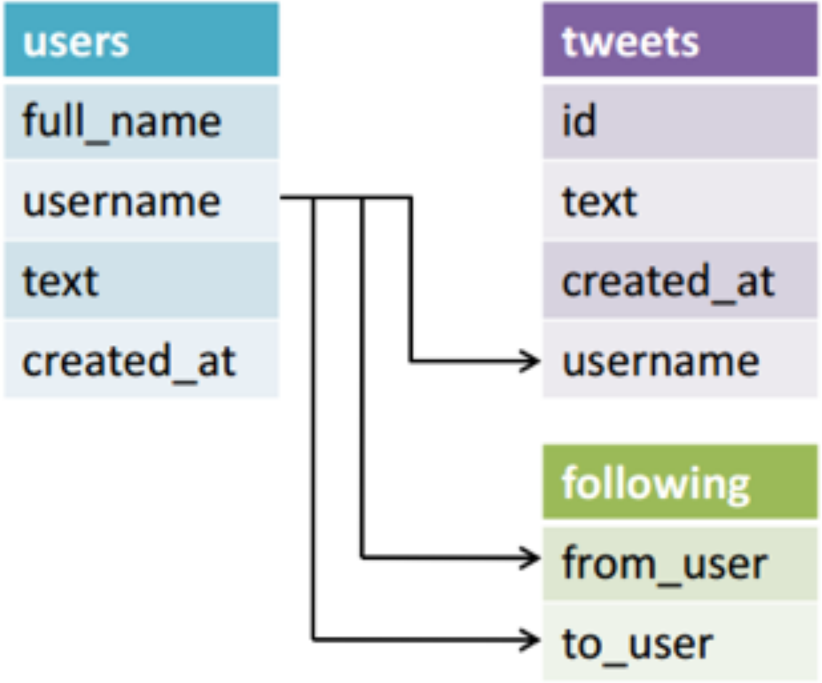
Each table should have a **primary key** column- a unique identifier for that row


users
full_name
username
text
created_at
tweets
id
text
created_at
username
following
from_user
to_user

Each table should have a **primary key** *column– a unique identifier for that row*

Additionally each table can have a **foreign key** *column- an id that references a unique entry in another table*

RELATIONAL DATABASES



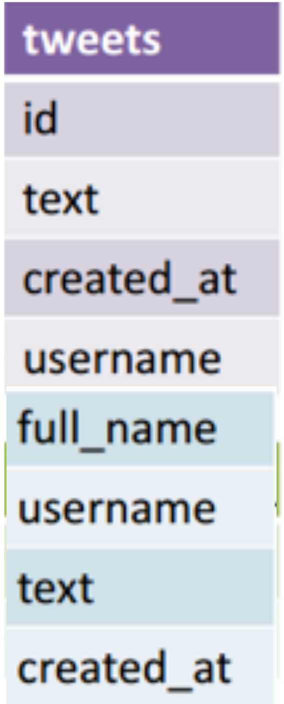
users
- full_name
- username
- text
- created_at

tweets
- id
- text
- created_at
- username

following
- from_user
- to_user

We could have had a table structure as follows:

Why is this different?



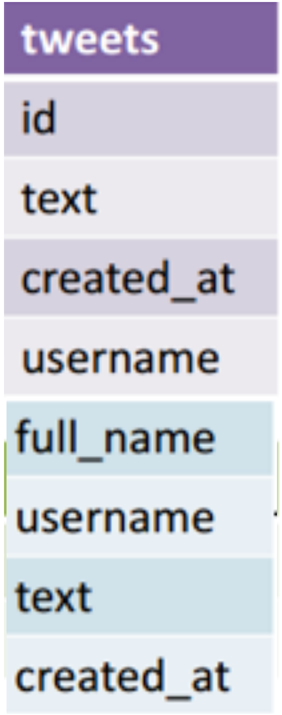
tweets
id
text
created_at
username
full_name
username
text
created_at

We could have had a table structure as follow:

Why is this different?

We would repeat the user information on each row.

This is called **denormalization**



tweets
id
text
created_at
username
full_name
username
text
created_at

Normalized Data: Many tables to reduce redundant or repeated data in a table

Denormalized Data:
Wide data, fields are often repeated but removes the need to join together multiple tables

Trade off of speed vs. storage

Q: How do we commonly evaluate databases?

Q: How do we commonly evaluate databases?

read-speed vs. write speed

Q: How do we commonly evaluate databases?

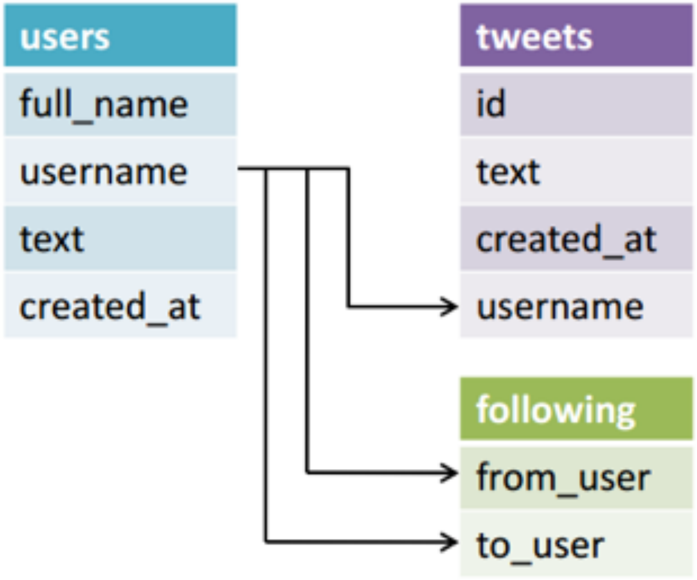
read speed vs. write speed

space considerations

(...and many other criteria)

NORMALIZED VS DENORMALIZED

Q: Why are normalized tables (possibly) slower to read?



users
full_name
username
text
created_at

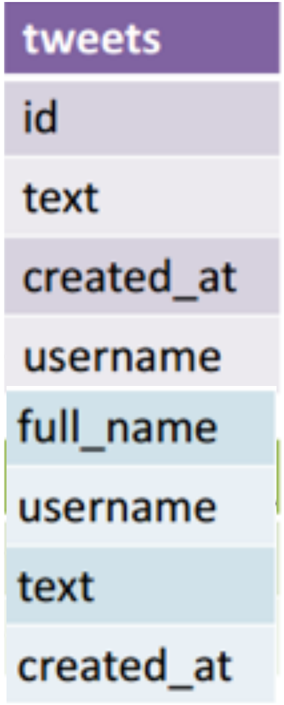
tweets
id
text
created_at
username

following
from_user
to_user

Q: Why are normalized tables (possibly) slower to read?

A: We'll have to get data from multiple tables to answer some questions.

Q: Why are denormalized tables (possibly) slower to write?



tweets
id
text
created_at
username
full_name
username
text
created_at

Q: Why are denormalized tables (possibly) slower to write?

A: We’ll have to write more information on each write

III. SQL

SQL is a query language to load, retrieve and update data in relational databases

SQL is **declarative***:*

- *Tell a database* **what** *you want, not* **how** *to do it*
- *SQL interfaces can be built on top of many tools*
- *The underlying concepts are general!*

THE SELECT COMMAND

SELECT: *Allows you to* **retrieve** *information from a table*

Syntax:

```
SELECT col1, col2 FROM table WHERE <some condition>
```

Example:

```
SELECT poll_title, poll_date FROM polls WHERE trump_pct > clinton_pct
```

GROUP BY: *Allows you to* **aggregate** *information from a table*

Syntax:

```
SELECT col1, AVG(col2) FROM table GROUP BY col1
```

Example:

```
SELECT poll_date, AVG(clinton_pct) FROM polls GROUP BY poll_date
```

GROUP BY: *Allows you to* **aggregate** *information from a table*

Syntax:

SELECT col1, AVG(col2) FROM table GROUP BY col1

There are usually a few common built-in operations:
SUM, AVG, MIN, MAX, COUNT

JOIN: *Allows you to* **combine** *multiple tables*

Syntax:

```
SELECT table1.col1, table1.col2, table2.col2
FROM  table1 JOIN table2 ON table1.col1 = table2.col2
```

JOIN: *Allows you to* **combine** *multiple tables*

Syntax:

```
SELECT table1.col1, table1.col2, table2.col2
FROM  (JOIN table1, table2 ON table1.col1 = table2.col2)
```

INSERT: *Allows you to* **add** *data to tables*

Syntax and Example:

```
INSERT INTO <table> (col1, col2)
VALUES( ...)
```

```
INSERT INTO classroom (first_name, last_name)
VALUES(‘John’, ‘Doe’);
```

OTHER COMMON SQL COMMANDS

Tutorial: http://www.w3schools.com/sql/default.asp

Other Commands: DISTINCT, ORDER BY, AND/OR, UPDATE, DELETE, LIKE, IN, HAVING, CREATE, DROP, ALTER…

HANDS-ON: FUN WITH SQL