GENERAL ASSEMBLY

INTRO TO DATA SCIENCE LESSON 3: INTRO TO DATABASES

LAST TIME...

WHAT IS MACHINE LEARNING?
TYPES OF MACHINE LEARNING PROBLEMS
DESIGNING A REGRESSION USING MATRICES
USING NUMPY, SCIPY, AND PANDAS

QUESTIONS?

I. INTRO TO DATABASES
II. RELATIONAL DATABASES / SQL

LAB
III. SQL LAB
IV. INTERACTING WITH SQL FROM PANDAS

I. INTRO TO DATABASES

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

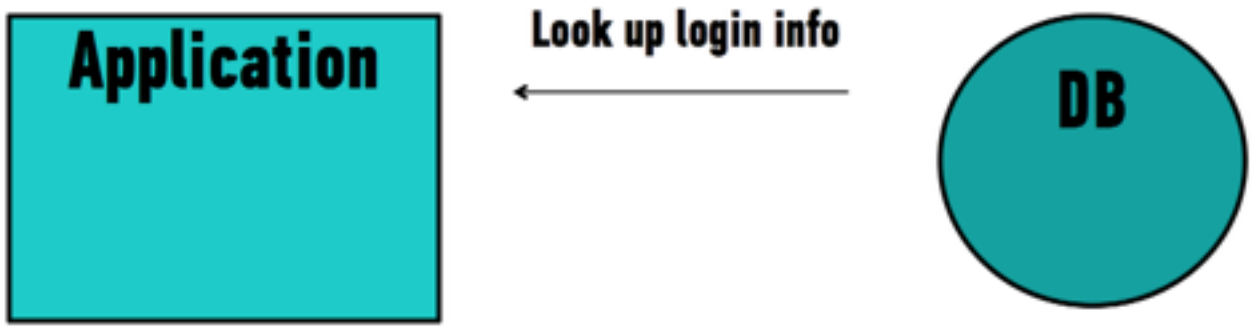
Databases are a **structured** data source optimized for efficient **retrieval and persistent storage.**

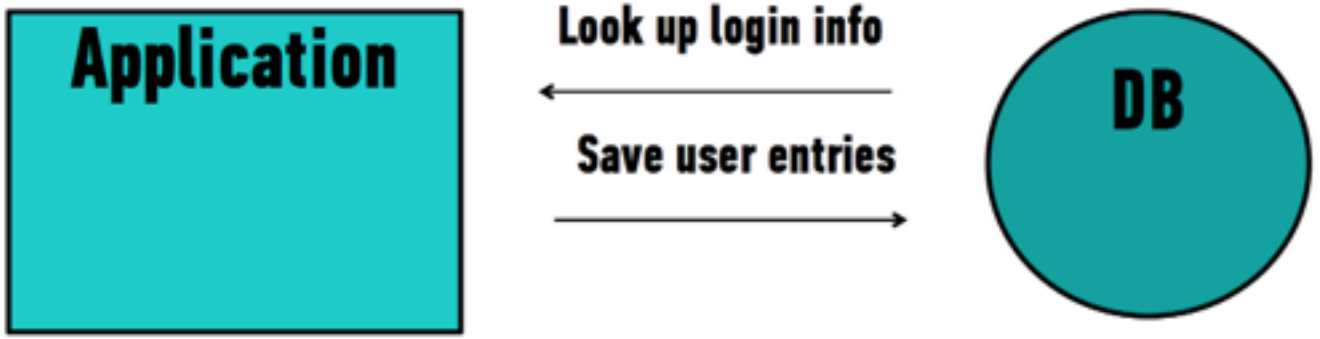
structured: we have to pre-define organization strategy

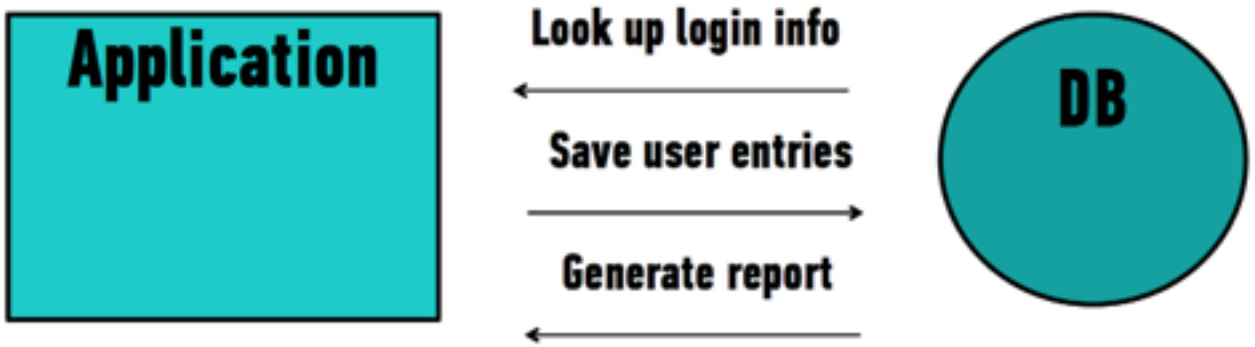
retrieval: the ability to read data out

storage: the ability to write data and save it


Application
DB


Application
Look up login info
DB


Application
Look up login info
Save user entries
DB


Application
Look up login info
Save user entries
Generate report
DB

II. RELATIONAL DATABASES

Relational databases are traditionally organized in the following manner:

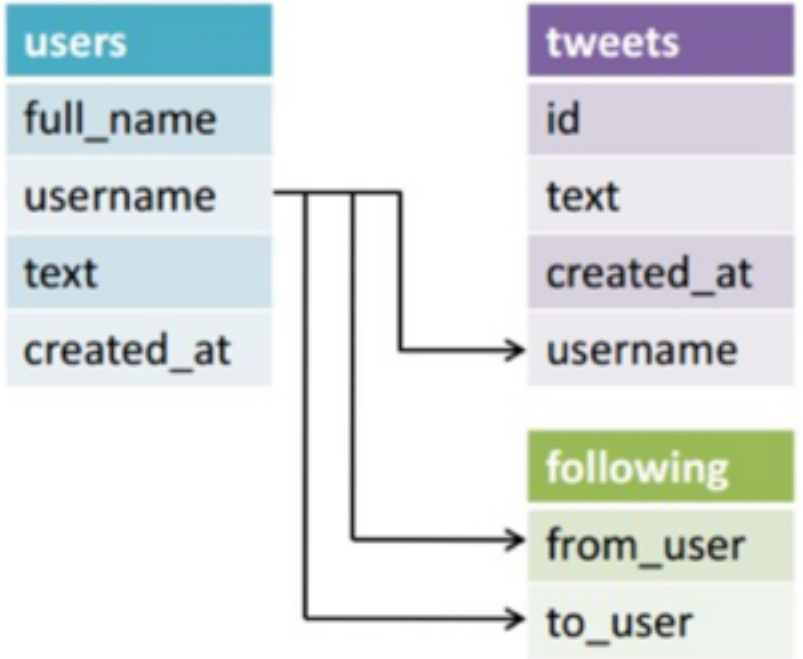
A database has tables which represent individual entities or objects

Tables have predefined schema – rules that tell it what the data will look like

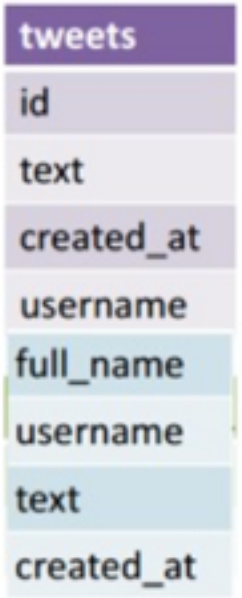
Each table should have a **primary key** column – a unique identifier for that row

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 links this to another table.


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

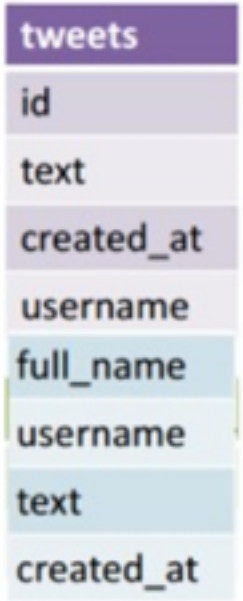
We could have had a table structure as follows:
Why is this different?



We could have had a table structure as follows:
Why is this different?

We would repeat the user information in each row.

This is called **denormalization**.



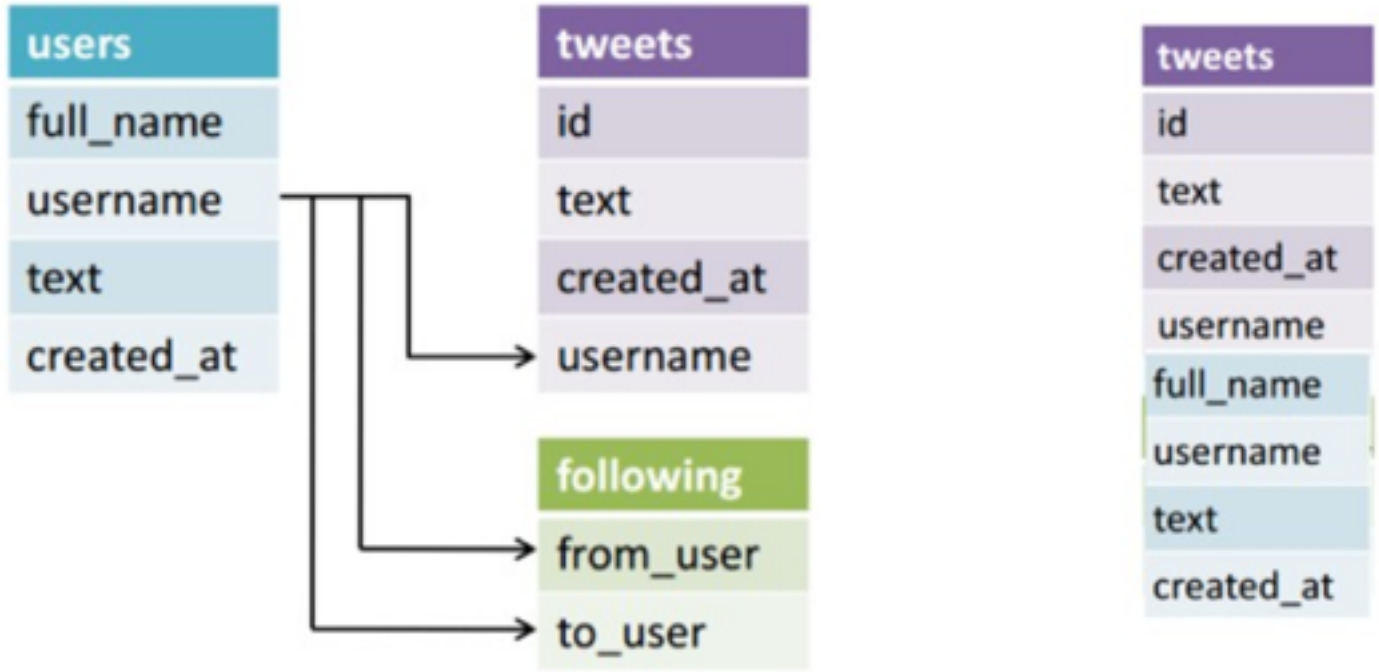
Normalized Data: Many tables to reduce redundant or repeated data in a table

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

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

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

This is a trade off of speed vs storage.

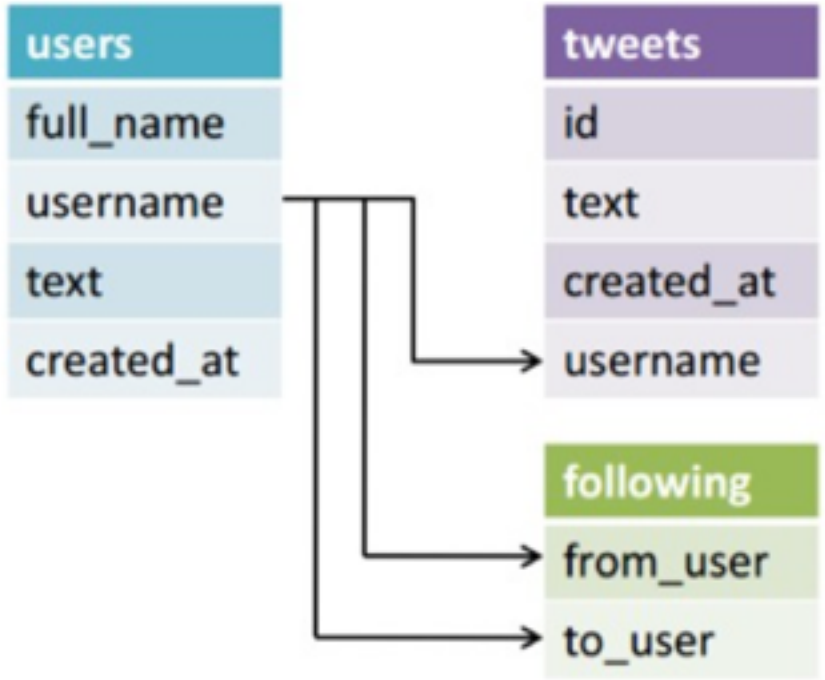

users
full_name
username
text
created_at
tweets
id
text
created_at
username
following
from_user
to_user
tweets
id
text
created_at
username
full_name
username
text
created_at

Q: How do we commonly evaluate databases?

read-speed vs write-speed
space considerations
(and many, many other criteria)

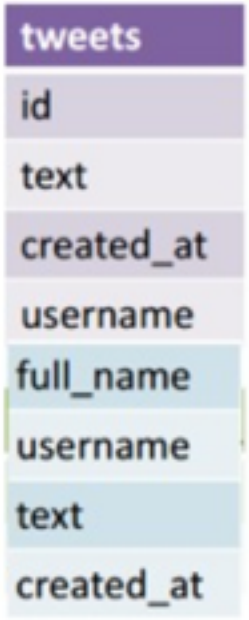
Q: Why are normalized tables (possibly) slower to read?

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


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

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

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


tweets
id
text
created_at
username
full_name
username
text
created_at

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

Most commonly known SQL-like Databases include:
Oracle
MySQL
PostgreSQL

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
romney_pct > obama_pct
```

GROUP BY: Allows you to aggregate information.

Syntax:

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

Example:

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

GROUP BY: Allows you to aggregate information.

Syntax:

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

Example:

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

GROUP BY: Allows you to aggregate information.

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 t1.c1, t1.c2, t2.c2
FROM t1 JOIN t2 ON t1.c1 = t2.c2
```

JOIN: Allows you to combine multiple tables

Syntax:

```
SELECT t1.c1, t1.c2, t2.c2
FROM t1 JOIN t2 ON t1.c1 = t2.c2
```

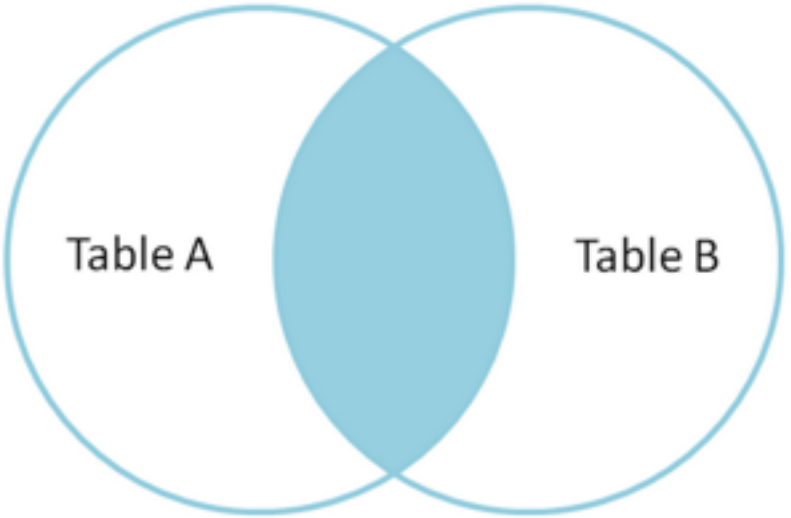
```
id name        id  name
-- ----        --  ----
1  Pirate      1   Rutabaga
2  Monkey      2   Pirate
3  Ninja       3   Darth Vader
4  Spaghetti   4   Ninja
```

Let's join these tables by the name field in a few different ways and see if we can get a conceptual match to those nifty Venn diagrams.

```
SELECT * FROM TableA
INNER JOIN TableB
ON TableA.name = TableB.name

id  name        id   name
--  ----        --   ----
1   Pirate      2    Pirate
3   Ninja       4    Ninja
```

Inner join produces only the set of records that match in both Table A and Table B.



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

Syntax:

```
INSERT INTO table1 (col1, col2)
VALUES (...)
```

```
INSERT INTO classroom (first_name, last_name)
VALUES ('John', 'Doe')
```

NoSQL databases are a new trend in databases

The name **NoSQL** refers to the lack of a relational structure between stored objects.

Most importantly they attempt to minimize the need for **JOIN** operations, or solve other data needs

Memcached	::	Livejournal
Apache HBase	::	Google BigTable
Cassandra	::	Amazon Dynamo
MongoDB	::	10Gen
Hadoop	::	Google MapReduce

LAB