

DATA SCIENCE

LECTURE 3: DATA FORMAT, ACCESS & TRANSFORMATION

YUCHEN ZHAO / DAT-14

LAST TIME:

I. WHAT IS MACHINE LEARNING?

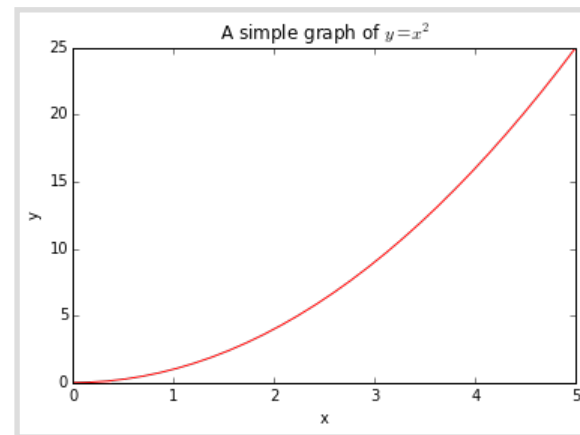
II. MACHINE LEARNING PROBLEMS

EXERCISES:

III. I-PYTHON NOTEBOOK INTRO

QUESTIONS?

	<i>continuous</i>	<i>categorical</i>
<i>supervised</i>	<i>regression</i>	<i>classification</i>
<i>unsupervised</i>	<i>dimension reduction</i>	<i>clustering</i>





I. DATA RETRIEVAL

II. ETL INTRO

III. VISUALIZATION

EXERCISES:

IV. PANDAS

V. MINING TWITTER VIA API

WHERE DOES THE DATA COME FROM?

DATA FLOW

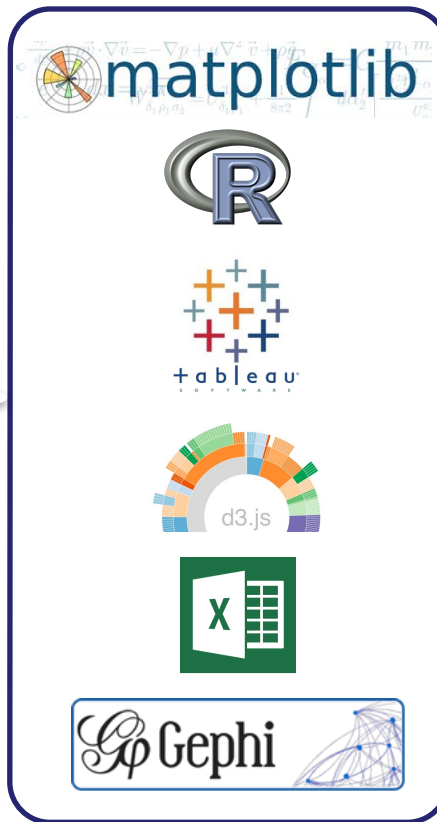
Data Retrieval



Data ETL and Aggregation



Data Visualization

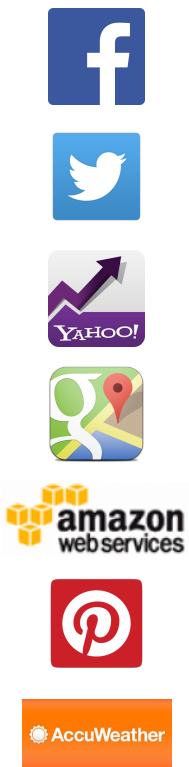


Machine Learning



DATA FLOW

Data Retrieval



Data ETL and Aggregation



Data Visualization



Machine Learning



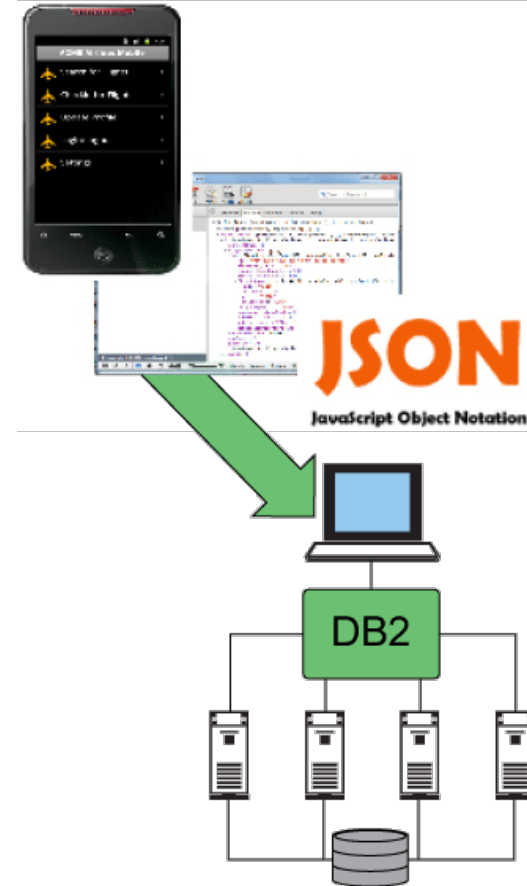
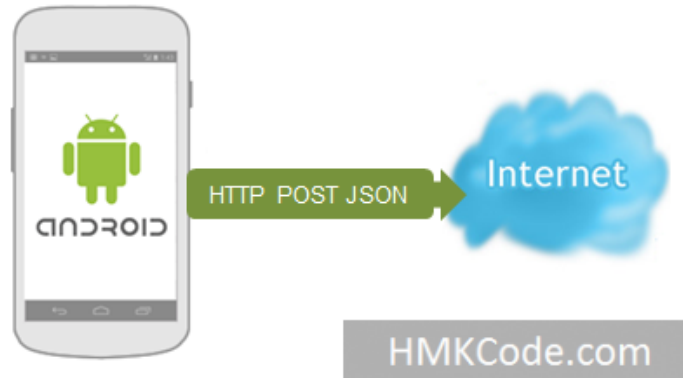
I. APIS AND JSON

JSON (JavaScript Object Notation) is:
a lightweight **data-interchange format**
a **string**

JSON

JSON can be passed
between applications

easy for machines to parse and generate



JSON are passed through applications
as **strings**
and converted into native objects per language.

JSON

JSON are passed
through applications
as **strings**
and converted into native
objects per language.

```
{ "empinfo" :  
  {  
    "employees" : [  
      {  
        "name" : "Scott Philip",  
        "salary" : £44k,  
        "age" : 27,  
      },  
      {  
        "name" : "Tim Henn",  
        "salary" : £40k,  
        "age" : 27,  
      },  
      {  
        "name" : "Long Yong",  
        "salary" : £40k,  
        "age" : 28,  
      }  
    ]  
  }  
}
```

```
import json  
  
py_object = [ { 'a':'A', 'b':(2, 4), 'c':3.0 } ]  
  
json_string = json.dumps(py_object)  
  
print 'JSON:', json_string
```

```
JSON: [{"a": "A", "c": 3.0, "b": [2, 4]}]
```

```
decoded = json.loads(json_string)
```

<https://docs.python.org/2/library/json.html>

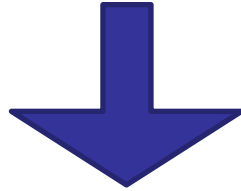
<https://docs.python.org/2/library/csv.html>

APIs (Application Programming Interface) allow people to **interact** with the structures of an application

- get
- put
- delete
- update
- ...

Best practices for APIs are to
use **RESTful** principles.

Best practices for APIs are to
use **RESTful** principles.



Representational State Transfer (REST)

RESTFUL EXAMPLE

RESTful API HTTP methods

Resource	GET	PUT	POST	DELETE
Collection URI, such as <code>http://example.com/resources/</code>	List the URIs and perhaps other details of the collection's members.	Replace the entire collection with another collection.	Create a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation. ^[9]	Delete the entire collection.
Element URI, such as <code>http://example.com/resources/item17</code>	Retrieve a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	Replace the addressed member of the collection, or if it does not exist, create it.	Not generally used. Treat the addressed member as a collection in its own right and create a new entry in it. ^[9]	Delete the addressed member of the collection.

http://en.wikipedia.org/wiki/Representational_state_transfer

- The Base URL
- An interactive media type (usually JSON)
- Operations (GET, PUT, POST, DELETE)
- Driven by http requests

TWITTER REST API

<https://dev.twitter.com/rest/public>

LINKEDIN REST API

<https://developer.linkedin.com/docs/signin-with-linkedin>

AWESOME PUBLIC DATASETS

<https://github.com/caesar0301/awesome-public-datasets>

DATA FLOW

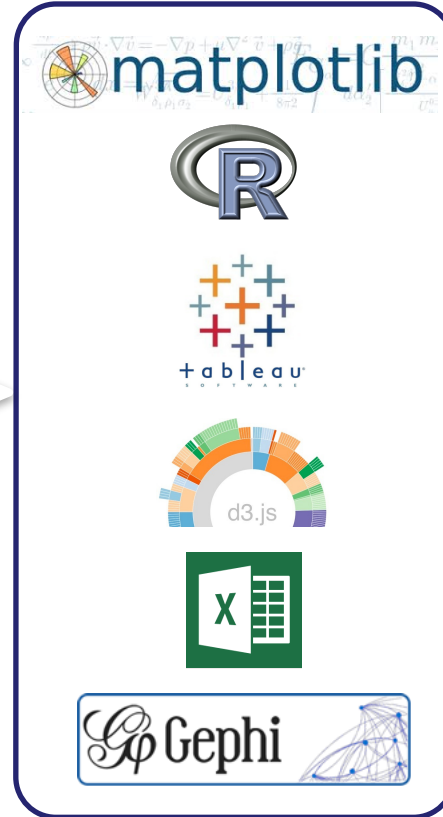
Data Retrieval



Data ETL and Aggregation



Data Visualization



Machine Learning



II. INTRO TO RELATIONAL DATABASE

What is ETL?

- **E**xtract data
- **T**ransform data
- **L**oad data

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

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

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

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

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



Application

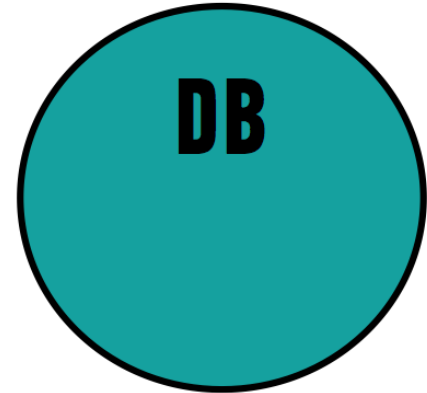
DB

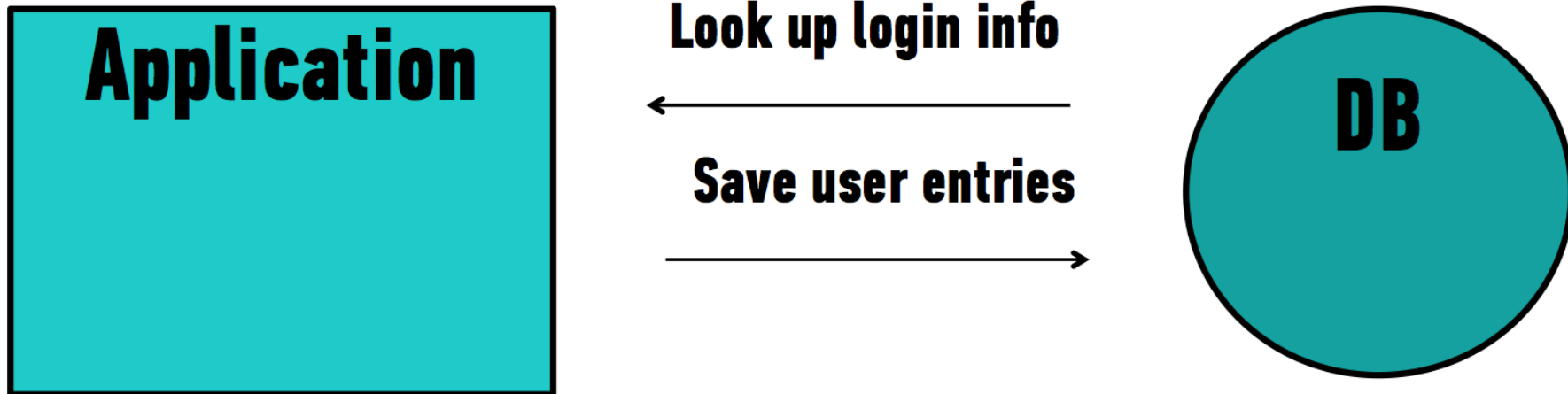


Look up login info



A horizontal black arrow pointing from the database towards the application.



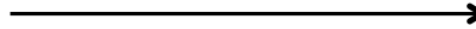




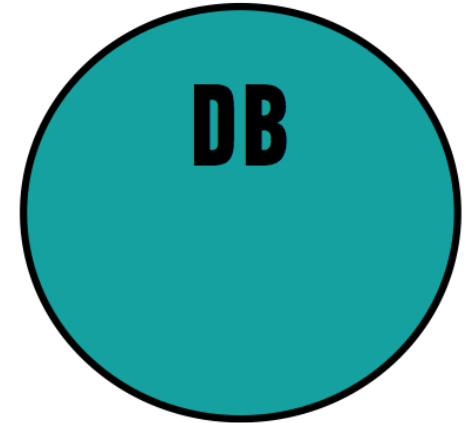
Look up login info



Save user entries



Generate report



INTRO TO DATA SCIENCE

RELATIONAL DATABASES

Relational database are traditionally organized in the following manner:

- *A database has **tables** which represent individual entities or objects*

RELATIONAL DATABASES

Students Table

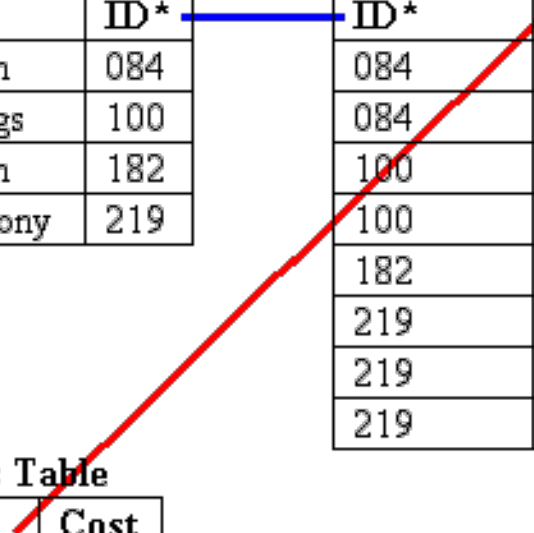
Student	ID *
John Smith	084
Jane Bloggs	100
John Smith	182
Mark Antony	219

Participants Table

ID *	Activity *
084	Tennis
084	Swimming
100	Squash
100	Swimming
182	Tennis
219	Golf
219	Swimming
219	Squash

Activities Table

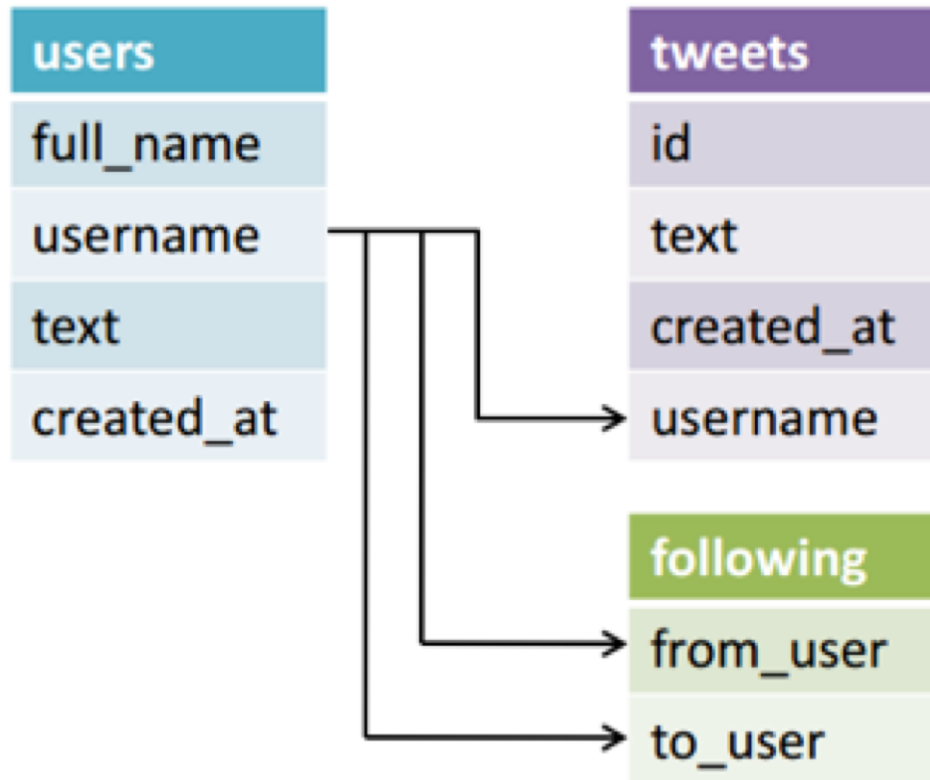
Activity *	Cost
Golf	\$47
Sailing	\$50
Squash	\$40
Swimming	\$15
Tennis	\$36



Relational database are traditionally organized in the following manner:

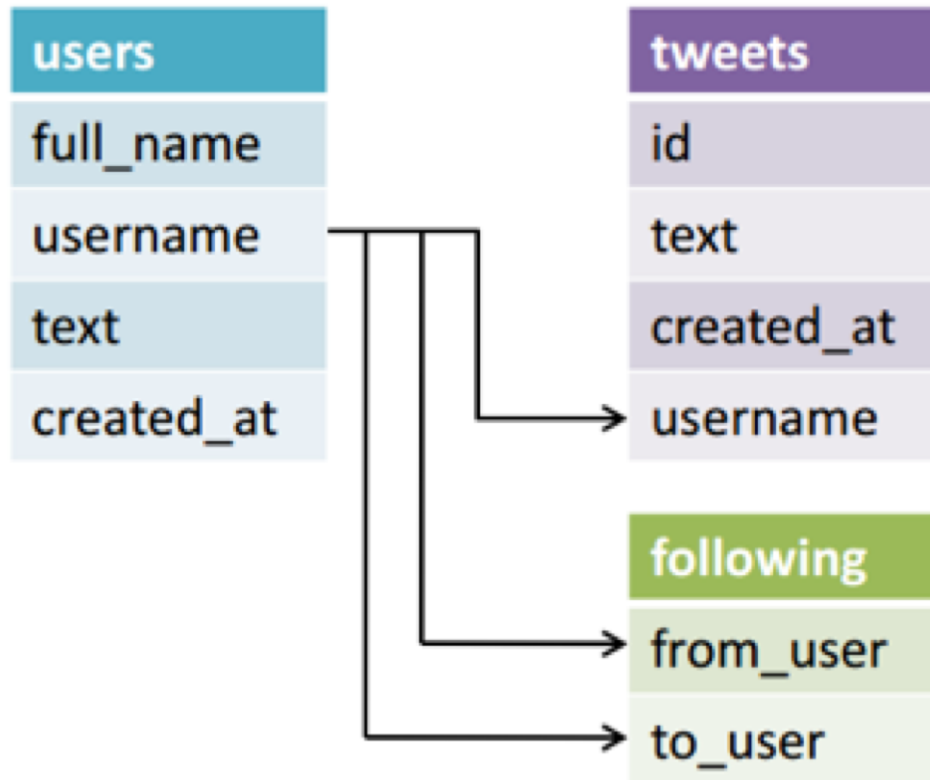
- *A database has **tables** which represent individual entities or objects*
- *Tables have a predefined **schema** - rules that tell it what columns exist and what they look like*

RELATIONAL DATABASES



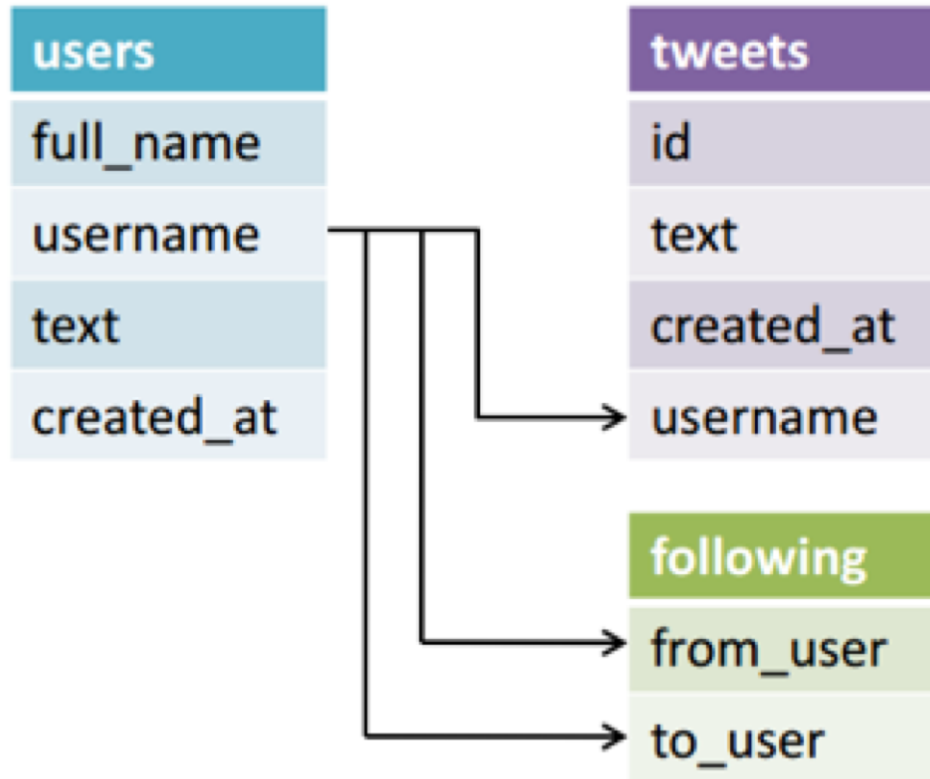
Each table should have a **primary key** column- a unique identifier for that row

RELATIONAL DATABASES



*Additionally each table can have a
foreign key column- an id that links this
to table to another*

RELATIONAL DATABASES



We could have had a table structure like this:

Why is this different?

tweets

id

text

created_at

username

full_name

username

text

created_at

We would repeat the user information on each row.

This is called **denormalization**.

tweets

id

text

created_at

username

full_name

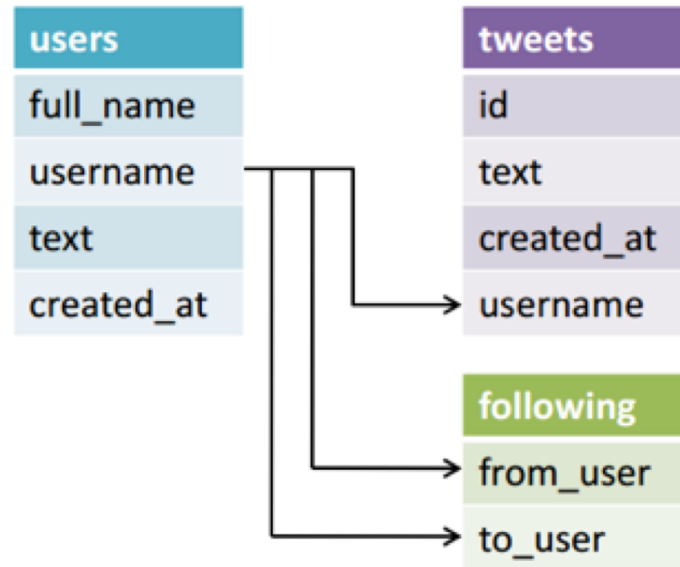
username

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

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

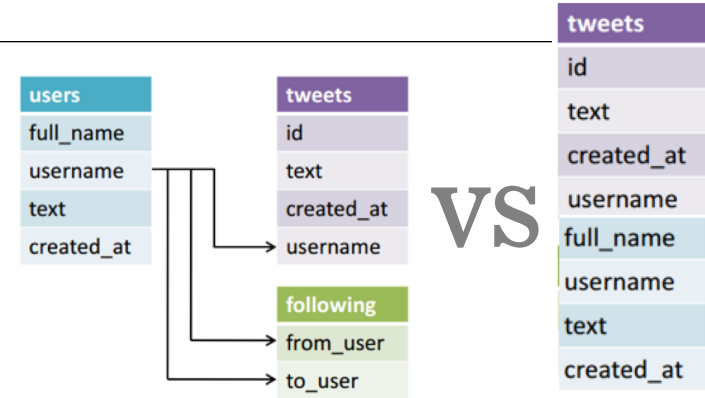
space considerations

(...and many other criteria)

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

NORMALIZED VS DENORMALIZED

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

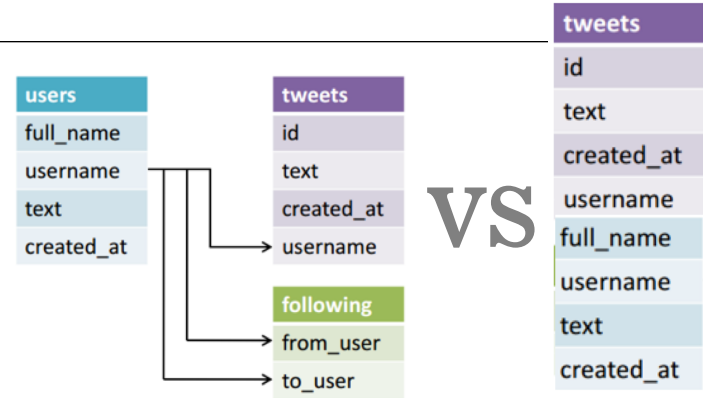


A: We'll have to get data from multiple tables to answer some questions.

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

NORMALIZED VS DENORMALIZED

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



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

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

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

Syntax:

SELECT *col1, col2, ...*

FROM *table*

WHERE *<some condition>*

THE SELECT COMMAND

Syntax:

SELECT col1, col2, ...

FROM table

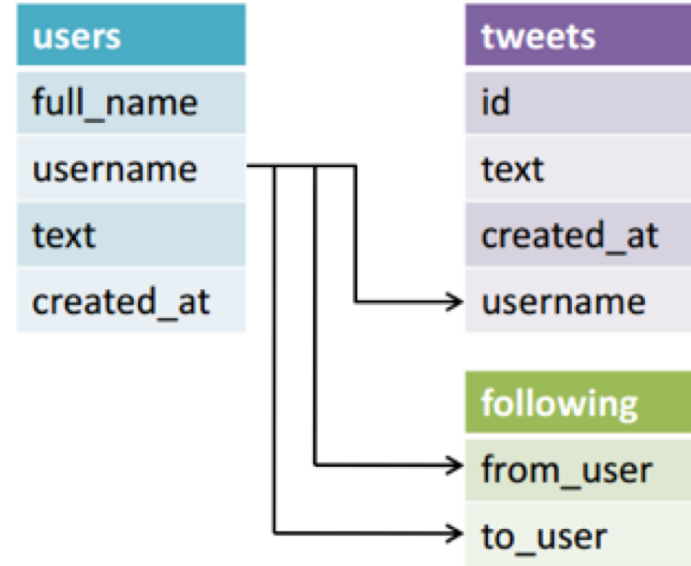
WHERE <some condition>

Example:

SELECT full_name, text

FROM users

WHERE created_at > '2015-05-01'



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

Syntax:

SELECT ***coll***, *count(col2), ...*

FROM *table*

GROUP BY ***coll***

THE GROUP BY COMMAND

Syntax:

SELECT col1, count(col2), ...

FROM table

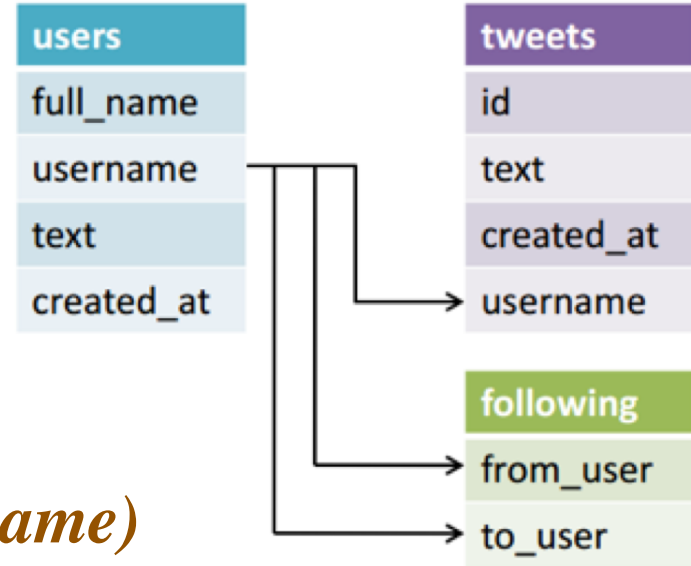
GROUP BY col1

Example:

SELECT created_at, count(username)

FROM users

GROUP BY created_at



THE GROUP BY COMMAND

common group by functions:

- **count**
- **max**
- **min**
- **avg**
- **sum**

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

Syntax:

***SELECT** table1.col1, table1.col2, table2.col2, ...*

***FROM** table1 **JOIN** table 2*

***ON** table1.col1 = table2.col1*

THE JOIN COMMAND

Syntax:

SELECT table1.col1, table1.col2, table2.col2, ...

*FROM table1 **JOIN** table 2*

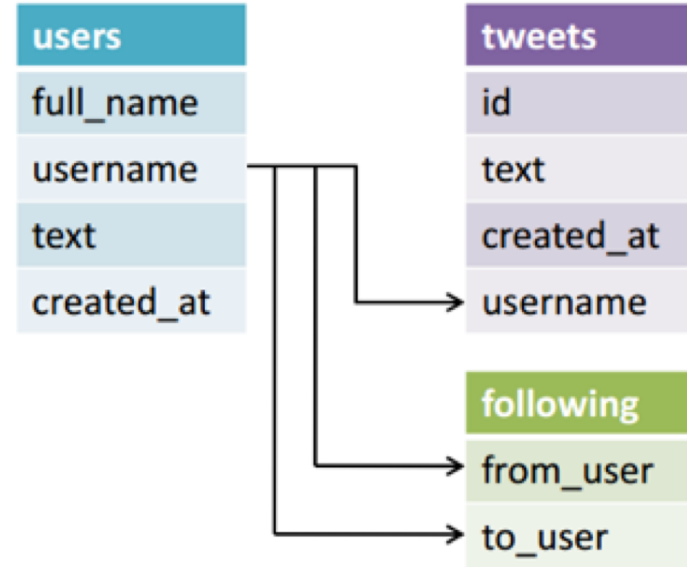
***ON** table1.col1 = table2.col1*

Example:

SELECT users.full_name, tweets.text

*FROM users **JOIN** tweets*

***ON** users.username = tweets.username*



WANT TO KNOW MORE ABOUT SQL?

<http://www.w3schools.com/sql/>

III. VISUALIZATION

DATA FLOW

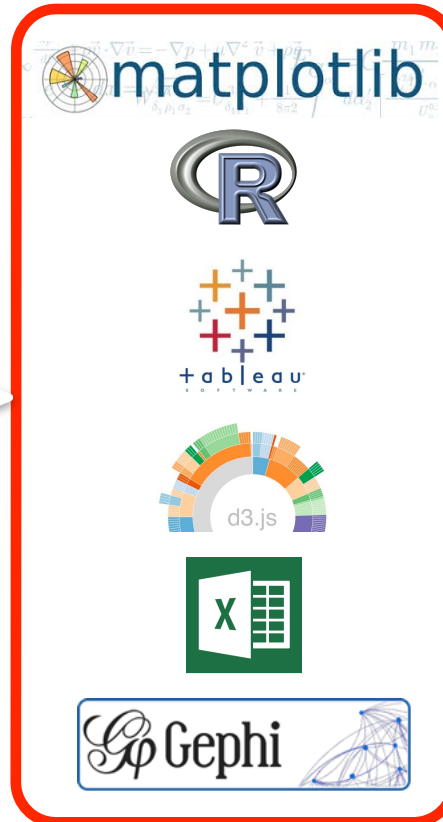
Data Retrieval



Data ETL and Aggregation



Data Visualization



Machine Learning



[https://github.com/mbostock/
d3/wiki/Gallery](https://github.com/mbostock/d3/wiki/Gallery)

IV. PANDAS INTRO

V. MINING TWITTER VIA API

HOMEWORK 1:

**[HTTPS://GITHUB.COM/GA-STUDENTS/
DAT_SF_14/TREE/MASTER/HOMEWORK/
HW1](https://github.com/GA-STUDENTS/DAT_SF_14/TREE/MASTER/HOMEWORK/HW1)**

INTRO TO DATA SCIENCE

DISCUSSION