

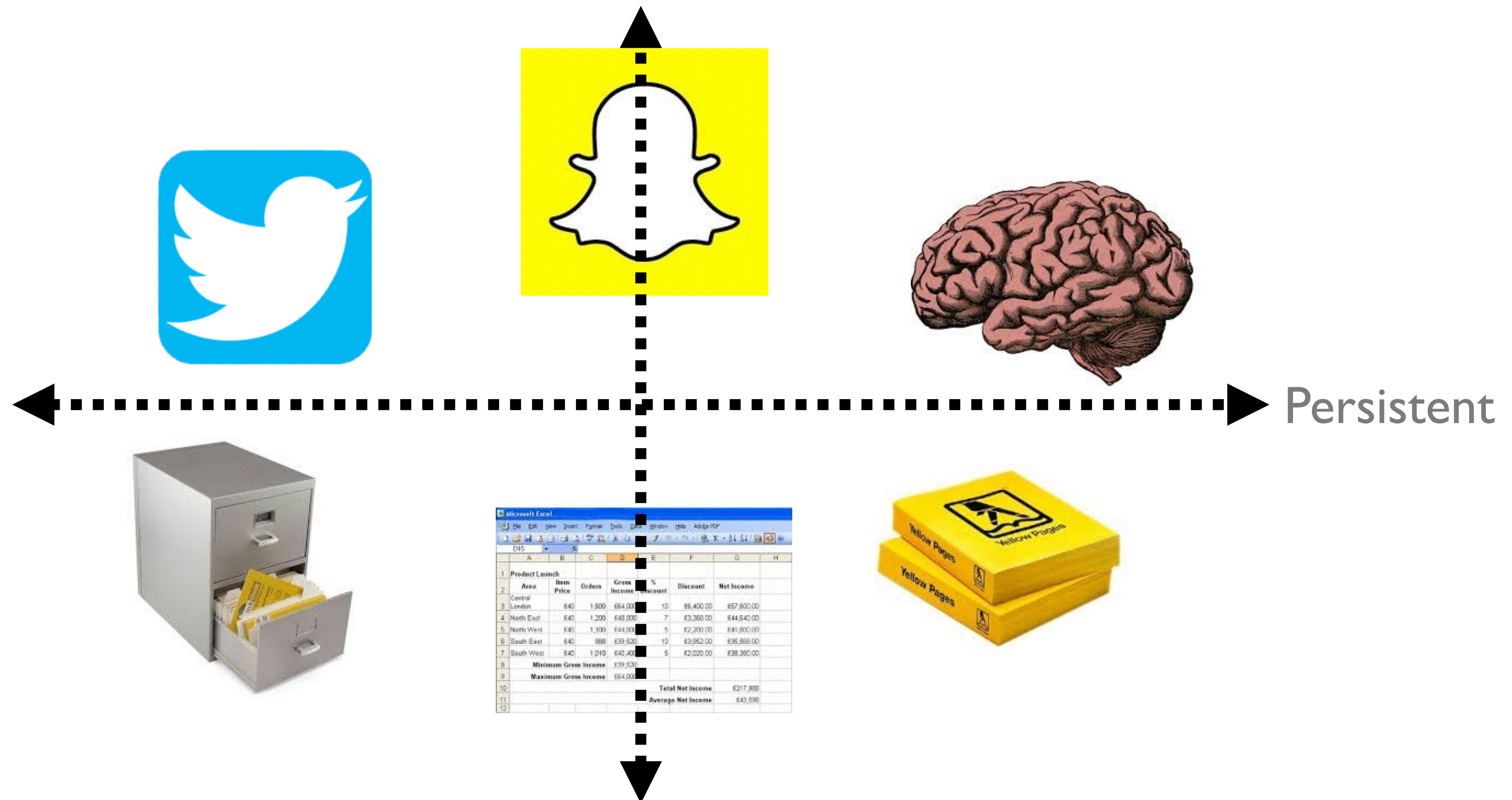
Intro to Databases

SQL

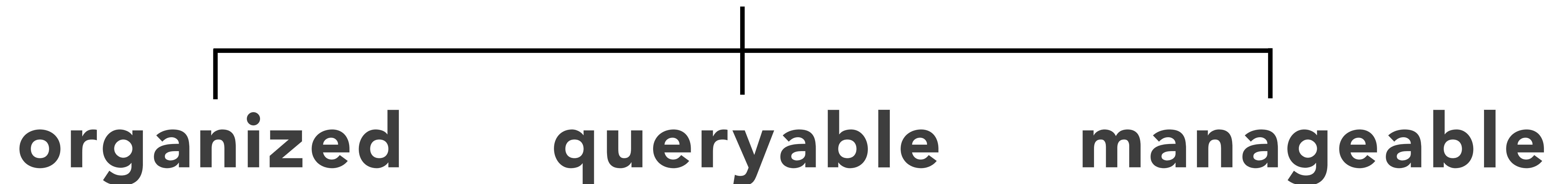
What is a database?

Things that hold info

Accessible

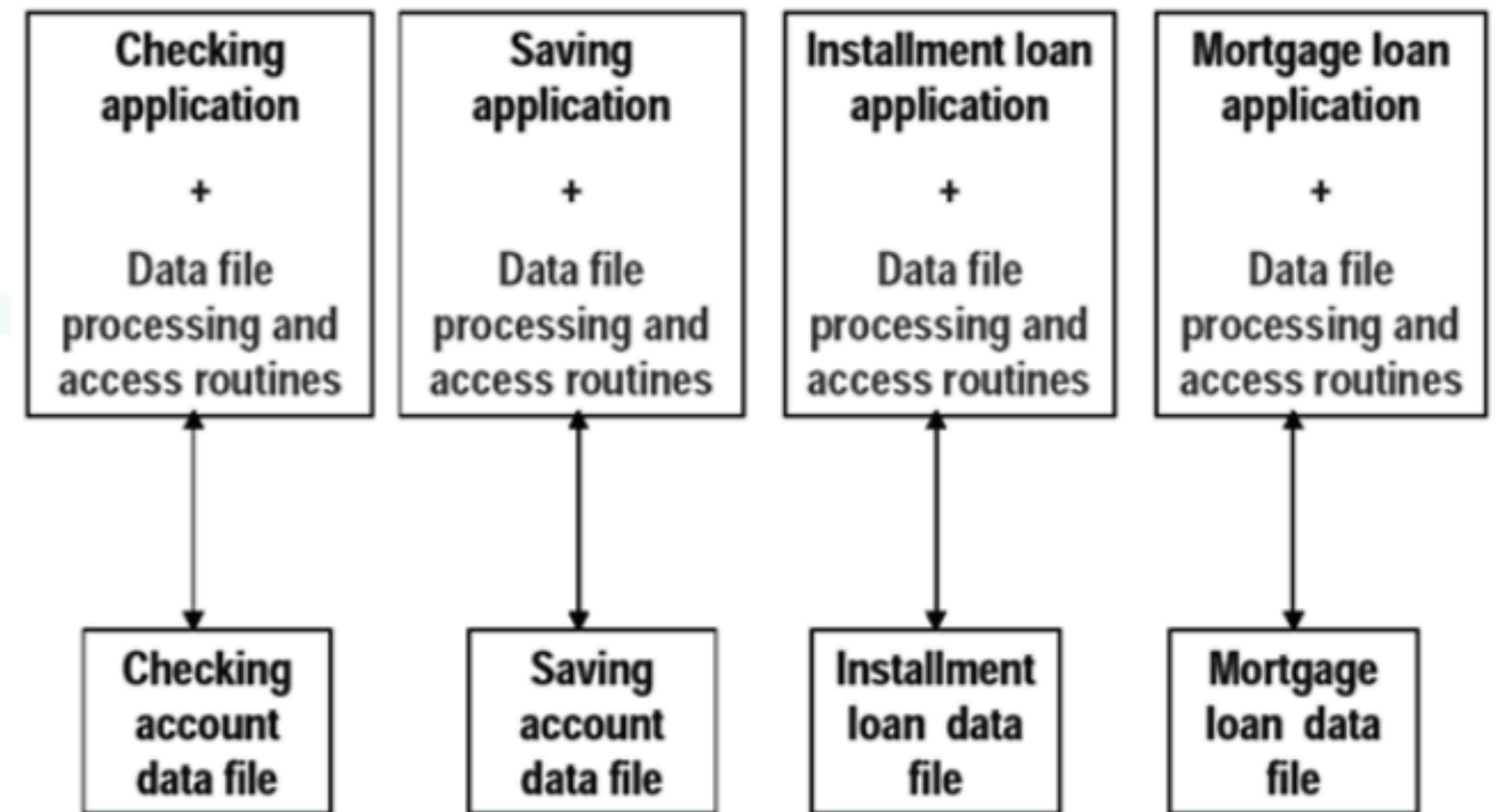


A database **persists** information
and is **accessible** via code

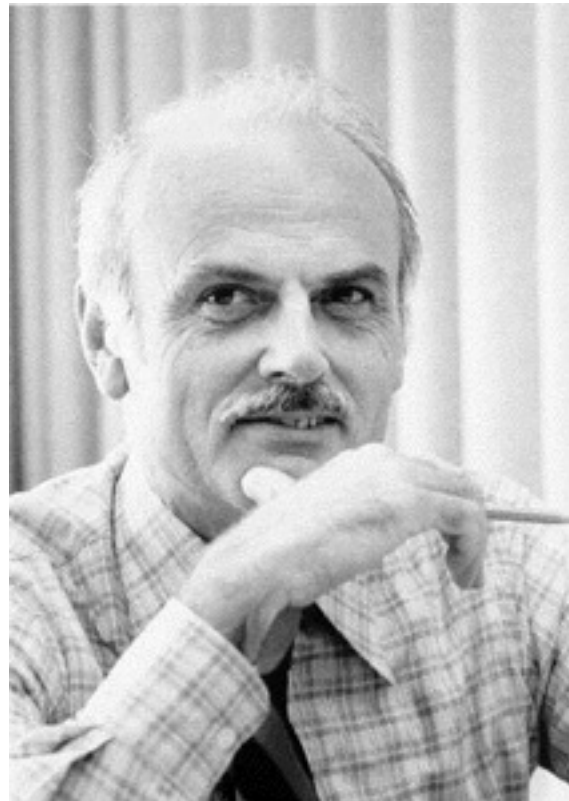


Before Relational DBs (ca. < 1970s)

- 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



Relational Databases & Logic



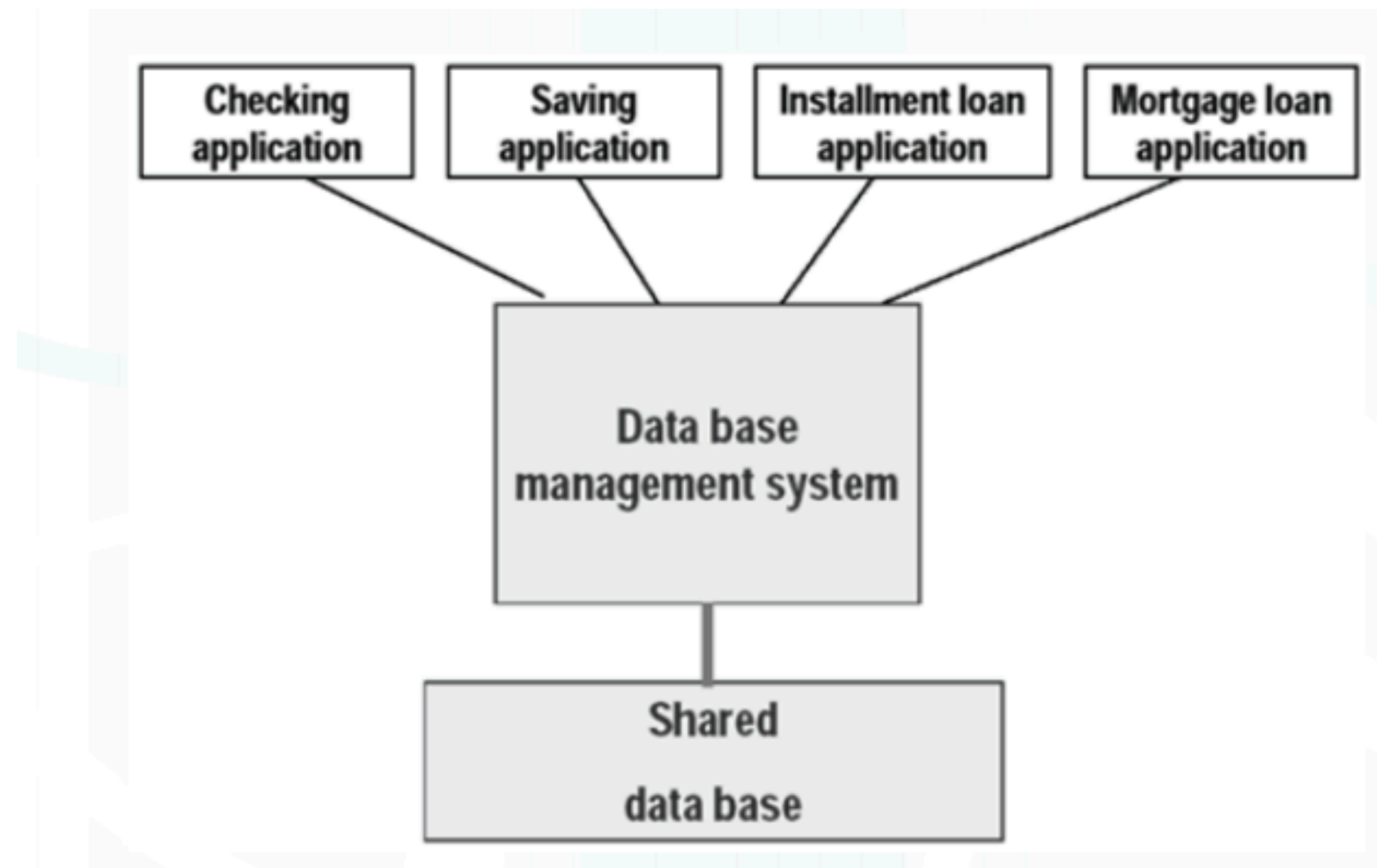
- 1969: Edgar Frank "Ted" Codd outlines *relational model* of data
- Wrote Alpha (never implemented) as a *query language*
- IBM slow to adopt his ideas
 - Competitors started to do so
 - IBM team formed without Codd, created Structured English Query Lang
- SEQUEL way better than what came before
 - 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



Database Management Systems (DBMS)



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

Appreciating Databases

- **Ubiquitous**
- **Standardized**
- **Complex / deep**
- **Powerful**
 - **database admins are**
 - Important!!!!
 - ...but also taken for granted until things break
 - Befriended/Contacted by business-side — the government (NSA) — for data mining

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**
 - e.g. {ID, Name, Age, Gender}
- **Content: actual data (a row)**
 - e.g. {1, "Bart S.", 10, "M"}
- **A schema is used to validate incoming content**

SQL is used to:

- ◉ INSERT: Insert new rows into a table
- ◉ SELECT: Get data from a database/table(s)
- ◉ UPDATE: Update existing rows in a table
- ◉ DELETE: Delete rows from a table
- ◉ (bonus) CREATE: Make new tables/views/indexes
- ◉ Create
- ◉ Read
- ◉ Update
- ◉ Delete
- ◉ (Create)



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
```

Data returned from query

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



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



A more interesting select

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?

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



A more interesting select

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?

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

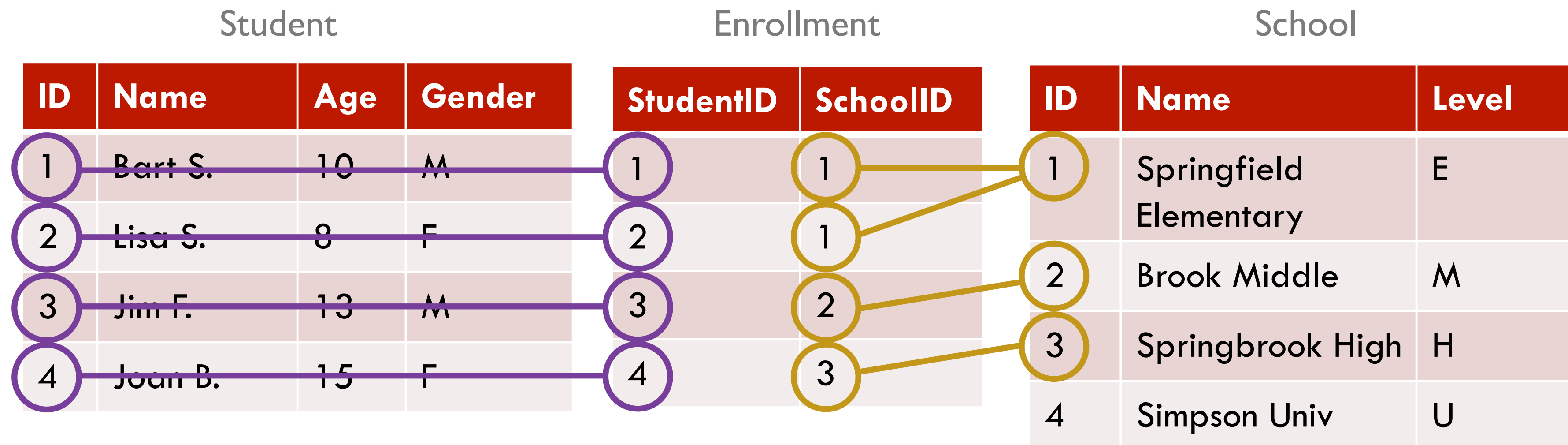
School

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



A more interesting select

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

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

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



SQL Joining

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

```
SELECT *  
FROM Student  
  INNER JOIN Enrollment ON Student.id = Enrollment.StudentID  
  INNER JOIN School ON Enrollment.SchoolID = School.id  
WHERE School.SchoolName = 'Springfield Elementary'
```

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



SQL Joining?

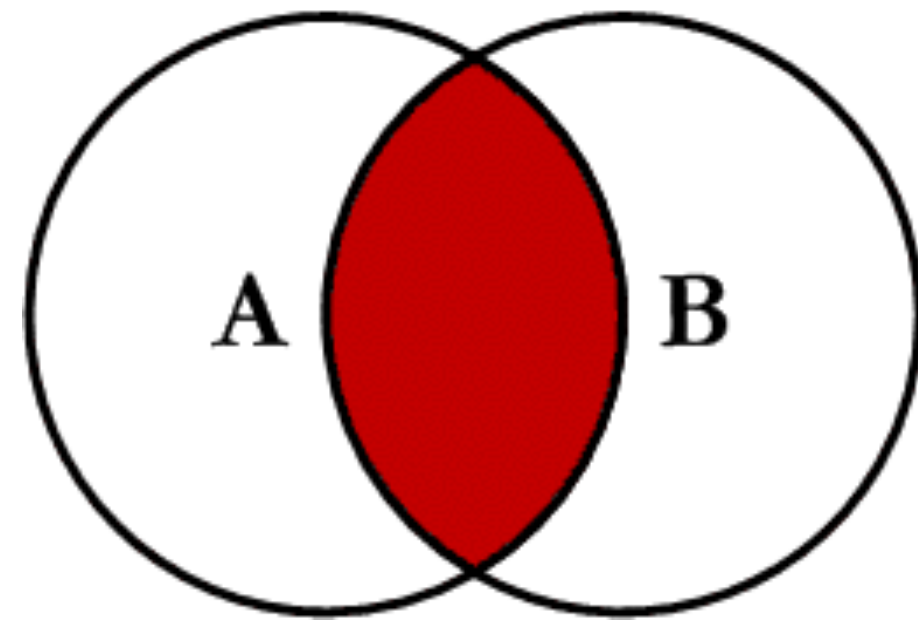
If we joined the **Student** and **School** tables using the data in the Enrollment table, here is how it could look

```
SELECT *  
FROM Student, Enrollment, School  
WHERE Student.id = Enrollment.StudentID  
      AND Enrollment.SchoolID = School.id  
      AND Enrollment.SchoolName = 'Springfield Elementary'
```

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

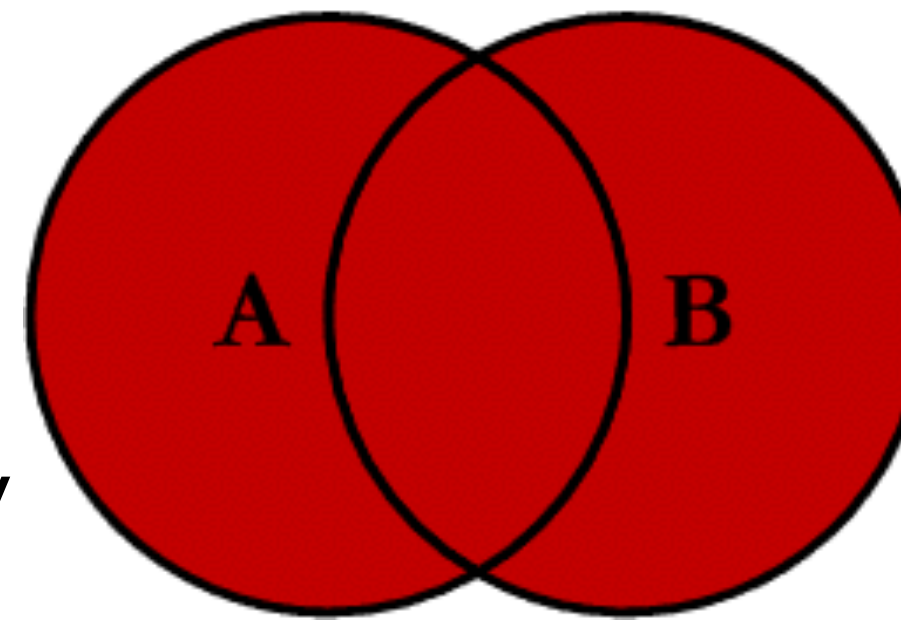


Inner Join



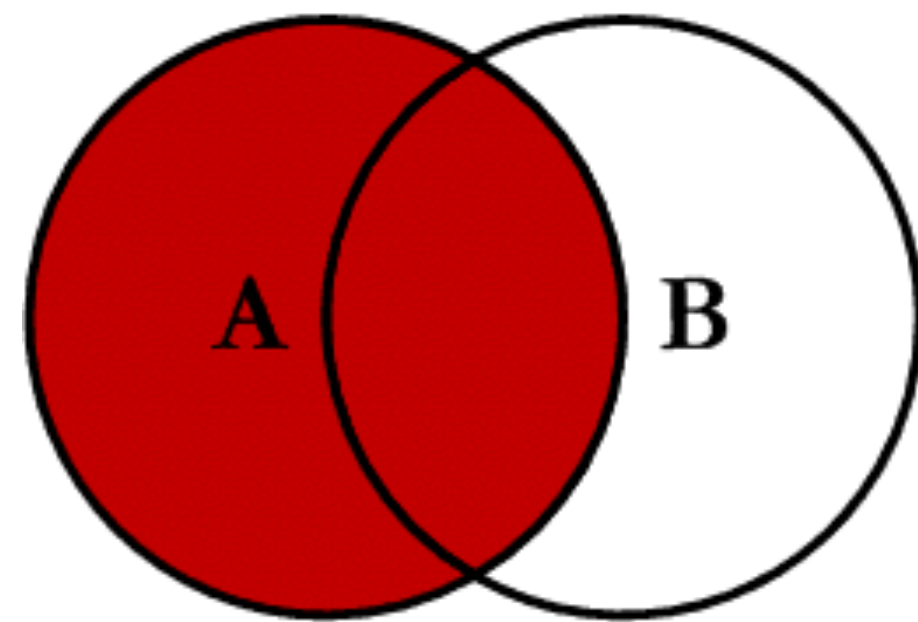
```
SELECT *  
FROM A  
INNER JOIN B  
ON A.Key = B.Key
```

Outer Join



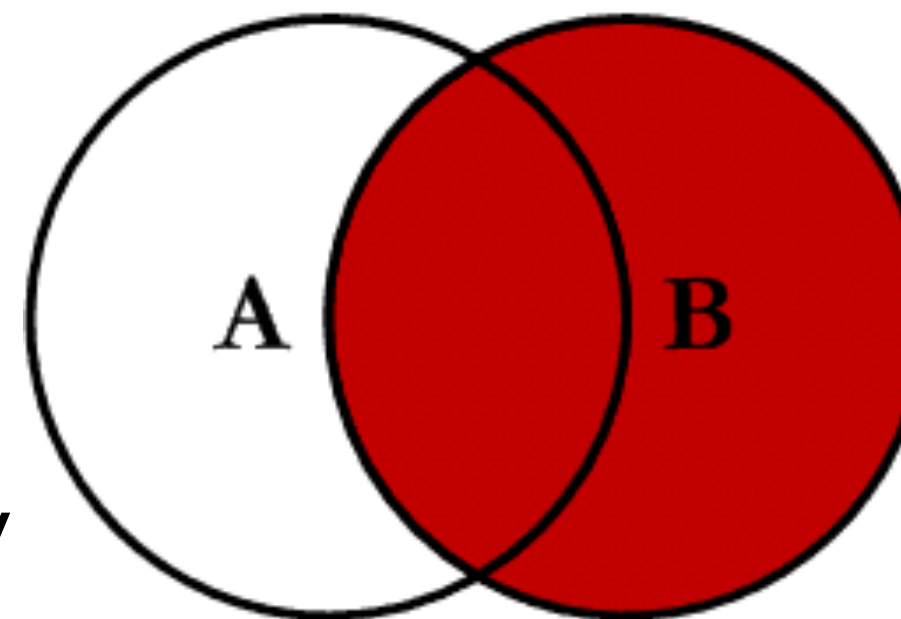
```
SELECT *  
FROM A  
FULL OUTER JOIN B  
ON A.Key = B.Key
```

Left Join



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

Right Join

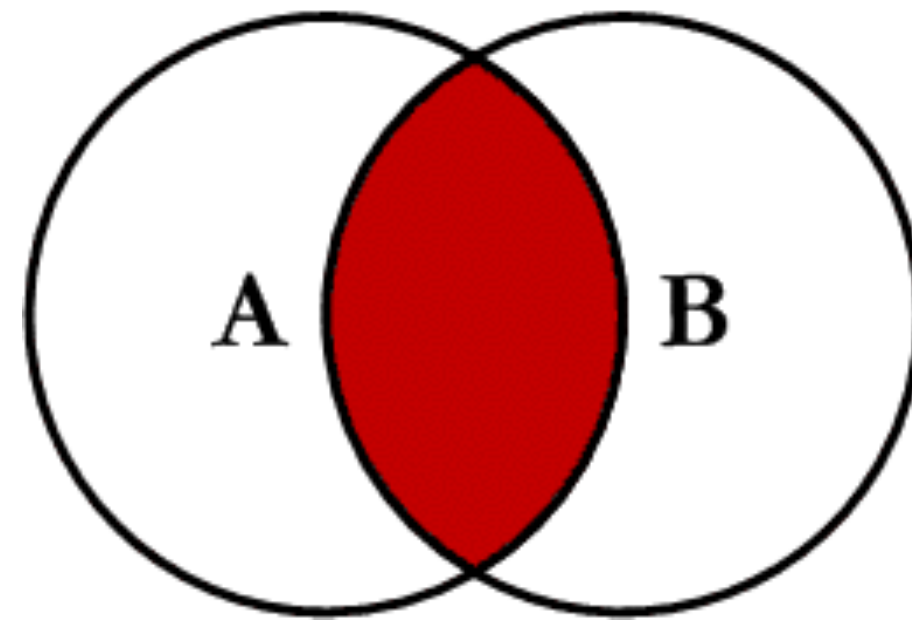


```
SELECT *  
FROM A  
RIGHT JOIN B  
ON A.Key = B.Key
```

<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	Spok

PETS

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

Result

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

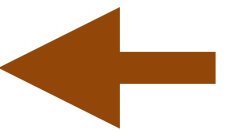


PETS

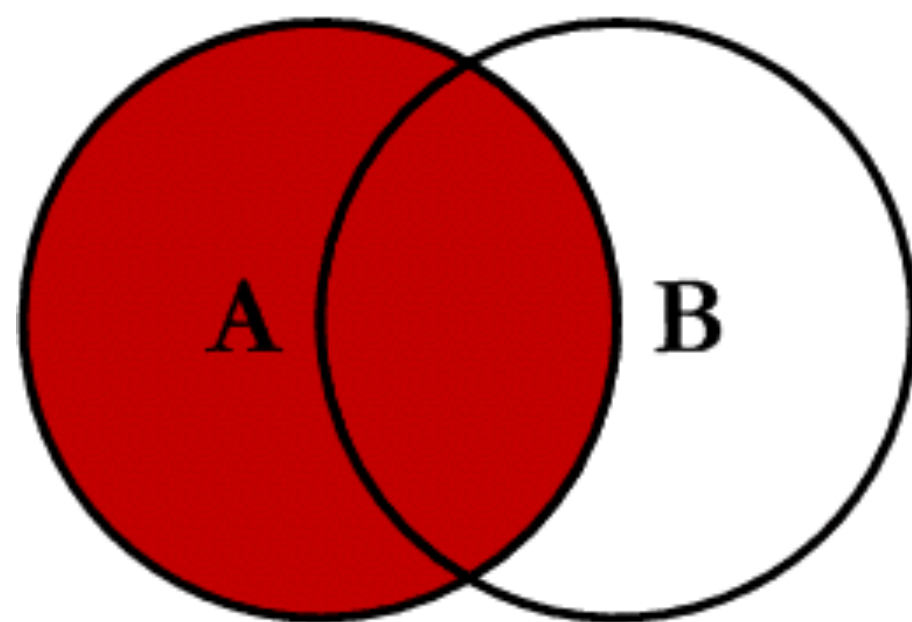
ID	ownerID	type	name
1	4	Monkey	Mittens
2	null	Lizard	Carol
3	1	Dog	Rufus
4	2	Cat	Fireball

Result

pets.name	owners.name
Mittens	Spok
Rufus	Geordi
Fireball	Janeway
null	Data



Left Join



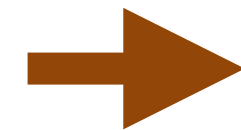
```
SELECT pets.name, owners.name  
FROM owners  
LEFT JOIN pets  
ON pets.OwnerID = owners.ID
```

OWNERS

ID	name
1	Geordi
2	Janeway
3	Data
4	Spok



Result



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

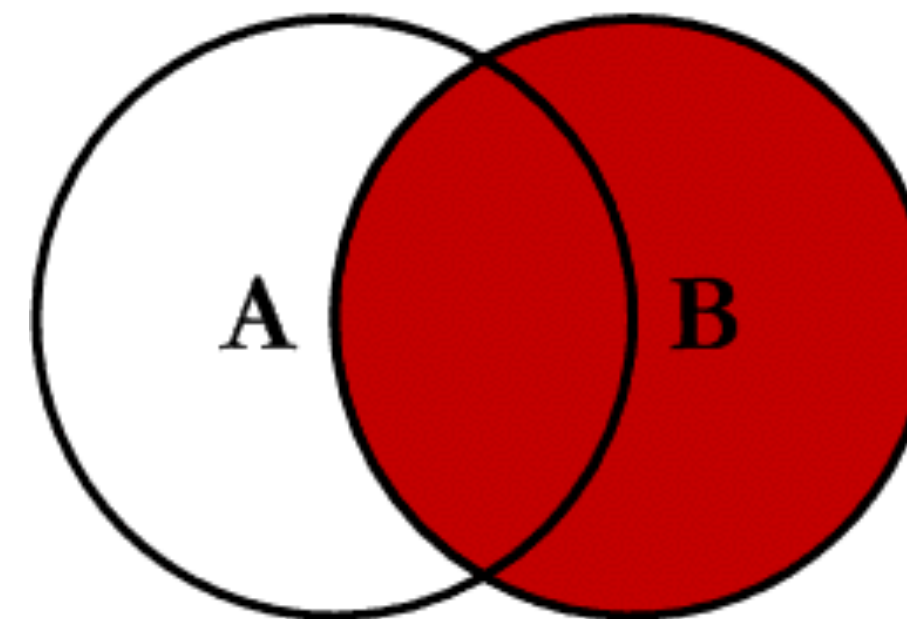
PETS

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	Spok

Right Join



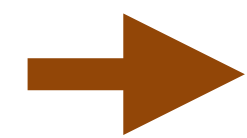
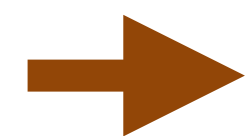
```
SELECT pets.name, owners.name  
FROM owners  
RIGHT JOIN pets  
ON pets.OwnerID = owners.ID
```




OWNERS

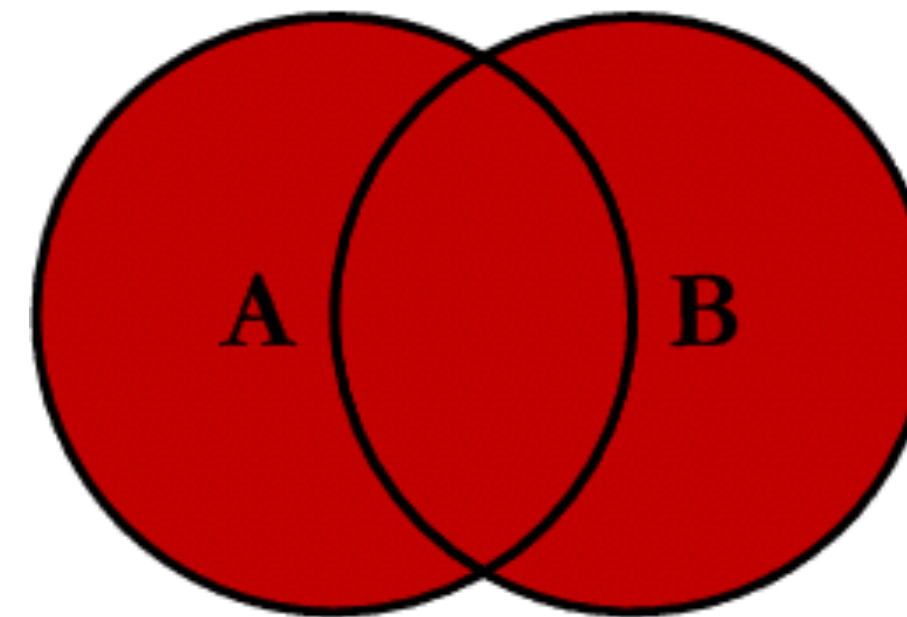
ID	name
1	Geordi
2	Janeway
3	Data
4	Spok

Result



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

Outer Join



```
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

WORKSHOP