GENERAL ASSEMBLY

DATA SCIENCE

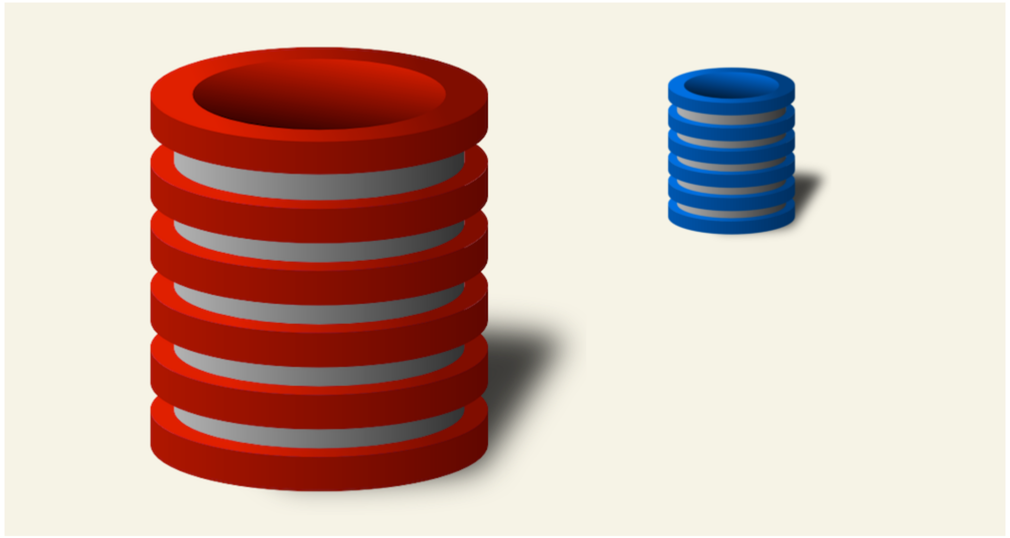
DATABASES / SQL

AGENDA

I. DATABASES

DATABASES

- An organized collection of data
- Organized using a schema (like a blueprint of a database)
- Organized into tables with different sets of data

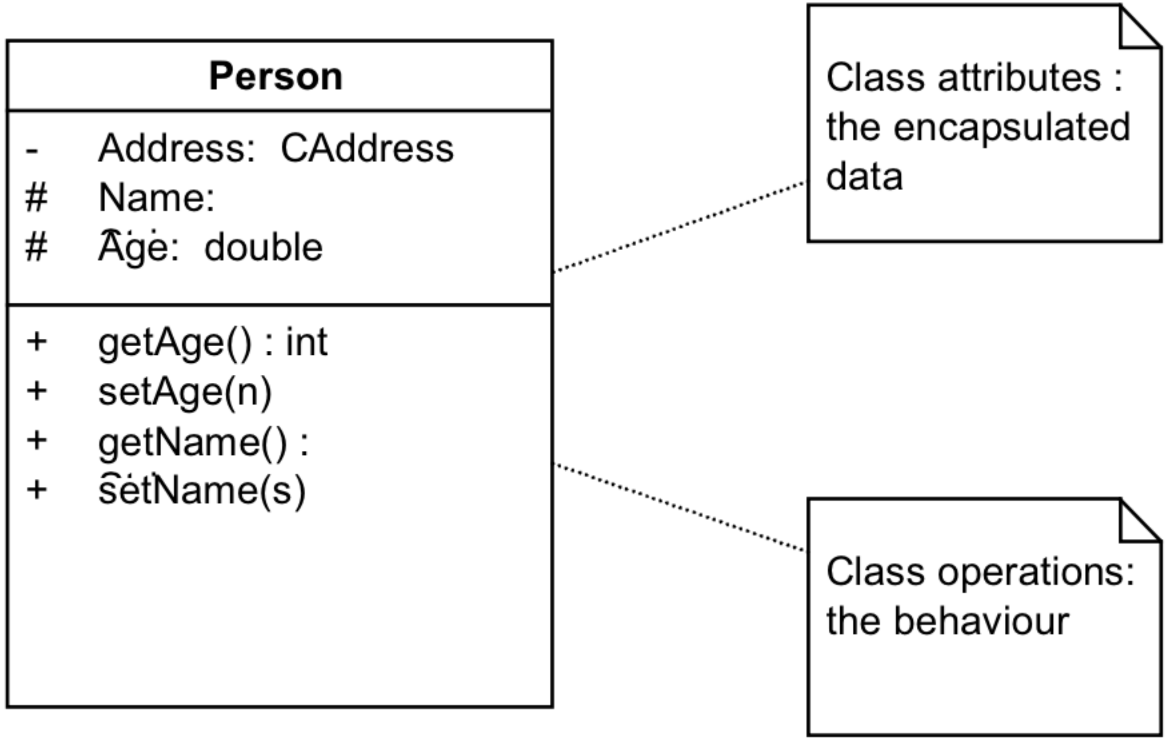


WHY EVEN USE A DATABASE?

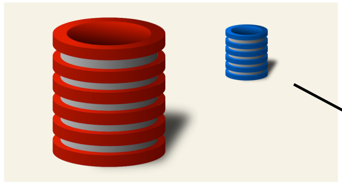
- Easy to store and more importantly, retrieve data
- Generally has a structured language for interacting with the data
- Reliable and **scalable**
- Access large amounts of data relatively quickly

HOW CAN YOU VISUALIZE A DATABASE?

UNIFIED
MODELING
LANGUAGE
UML™


Person
- Address: CAddress
# Name:
# Age: double
+ getAge() : int
+ setAge(n)
+ getName() :
+ setName(s)
Class attributes : the encapsulated data
Class operations: the behaviour

II. SQL / NOSQL

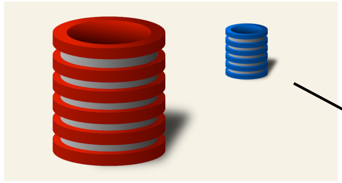


Relational

- Traditional rows and columns data
- **Strict** structure / Primary Keys
- Entire column for each feature
- Industry standard

NoSql

- No well defined data structure
- Works better for unstructured data
- Cheaper hardware
- Popular among Startups



Relational

Examples

- MySQL
- Oracle
- Postgres
- SQLite

NoSql

Examples

- MongoDB
- CouchDB
- Redis
- Casssandra

S
Q
L

Structured

Query

Language

Is a language for database communication

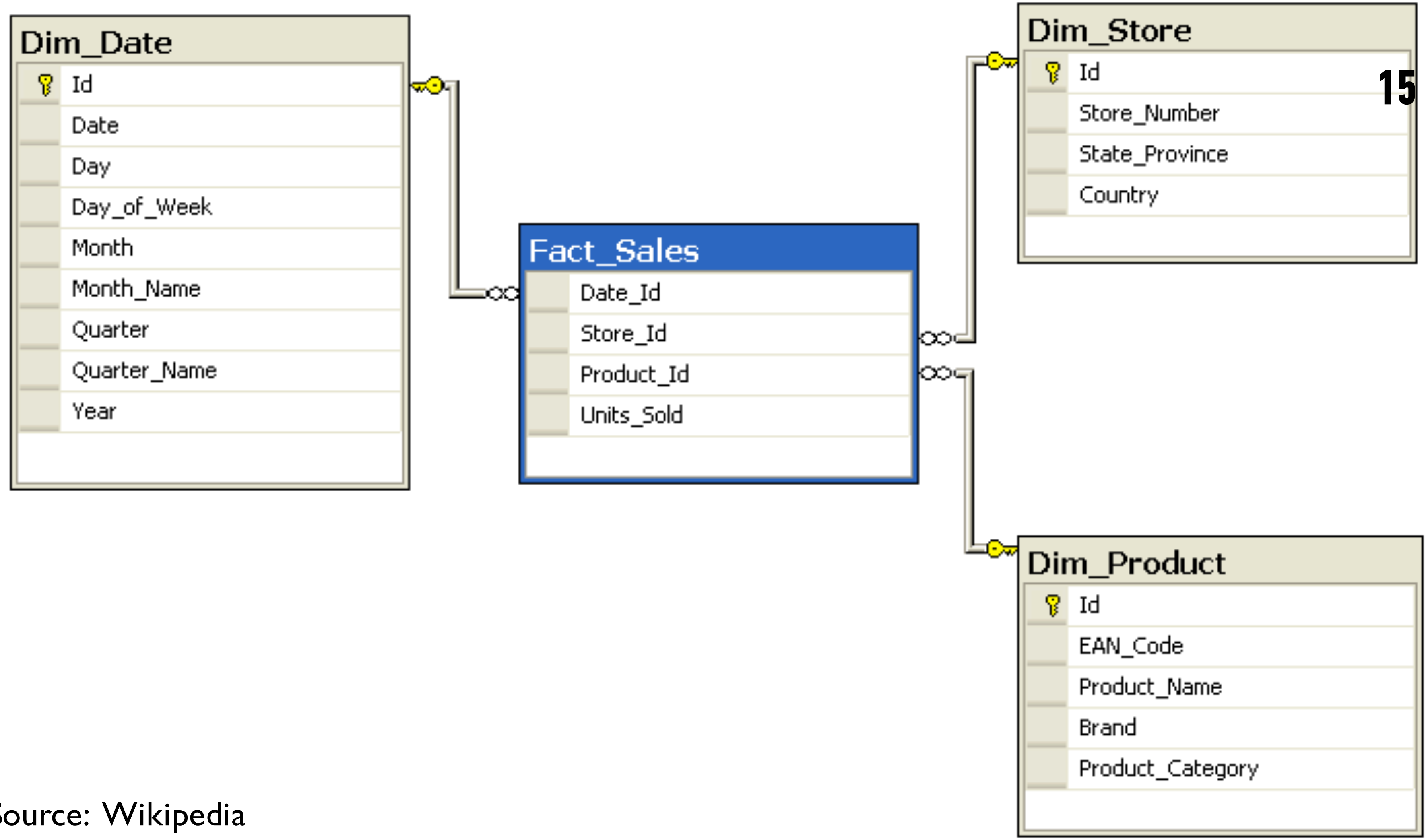
DATA TYPES

- BOOLEAN/TINY INT – 0/1
- INT – any whole number
- FLOAT(<n>,<m>) – number with n digits before the decimal and m digits after the decimal
- DATETIME, TIMESTAMP, and DATE – various date and time combinations
- CHAR(<length>) – text with a fixed length
- VARCHAR(<length>) – text with a given maximum length
- And many more…

STAR SCHEMAS

- The star schema consists of one or more fact tables referencing any number of dimension tables.
- A fact table contains "event" data. You can think of this as the type of information that we are really measuring ("measurements, metrics, or facts of a business process").
- A dimension table contains meta data or information that enhances "event" data ("structured labeling information").

Source: Wikipedia



Source: Wikipedia

IV. JOINS

JOINS

Easy way of combining rows from separate data tables

TYPES OF JOINS IN SQL

INNER JOIN: Returns rows when there is at least one match in **BOTH** tables (may **not** contain nulls)

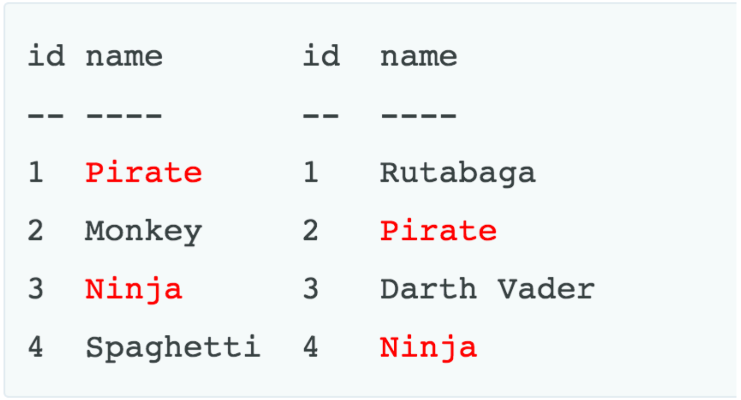
LEFT JOIN: Returns rows from the left table, and the matched rows from the right table (may contain nulls)

RIGHT JOIN: Returns rows from the right table, and the matched rows from the left table (may contain nulls)

FULL JOIN: Returns rows when there is a match in **AT LEAST ONE** of the tables (may contain nulls)

JOINS IN SQL

Table A Table B



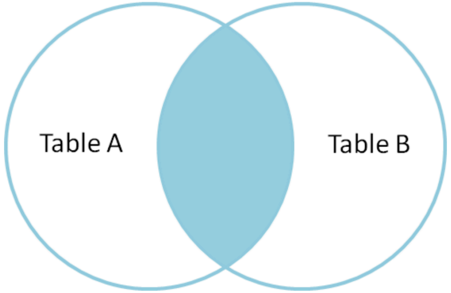
```
id name         id  name
-- ----         --  ----
1  Pirate       1   Rutabaga
2  Monkey       2   Pirate
3  Ninja        3   Darth Vader
4  Spaghetti    4   Ninja
```

Source: http://blog.codinghorror.com/a-visual-explanation-of-sql-joins/

INNER JOIN

Table A Table B

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



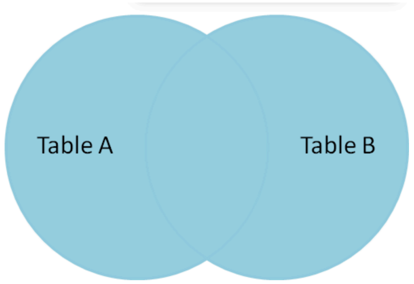
```
SELECT * FROM TableA
INNER JOIN TableB
ON TableA.name = TableB.name

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.

OUTER JOIN

Table A Table B

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



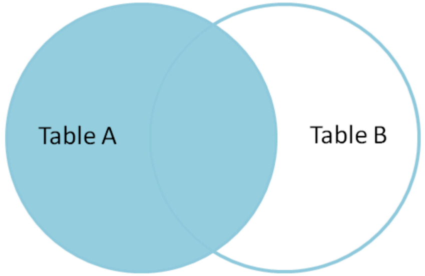
```
SELECT * FROM TableA
FULL OUTER JOIN TableB
ON TableA.name = TableB.name

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

Full outer join produces the set of all records in Table A and Table B, with matching records from both sides where available. If there is no match, the missing side will contain null.

LEFT JOIN

Table A Table B

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



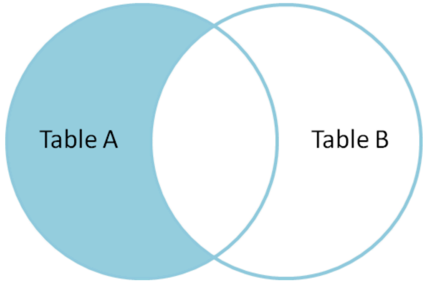
```
SELECT * FROM TableA
LEFT OUTER JOIN TableB
ON TableA.name = TableB.name

id  name       id    name
--  ----       --    ----
1   Pirate     2     Pirate
2   Monkey     null  null
3   Ninja      4     Ninja
4   Spaghetti  null  null
```

Left outer join produces a complete set of records from Table A, with the matching records (where available) in Table B. If there is no match, the right side will contain null.

LEFT JOIN CONTINUED

Table A Table B

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



```
SELECT * FROM TableA
LEFT OUTER JOIN TableB
ON TableA.name = TableB.name
WHERE TableB.id IS null


id   name         id      name
--   ----         --      ----
2    Monkey       null    null
4    Spaghetti    null    null
```

To produce the set of records only in Table A, but not in Table B, we perform the same left outer join, then **exclude the records we don't want from the right side via a where clause**.