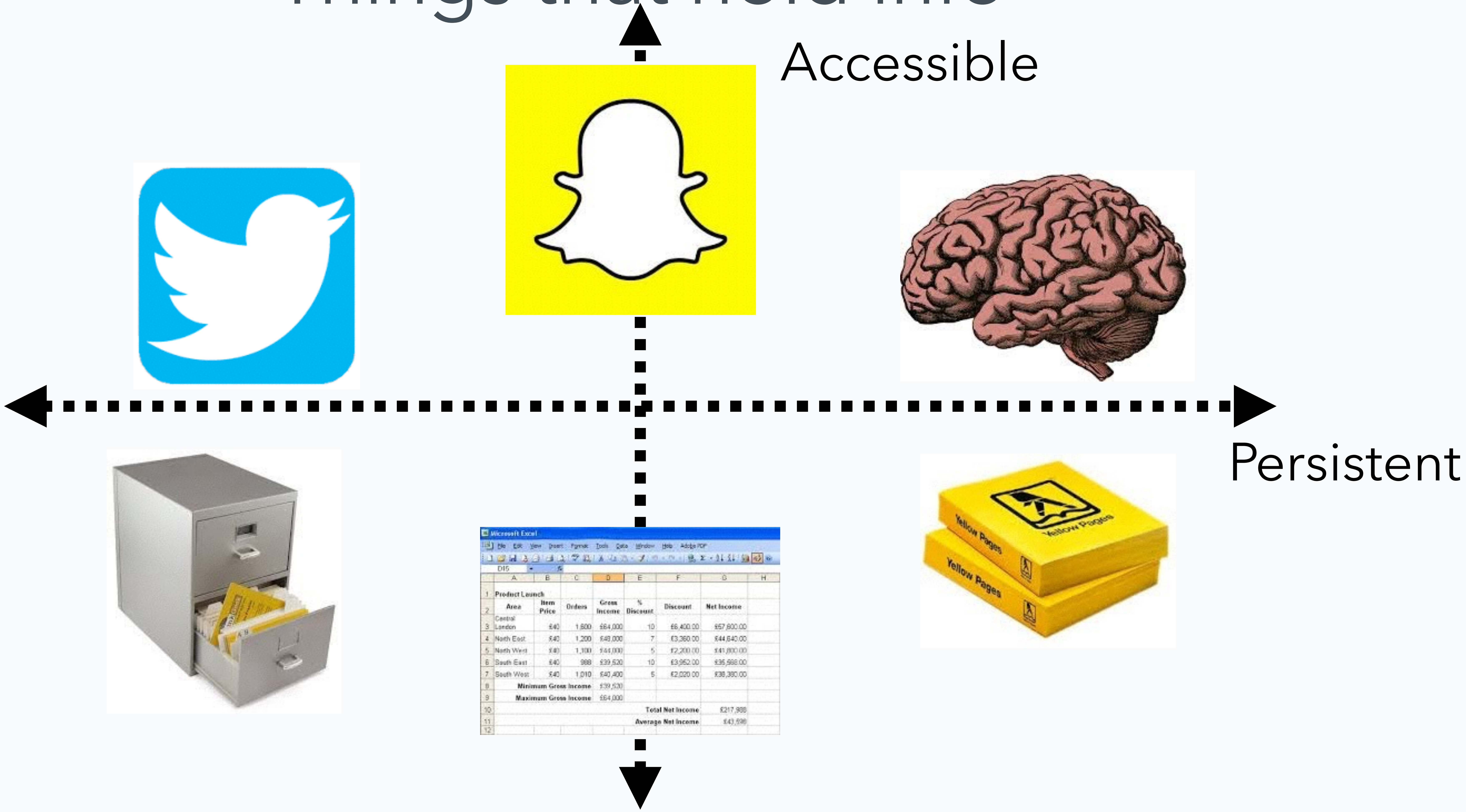
Intro to Databases

SQL

What is a database?

Things that hold info



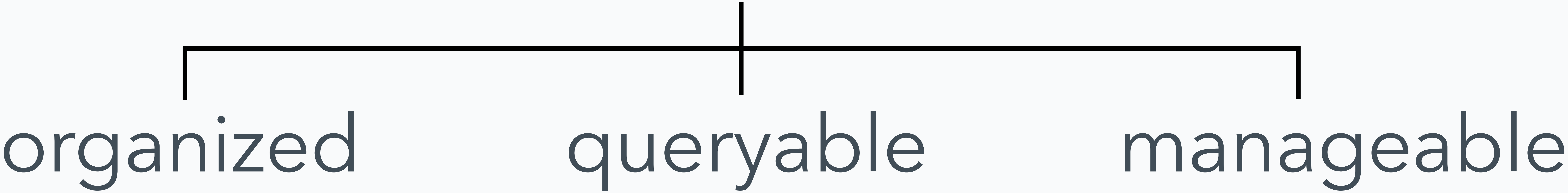
Accessible

Persistent

Yellow Pages
Yellow Pages
Yellow Pages
Microsoft Excel
File Edit View Insert Format Tools Data Window Help Adobe PDF
D15

	A	B	C	D	E	F	G	H
1	Product Launch							
2	Area	Item Price	Orders	Gross Income	% Discount	Discount	Net Income	
3	Central London	£40	1,600	£64,000	10	£6,400.00	£57,600.00	
4	North East	£40	1,200	£48,000	7	£3,360.00	£44,640.00	
5	North West	£40	1,100	£44,000	5	£2,200.00	£41,800.00	
6	South East	£40	988	£39,520	10	£3,952.00	£35,568.00	
7	South West	£40	1,010	£40,400	5	£2,020.00	£38,380.00	
8	Minimum Gross Income			£39,520				
9	Maximum Gross Income			£64,000				
10						Total Net Income	£217,988	
11						Average Net Income	£43,598	
12								

A database **persists** information and is **accessible** via code



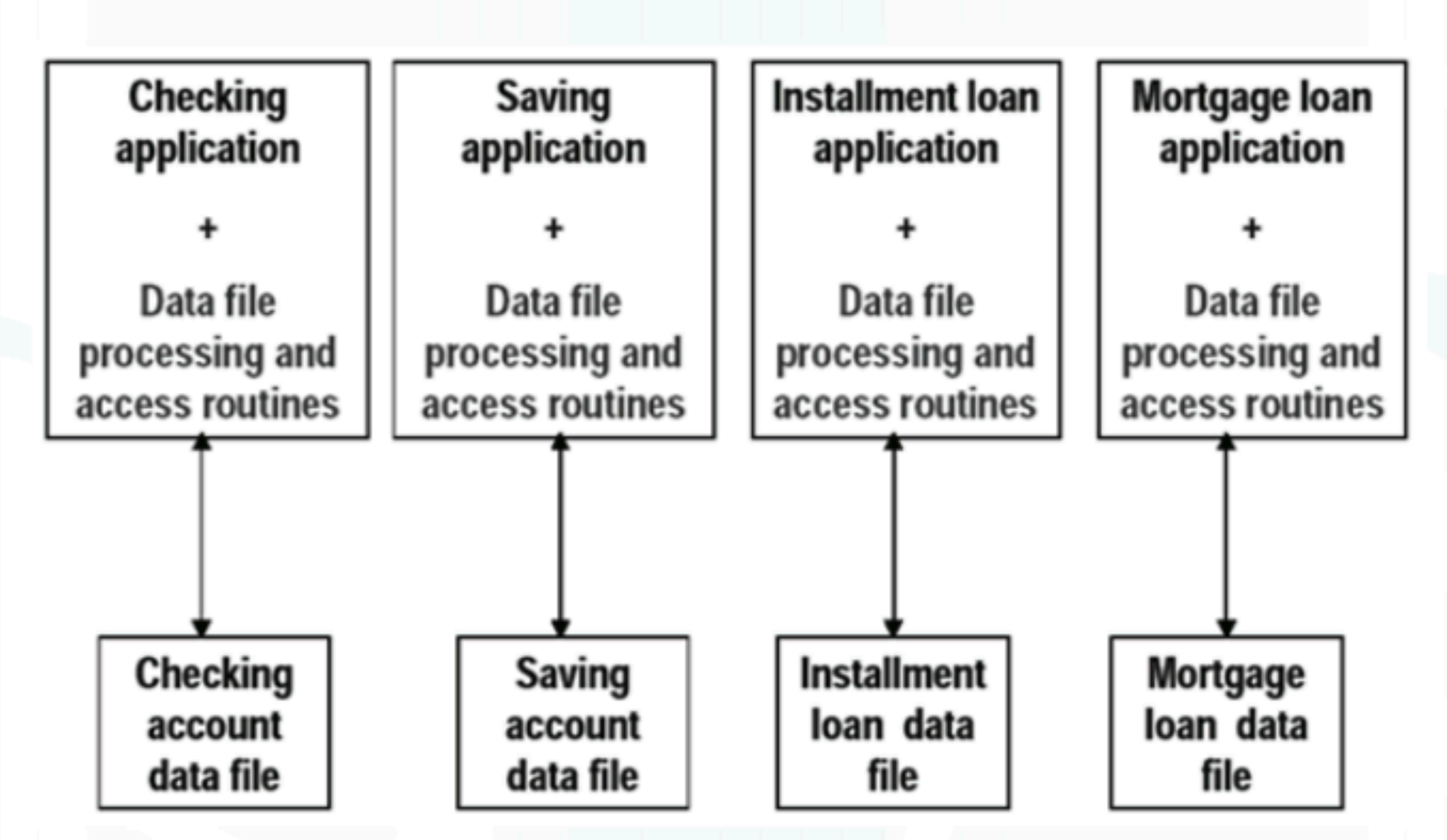
organized

queryable

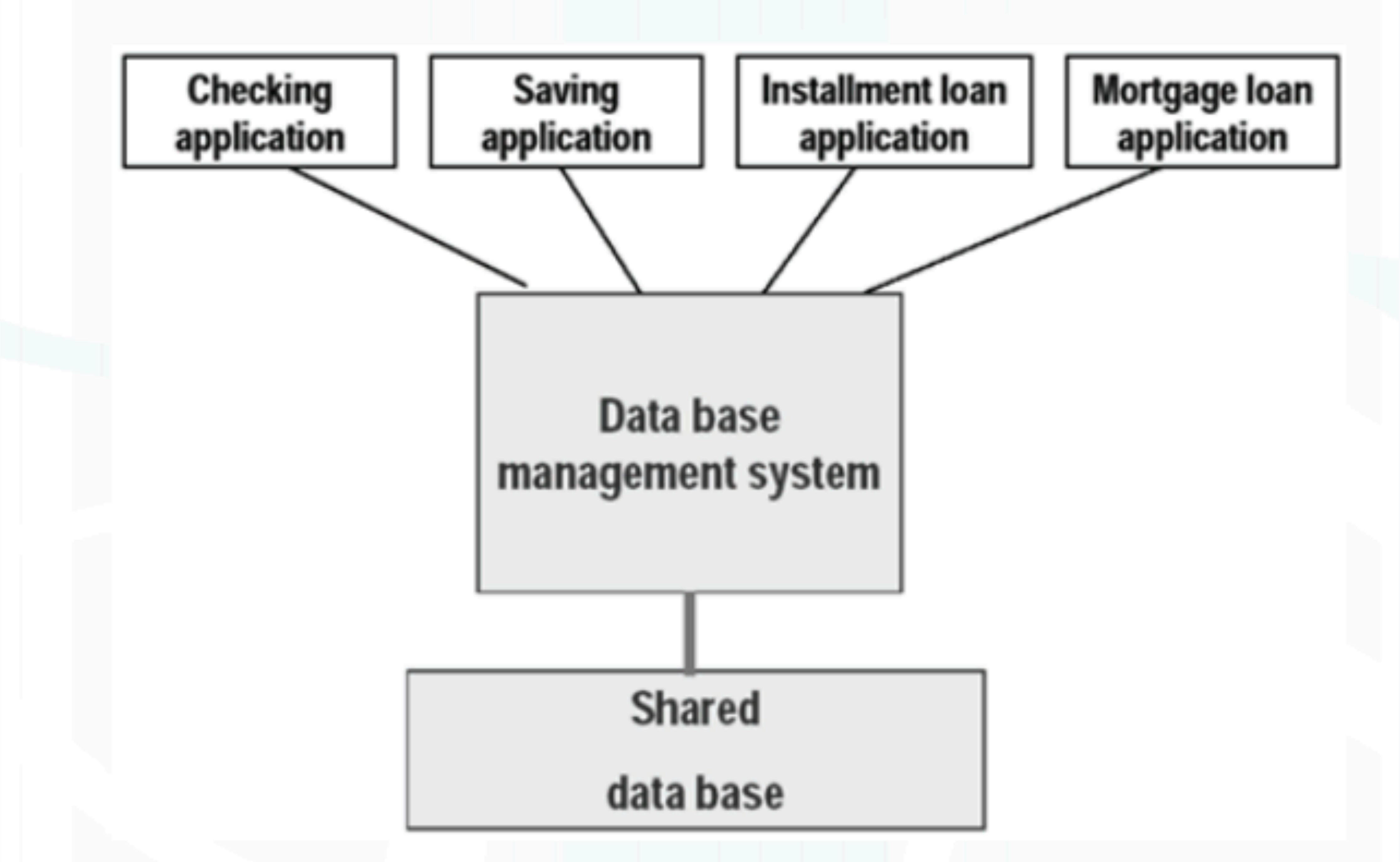
manageable

Before Relational DBs (ca. < 1970s)

- Data stored in custom “data files”
- Queried via application-specific code
- Advantages
  - Middle layer not needed
  - Solutions customized for each application
- Disadvantages
  - Hard to change the system
  - Knowledge not compounding
  - Data-transfer is difficult


Checking application + Data file processing and access routines
Saving application + Data file processing and access routines
Installment loan application + Data file processing and access routines
Mortgage loan application + Data file processing and access routines
Checking account data file
Saving account data file
Installment loan data file
Mortgage loan data file

Database Management Systems (DBMS)

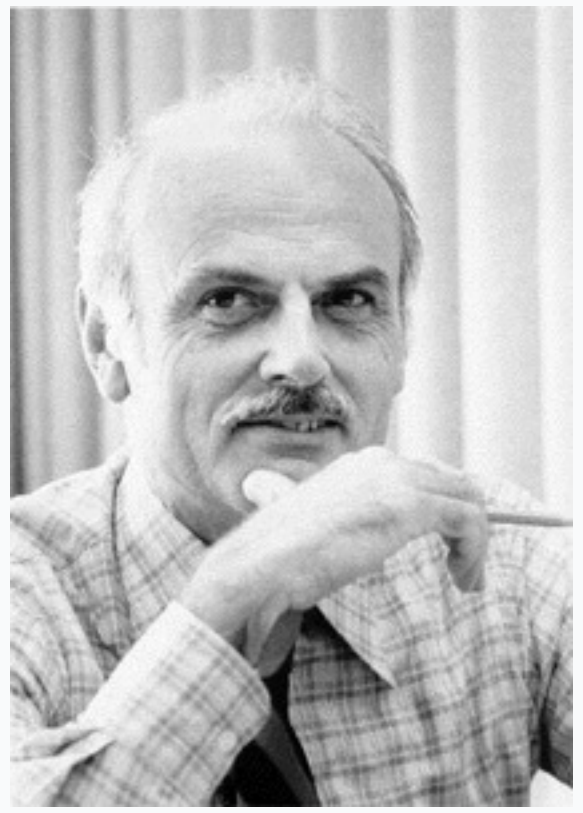

Checking application
Saving application
Installment loan application
Mortgage loan application
Data base management system
Shared data base

- One layer and language to store and access data
- Sold as a way for "non-technical people" to manage data

> “Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation).”

–E.F. Codd,
A Relational Model of Data for Large Shared Data Banks

Relational Databases & Logic



- 1969: Edgar Frank "Ted" Codd outlines *relational model* of data
- Wrote Alpha (never implemented) as a *query language*
- IBM slow to adopt his ideas
    - Competitors started to do so
    - IBM team formed without Codd, created **S**tructured **E**nglish **Qu**ery **L**ang
- SEQUEL way better than what came before
    - 1979: copied by Larry Ellison (from pre-launch papers / talks!) as "SQL"
- SQL became the standard (ANSI 1986, ISO 1987)
    - Codd continued to fault SQL compared to his theoretical model
    - The Third Manifesto: solve the *object-relational impedance mismatch*

Oracle

Ed Oates, Bruce Scott, Bob Miner, Larry Ellison

Appreciating Databases

- Ubiquitous
- Standardized
- Complex / deep
- Powerful: database admins are…
  - …feared by developers.
  - …but also taken for granted until things break.
  - …befriended by business people.
  - …contacted by the government for secret data (e.g. NSA).

RDBMS

- Data is stored in relations (tables)
- A simple, structured query language: SQL
  - Programmers can specify what answers a query should return, but not how the query is executed or where and how the data is stored
  - DBMS picks an execution strategy based on indexes, data, workload etc.
- Multi-user, Multi-threaded
  - Multiple processes can access database at same time

Definitions in a Database

- DBs are a collection of Tables (or relations)
- Tables have Columns (attributes) and Rows (instances or tuples)
- Duplicate rows are not allowed
- Rows often have a primary key (ID)

Schema and Content

- Schema: table’s blueprint for data shape/format
- Content: actual data (a row) e.g. {1, “Bart S.”, 10, “M”}
- A schema is used to validate incoming content

SQL

SQL is used to create/read/update/delete (CRUD) data from a database

- INSERT: Insert new rows into a table
- SELECT: The SELECT command is used to get data from a database
- UPDATE: Update existing rows in a table
- DELETE: Delete rows from a table
- (bonus) CREATE: Make new tables/views/indexes

Example DB

Student

ID	Name	Age	Gender
1	Bart S.	10	M
2	Lisa S.	8	F
3	Jim F.	13	M
4	Joan B.	15	F

Enrollment

StudentID	SchoolID
1	1
2	1
3	2
4	3

School

ID	Name	Level
1	Springfield Elementary	E
2	Brook Middle	M
3	Springbrook High	H
4	Simpson Univ	U

SQL by Example – Select

Student

ID	Name	Age	Gender
1	Bart S.	10	M
2	Lisa S.	8	F
3	Jim F.	13	M
4	Joan B.	15	F

```
SELECT *
FROM Student
WHERE age > 12
```

ID	Name	Age	Gender
3	Jim F.	13	M
4	Joan B.	15	F

An SQL query walks into a bar and sees two tables.
He walks up to them and says "Can I join you?"

https://lol.browserling.com/tables.png

A more interesting select

Student

ID	Name	Age	Gender
1	Bart S.	10	M
2	Lisa S.	8	F
3	Jim F.	13	M
4	Joan B.	15	F

Enrollment

StudentID	SchoolID
1	1
2	1
3	2
4	3

School

ID	Name	Level
1	Springfield Elementary	E
2	Brook Middle	M
3	Springbrook High	H
4	Simpson Univ	U

Let's say we want to find **all students from Springfield Elementary.** The student table doesn't list the school. We have to use the enrollment table. Will this take two steps?

A more interesting select

Student

ID	Name	Age	Gender
1	Bart S.	10	M
2	Lisa S.	8	F
3	Jim F.	13	M
4	Joan B.	15	F

Enrollment

StudentID	SchoolID
1	1
2	1
3	2
4	3

School

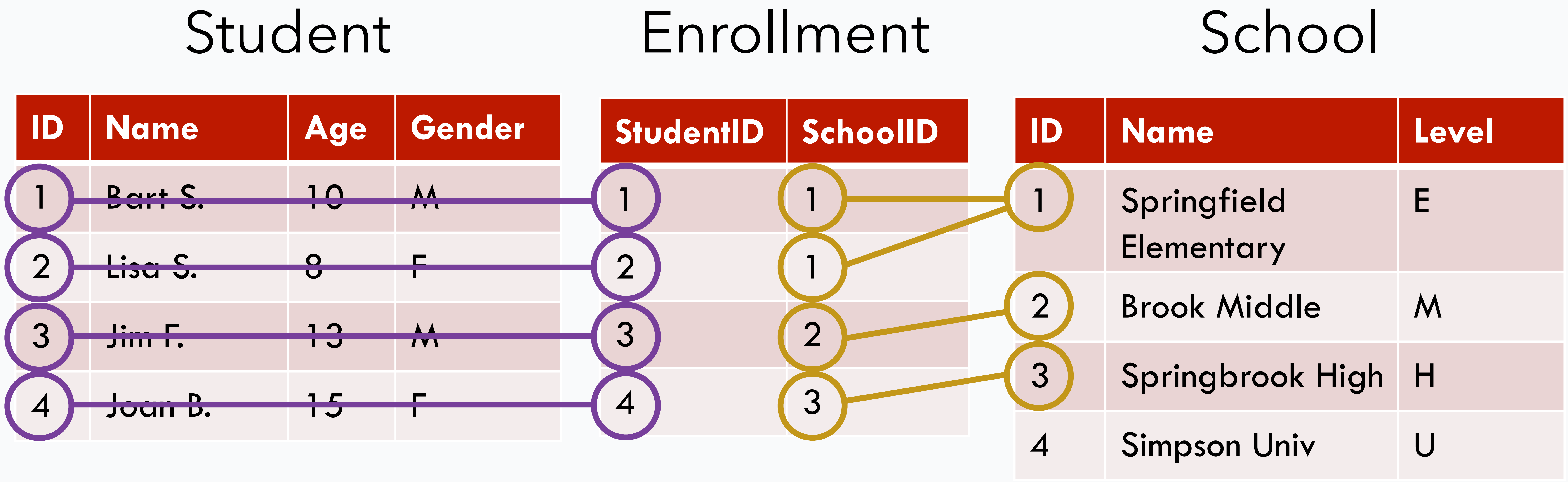
ID	Name	Level
1	Springfield Elementary	E
2	Brook Middle	M
3	Springbrook High	H
4	Simpson Univ	U

In fact, we can find all the students from Springfield Elementary (ID: 1) in one SQL statement using a JOIN

A SQL JOIN is used to combine rows from two or more tables, based on a common field between them.

Can you visualize it?

A more interesting select



Student

ID	Name	Age	Gender
1	Bart S.	10	M
2	Lisa S.	8	F
3	Jim F.	13	M
4	Joan B.	15	F

Enrollment

StudentID	SchoolID
1	1
2	1
3	2
4	3

School

ID	Name	Level
1	Springfield Elementary	E
2	Brook Middle	M
3	Springbrook High	H
4	Simpson Univ	U

In fact, we can find all the students from Springfield Elementary (ID: 1) in one SQL statement using a JOIN. A SQL JOIN is used to combine rows from two or more tables, based on a common field between them. Can you visualize it?

SQL Joining

If we joined the **Student** and **School** tables using the data in the Enrollment table, here is how it could look:

Student ID	Name	Age	Gender	School ID	School Name	Level
1	Bart S.	10	M	1	Springfield Elementary	E
2	Lisa S.	8	F	1	Springfield Elementary	E
3	Jim F.	13	M	2	Brook Middle	M
4	Joan B.	15	F	3	Springbrook High	H

```
SELECT *
FROM
    Student INNER JOIN Enrollment ON Student.id = Enrollment.StudentID
    INNER JOIN School ON Enrollment.SchoolID = School.id
```

SQL Joining

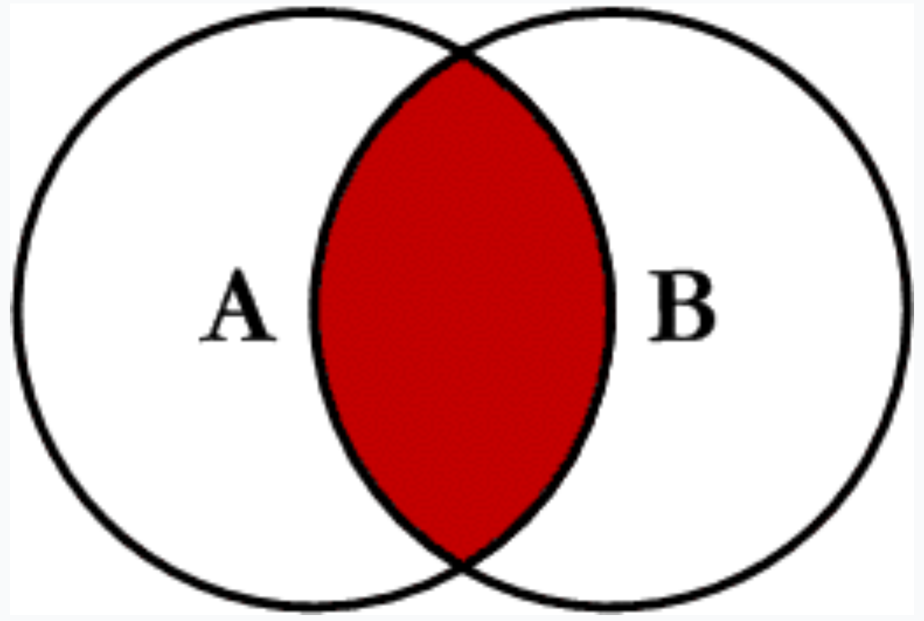
If we joined the **Student** and **School** tables using the data in the Enrollment table, here is how it could look:

Student ID	Name	Age	Gender	School ID	School Name	Level
1	Bart S.	10	M	1	Springfield Elementary	E
2	Lisa S.	8	F	1	Springfield Elementary	E
3	Jim F.	13	M	2	Brook Middle	M
4	Joan B.	15	F	3	Springbrook High	H

```
SELECT *
FROM
    Student INNER JOIN Enrollment ON Student.id = Enrollment.StudentID
    INNER JOIN School ON Enrollment.SchoolID = School.id
WHERE Enrollment.SchoolID = 1
```

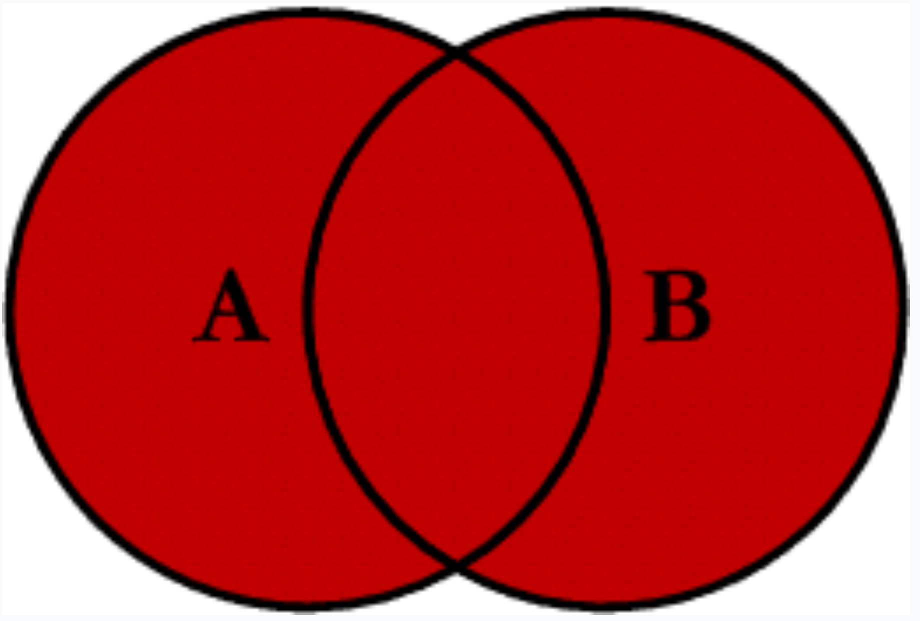
Inner Join

```
SELECT *
FROM A
INNER JOIN B
ON A.Key = B.Key
```



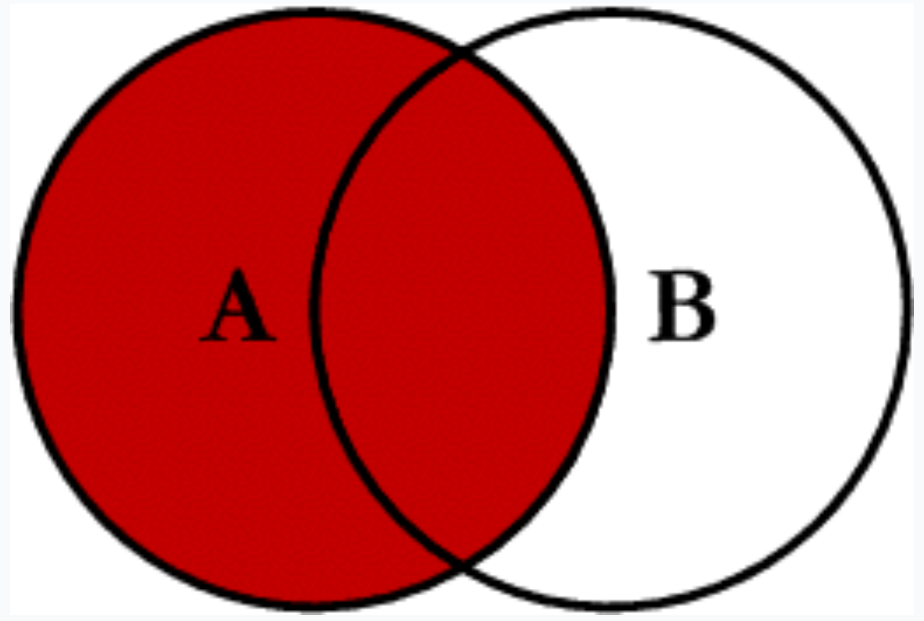
Outer Join

```
SELECT *
FROM A
FULL OUTER JOIN B
ON A.Key = B.Key
```



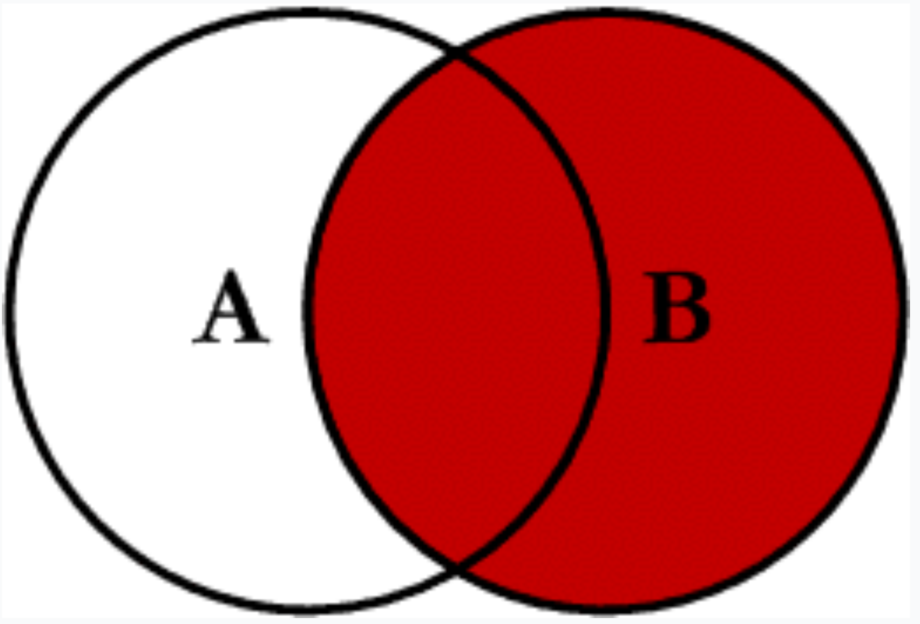
Left Join

```
SELECT *
FROM A
LEFT JOIN B
ON A.Key = B.Key
```



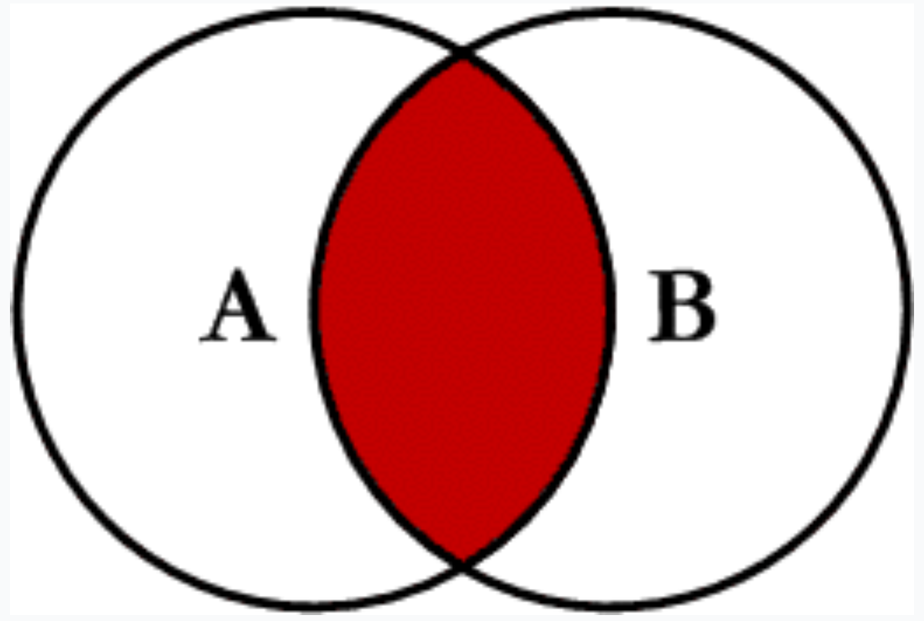
Right Join

```
SELECT *
FROM A
RIGHT JOIN B
ON A.Key = B.Key
```



http://www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins

Inner Join



```
SELECT pets.name, owners.name
FROM owners
INNER JOIN pets
ON pets.OwnerID = owners.ID
```

OWNERS

ID	name
1	Geordi
2	Janeway
3	Data
4	Spock

PETS

ID	ownerID	type	name
1	4	Monkey	Mittens
2	null	Lizard	Carol
3	1	Dog	Rufus
4	2	Cat	Fireball

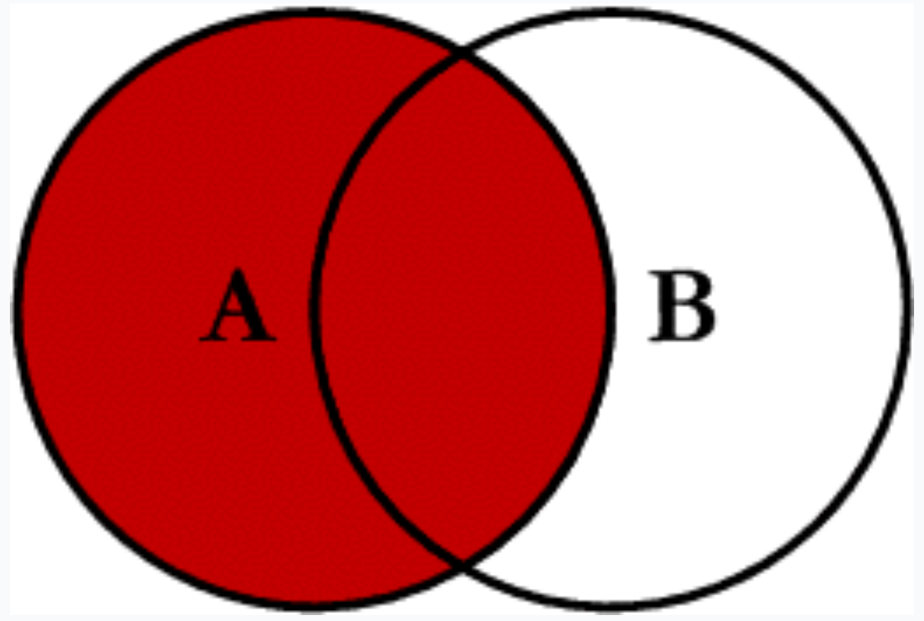
pets.name	owners.name
Mittens	Spock
Rufus	Geordi
Fireball	Janeway

PETS

ID	ownerID	type	name
1	4	Monkey	Mittens
2	null	Lizard	Carol
3	1	Dog	Rufus
4	2	Cat	Fireball

pets.name	owners.name
Mittens	Spock
Rufus	Geordi
Fireball	Janeway
null	Data ←

Left Join



```
SELECT pets.name, owners.name
FROM owners
LEFT JOIN pets
ON pets.OwnerID = owners.ID
```

OWNERS

ID	name
1	Geordi
2	Janeway
3	Data
4	Spock

pets.name	owners.name
Mittens	Spock
Carol	null
Rufus	Geordi
Fireball	Janeway

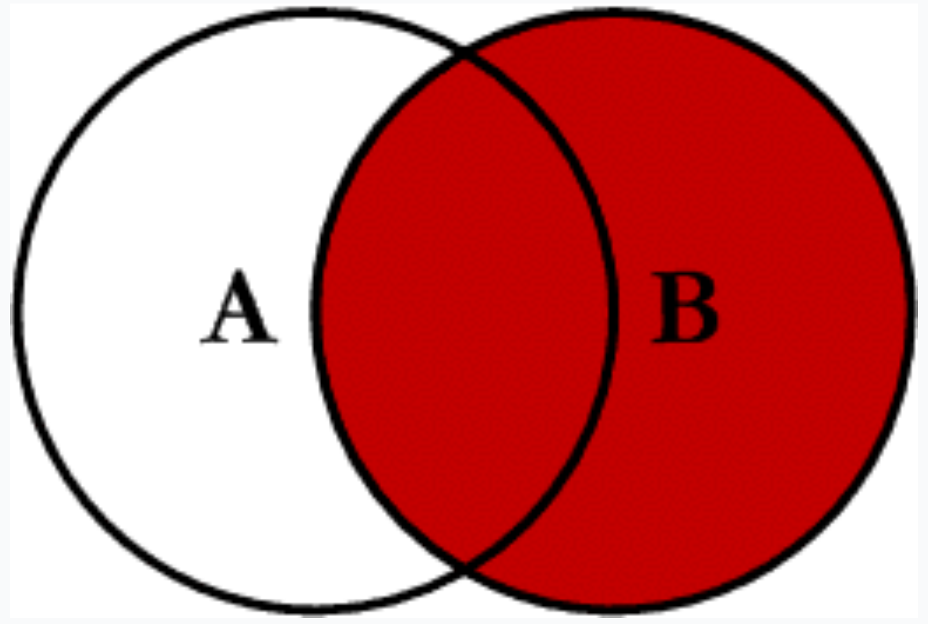
PETS

ID	ownerID	type	name
1	4	Monkey	Mittens
2	null	Lizard	Carol
3	1	Dog	Rufus
4	2	Cat	Fireball

OWNERS

ID	name
1	Geordi
2	Janeway
3	Data
4	Spock

Right Join

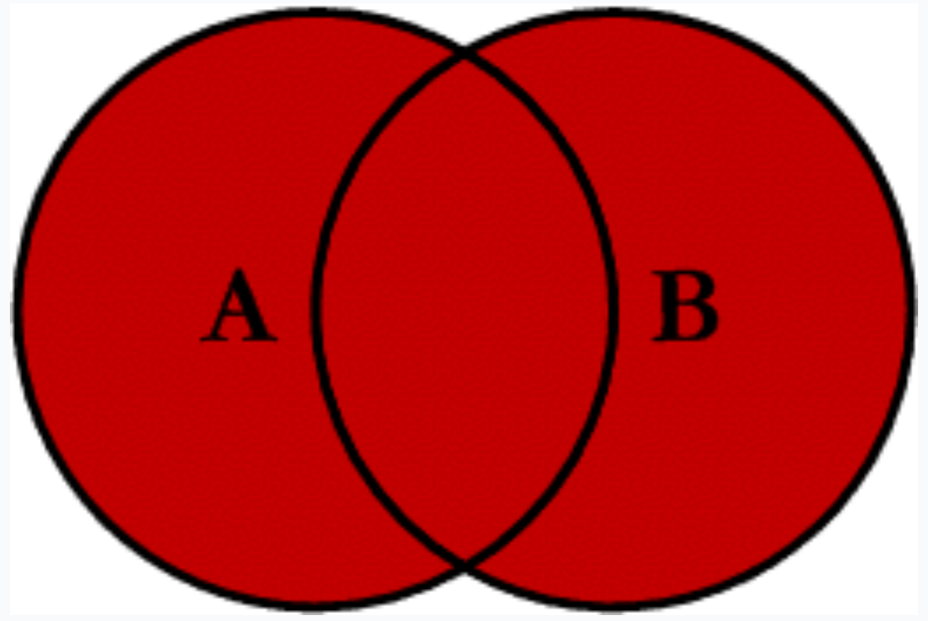


```
SELECT pets.name, owners.name
FROM owners
RIGHT JOIN pets
ON pets.OwnerID = owners.ID
```

Outer Join

OWNERS

ID	name
1	Geordi
2	Janeway
3	Data
4	Spock



```
SELECT pets.name, owners.name
FROM owners
FULL OUTER JOIN pets
ON pets.OwnerID = owners.ID
```

PETS

ID	ownerID	type	name
1	4	Monkey	Mittens
2	null	Lizard	Carol
3	1	Dog	Rufus
4	2	Cat	Fireball

pets.name	owners.name
Mittens	Spock
Carol	null
Rufus	Geordi
Fireball	Janeway
null	Data

GRACE HOPPER ACADEMY

WORKSHOP