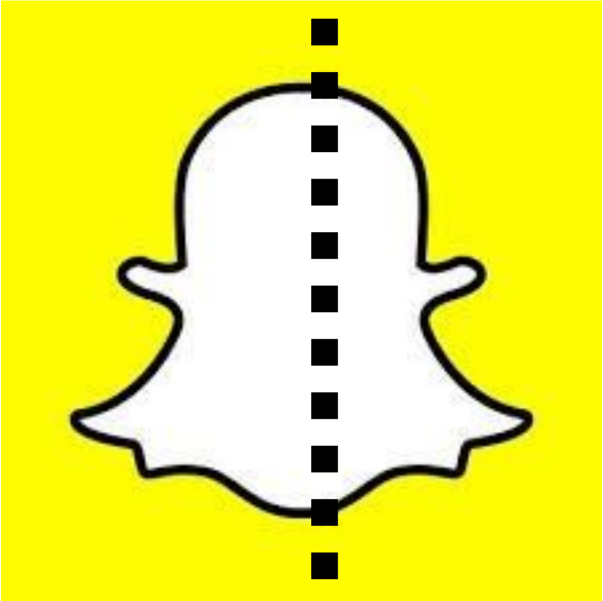
Intro to Databases

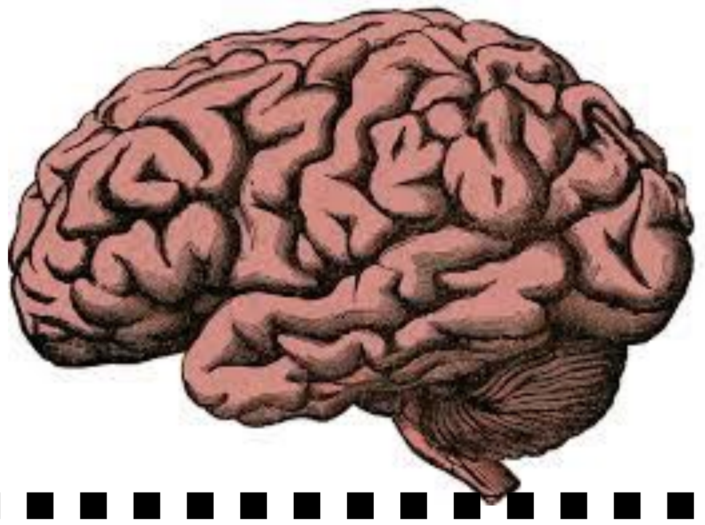
SQL

What is a database?

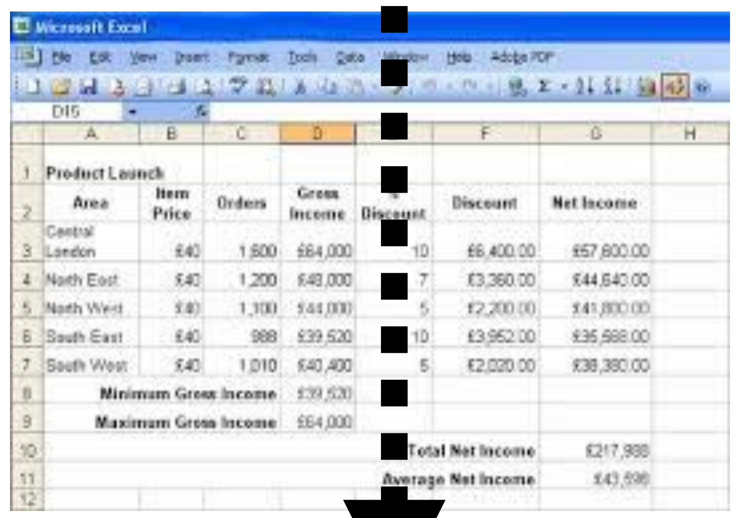
Things that hold info

Accessible



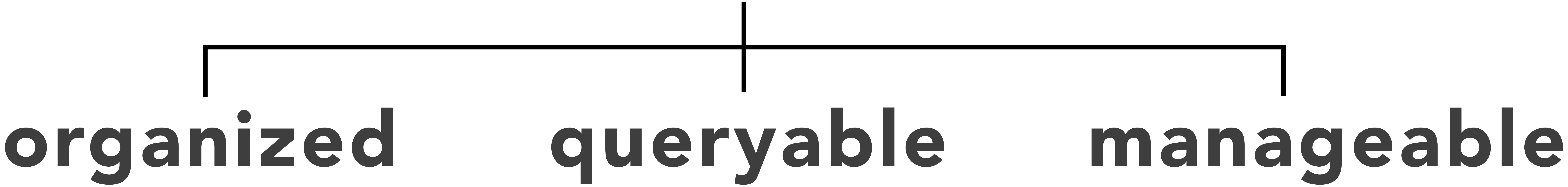


Persistent


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



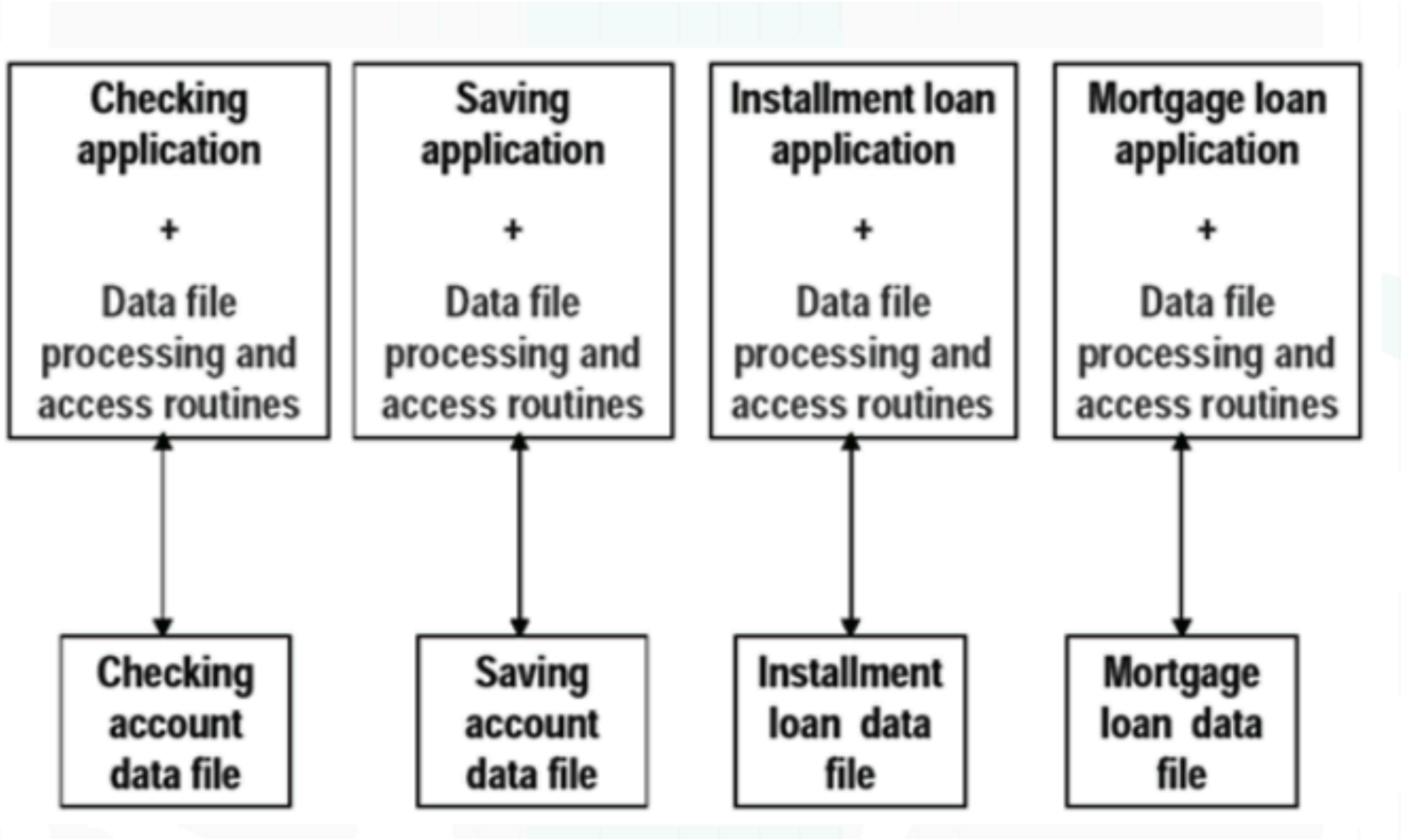
organized

queryable

manageable

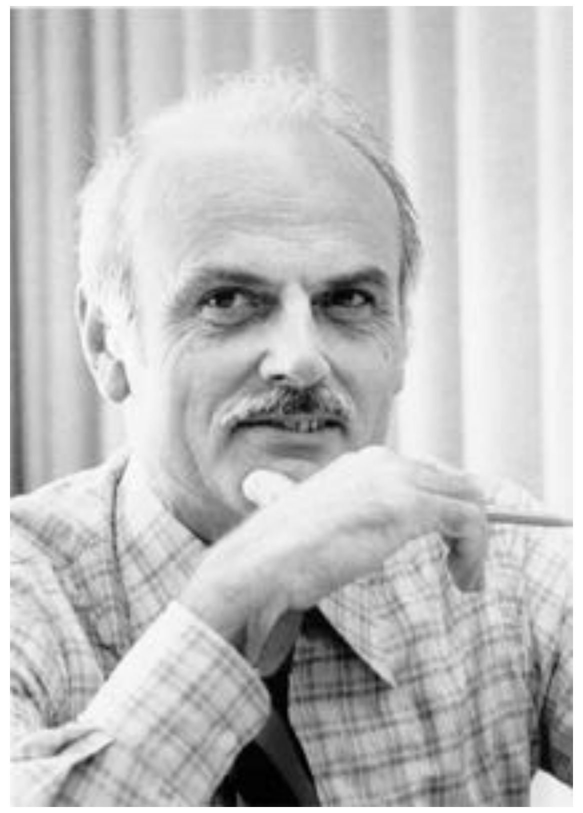
Before Relational DBs (ca. < 1970s)

- 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

Deadlock

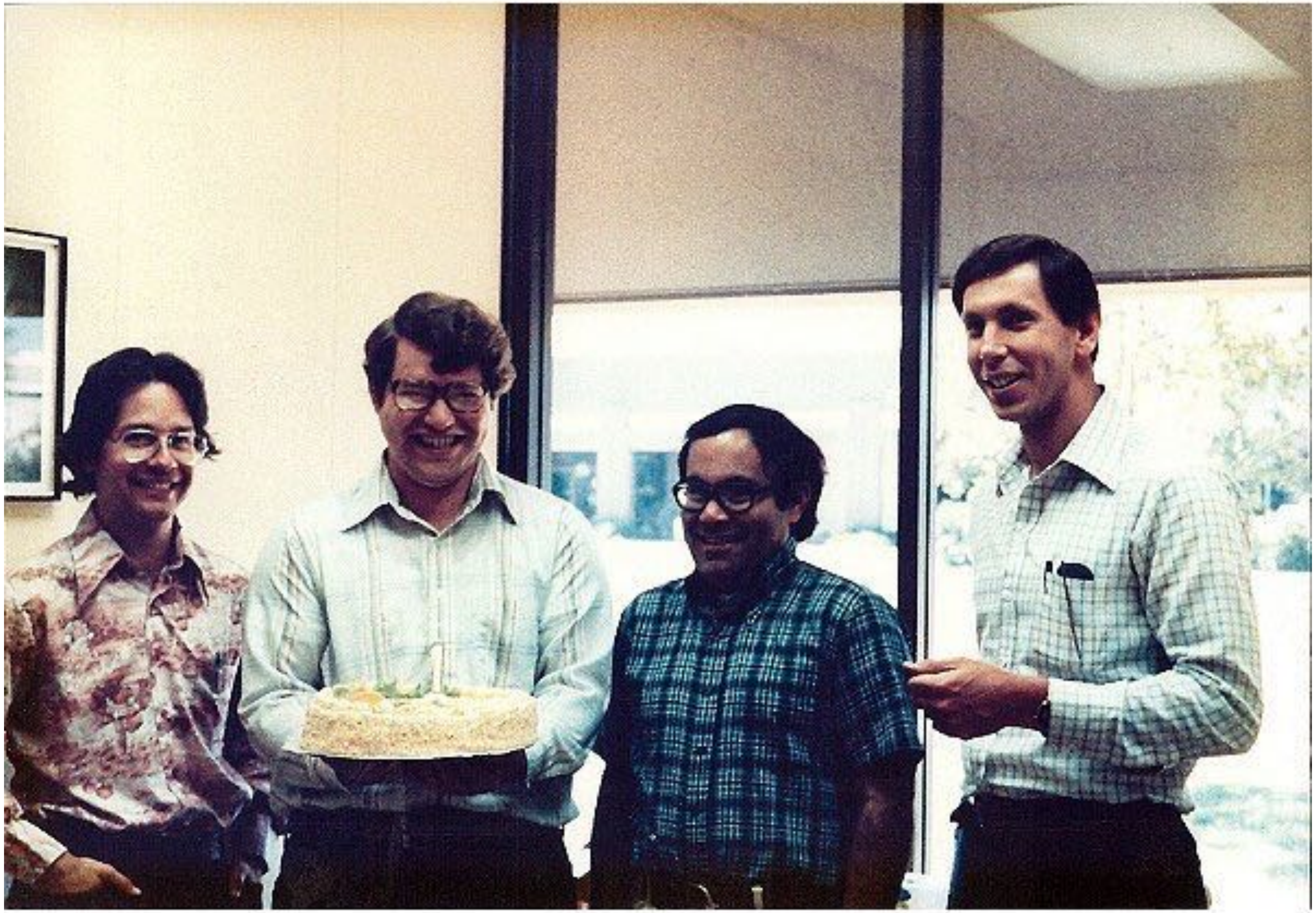
Relational Databases & Logic



- 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*

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

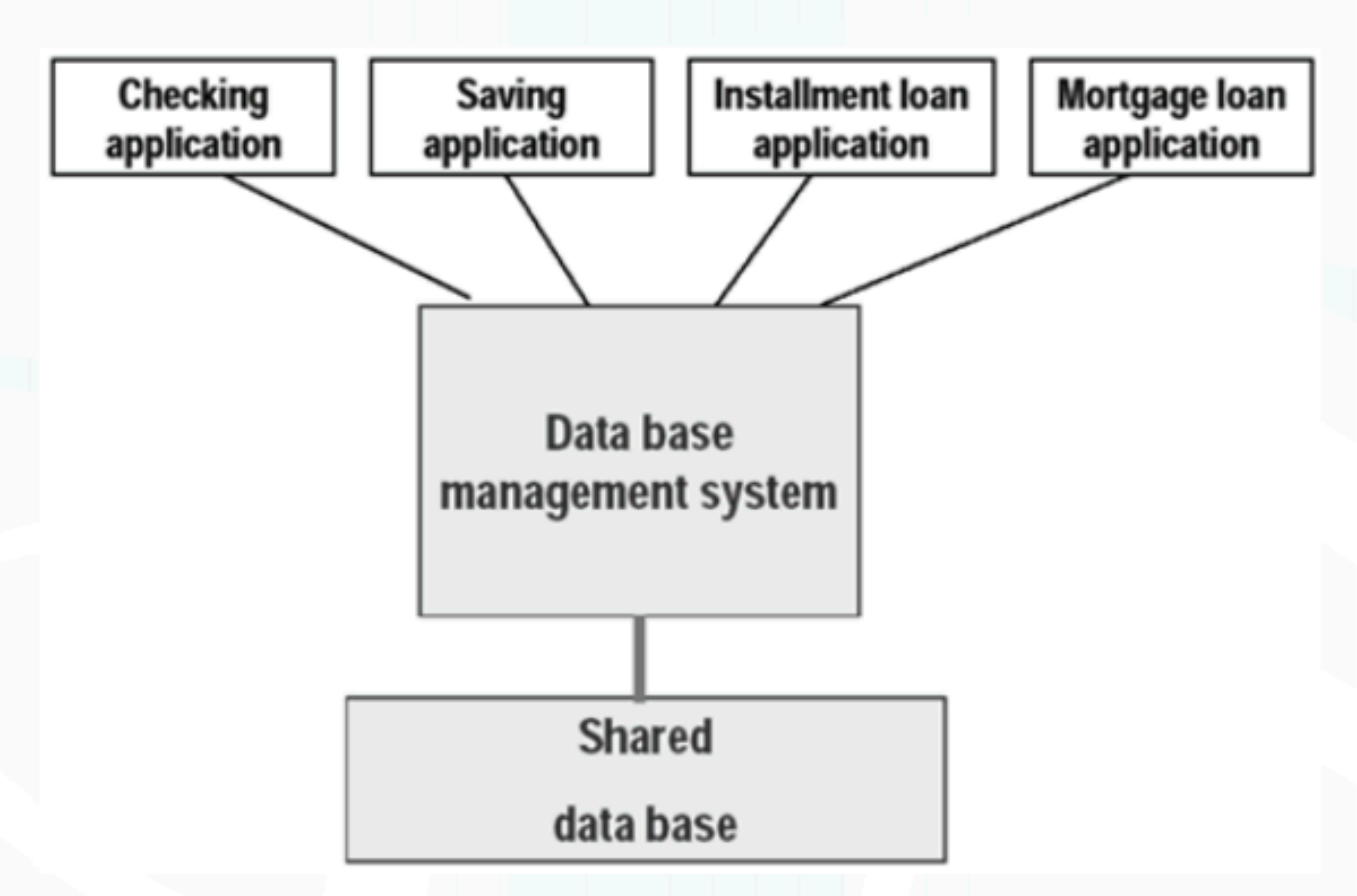
– E. F. CODD,
A RELATIONAL MODEL OF DATA FOR
LARGE SHARED DATA BANKS



Oracle

- Ed Oates
- Bruce Scott
- Bob Miner
- Larry Ellison

Relational Database Management Systems (RDBMS)


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

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 specify what answers a query should return, but not how the query is executed, or where/how the data is stored
  - the DBMS picks an execution strategy based on indexes, data, workload, etc.
- **Multi-user, Multi-Threaded**
  - Multiple process can access database at the same time

Definitions

- DBs are a collection of Tables (or *relations*)
- Tables have Columns (*attributes* / *fields*) that describe Rows (*instances* / *tuples*)
- Duplicate rows are not allowed
- Rows often have a primary key (unique identifier)

Table / Relation

	Column / Attribute / Field	Column / Attribute / Field	Column / Attribute / Field
	ID	**Name**	**Type**
Row / Tuple / Instance	1	Pikachu	lightning
Row / Tuple / Instance	2	Squirtle	water
Row / Tuple / Instance	3	Charmander	fire
Row / Tuple / Instance	4	Bulbasaur	grass

Schema and Content

- Schema: a table’s blueprint for data shape/formate
  - e.g. each instance will have ID, Name, Age, and Gender
- Content: the actual data (a row)
  - e.g. {1, “Bart Simpson”, 10, “M”}

- The *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

- **C**reate
- **R**ead
- **U**pdate
- **D**elete

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

Example SELECT statement

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

comic.browserling.com

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

A more interesting SELECT statement

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 SQL queries?

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

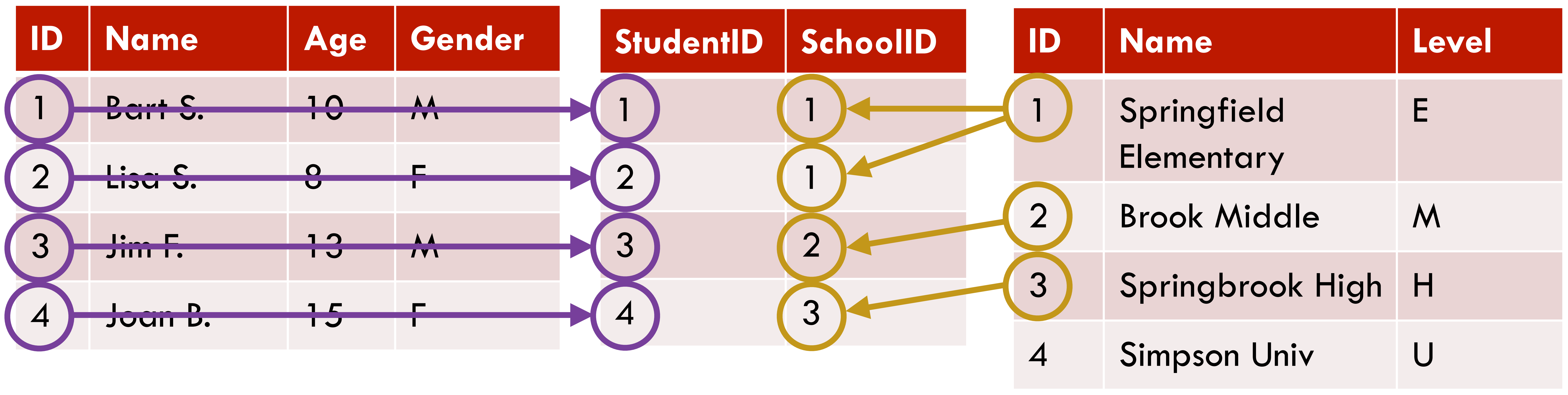
School

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

We can find **all 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

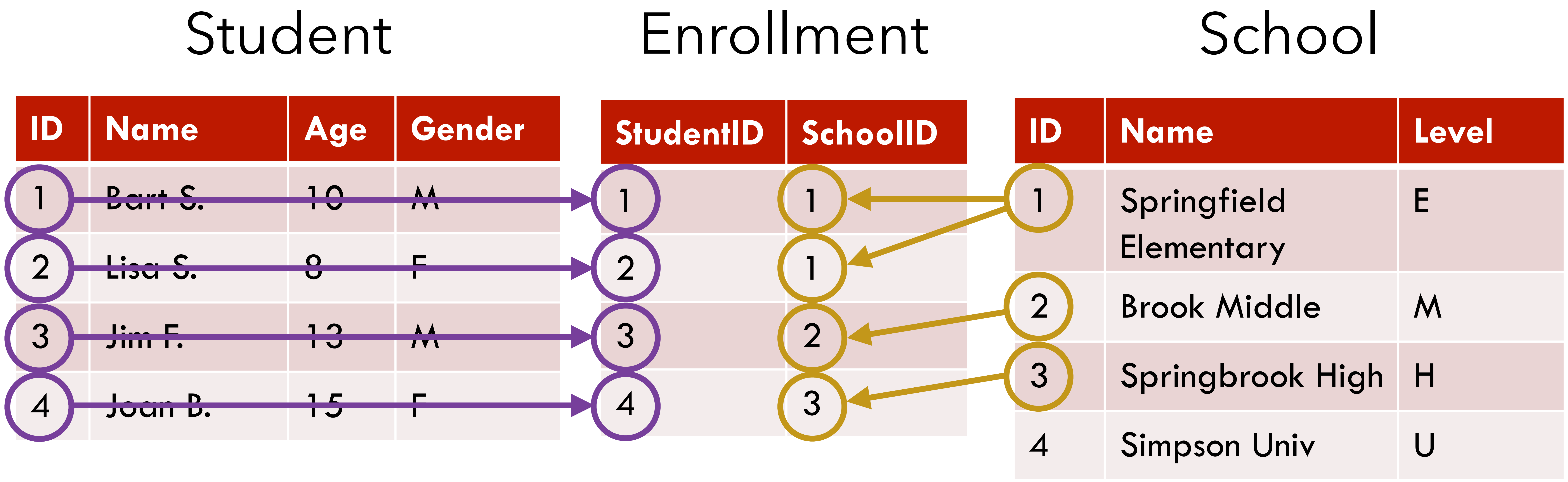
School

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

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

=

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

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

```
SELECT *
FROM Student INNER JOIN Enrollment ON Student.id = Enrollment.StudentId
         INNER JOIN School On Enrollment.SchoolID = School.ID
```

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

Step 1 - join the Student table and the Enrollment table…

```
SELECT *
FROM Student INNER JOIN Enrollment ON Student.id = Enrollment.StudentId
          INNER JOIN School On Enrollment.SchoolID = School.ID
```

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

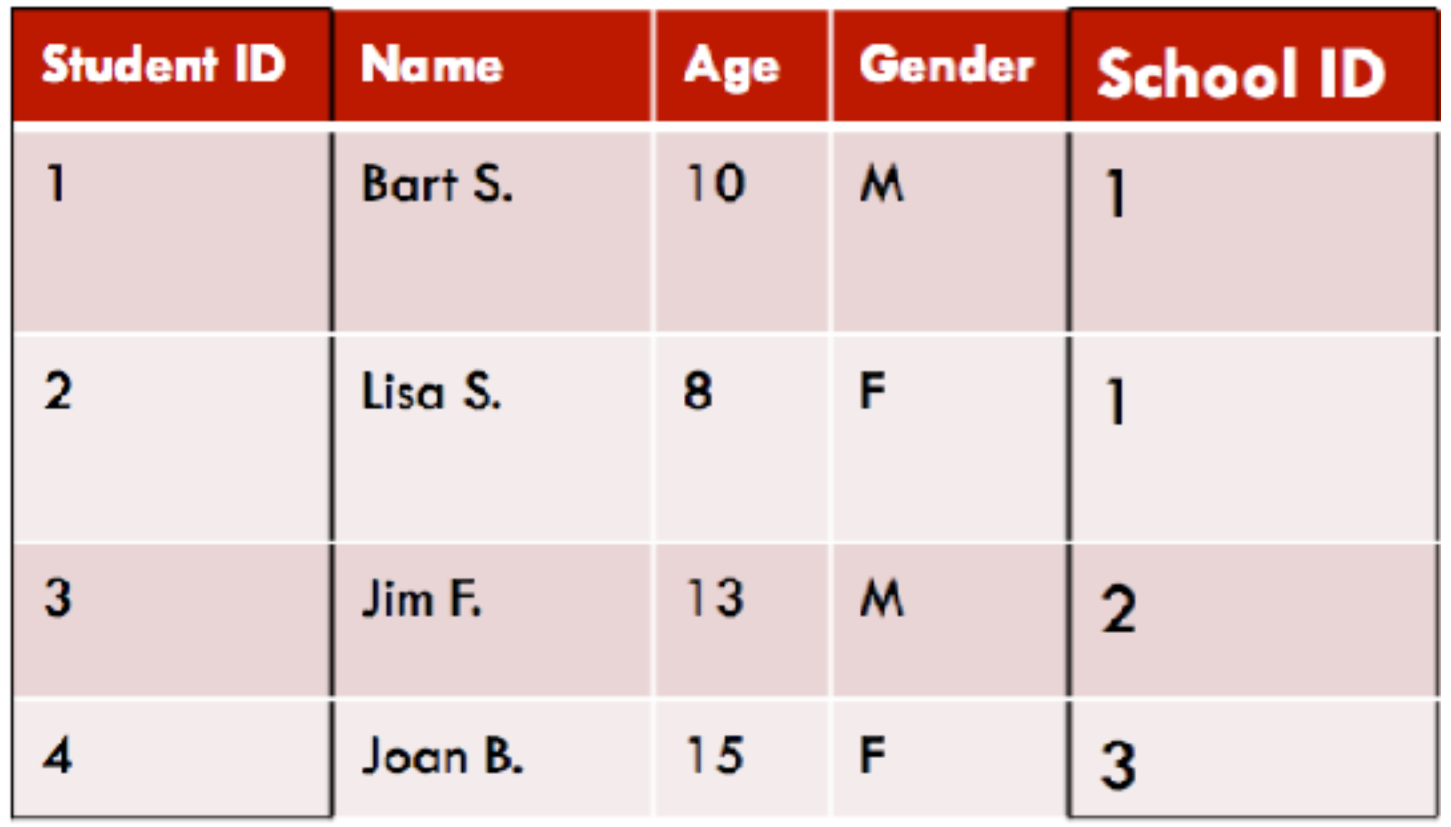
ID	Name	Level
1	Springfield Elementary	E
2	Brook Middle	M
3	Springbrook High	H
4	Simpson Univ	U

Step 1 - join the Student table and the Enrollment table… wherever the studentID in the enrollment table corresponds to an id in the Student table

```
SELECT *
FROM Student INNER JOIN Enrollment ON Student.id = Enrollment.StudentId
         INNER JOIN School On Enrollment.SchoolID = School.ID
```

=

Enrollment



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

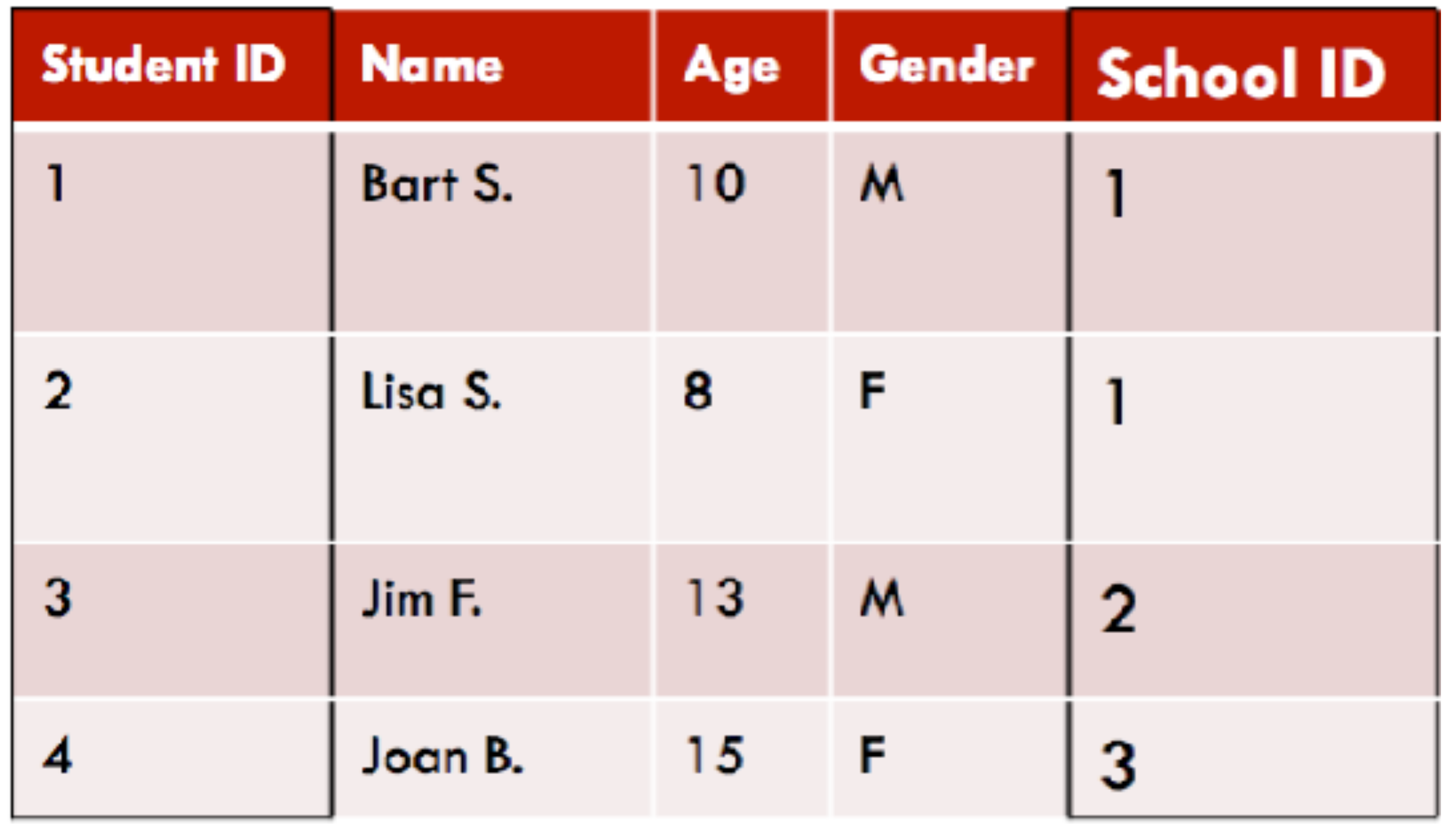
School

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

Step 1 - join the Student table and the Enrollment table… wherever the studentID in the enrollment table corresponds to an id in the Student table

```
SELECT *
FROM Student INNER JOIN Enrollment ON Student.id = Enrollment.StudentId
         INNER JOIN School On Enrollment.SchoolID = School.ID
```

Enrollment



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

+

School

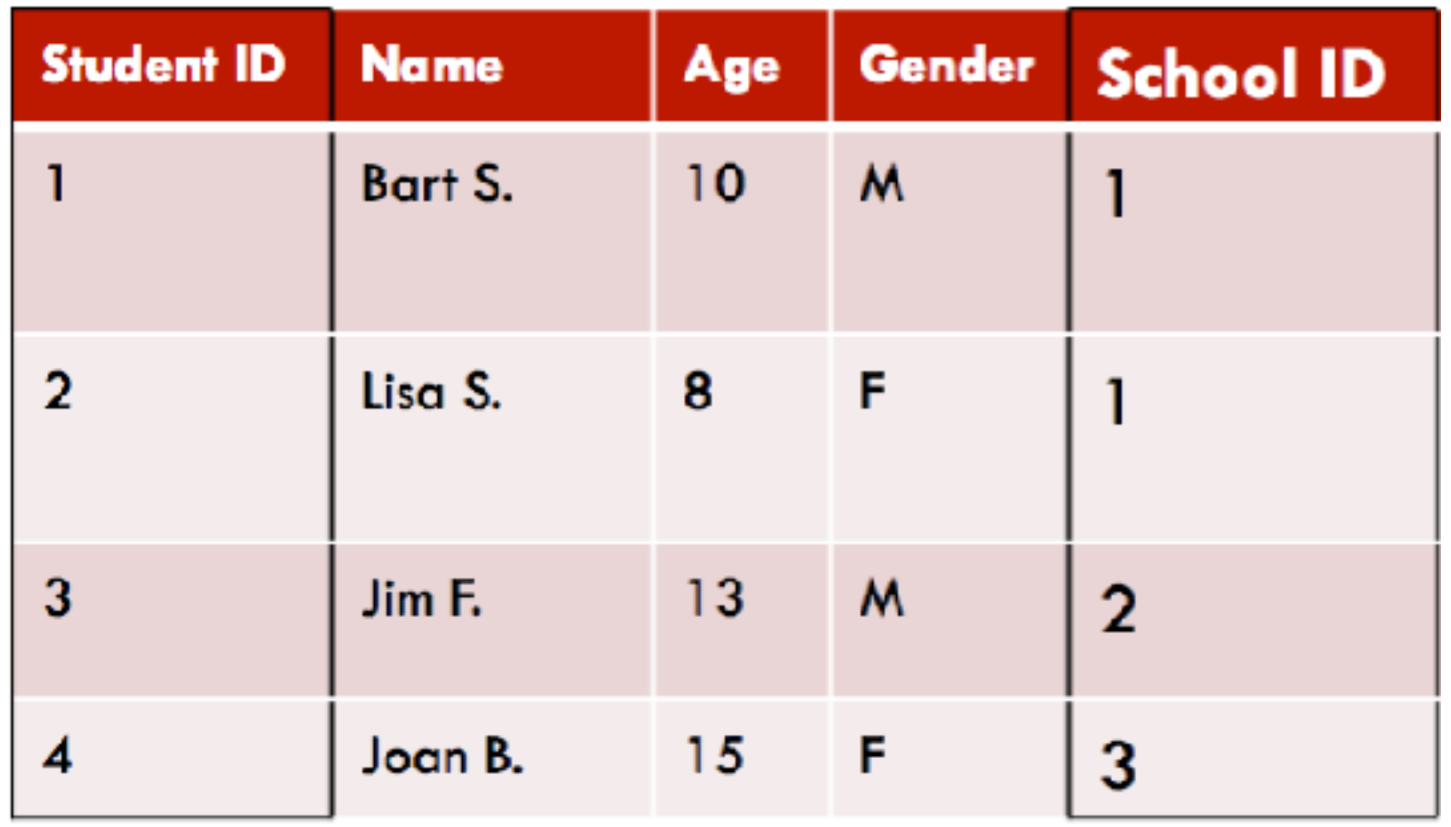
ID	Name	Level
1	Springfield Elementary	E
2	Brook Middle	M
3	Springbrook High	H
4	Simpson Univ	U

Step 1 - join the Student table and the Enrollment table… wherever the studentID in the enrollment table corresponds to an id in the Student table

Step 2 - join this new Enrollment table and the School table…

```
SELECT *
FROM Student INNER JOIN Enrollment ON Student.id = Enrollment.StudentId
        INNER JOIN School On Enrollment.SchoolID = School.ID
```

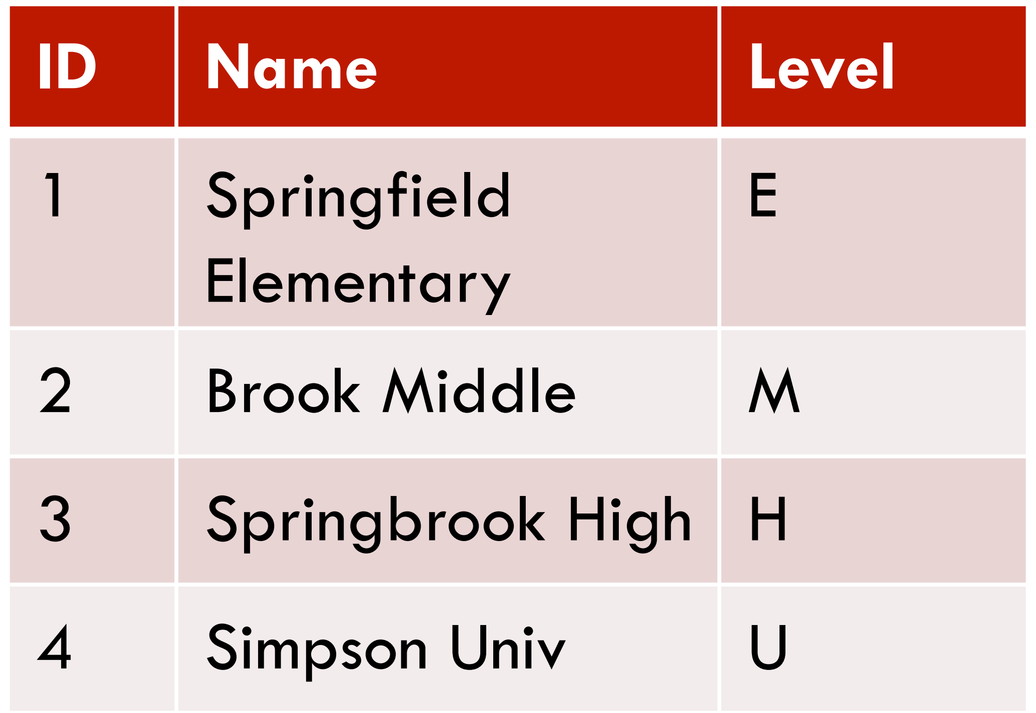
Enrollment



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

+

School



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

Step 1 - join the Student table and the Enrollment table… wherever the studentID in the enrollment table corresponds to an id in the Student table

Step 2 - join this new Enrollment table and the School table… wherever the SchoolID in the Enrollment table corresponds to an id in the School table

```
SELECT *
FROM Student INNER JOIN Enrollment ON Student.id = Enrollment.StudentId
         INNER JOIN School On Enrollment.SchoolID = School.ID
```

Join Table!

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

Lastly - we can search through this join table as normal!

```
SELECT *
FROM Student INNER JOIN Enrollment ON Student.id = Enrollment.StudentId
          INNER JOIN School On Enrollment.SchoolID = School.ID
WHERE Enrollment.SchoolName = ‘Springfield Elementary’
```

Join Table!

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.SchoolName = ‘Springfield Elementary’
```

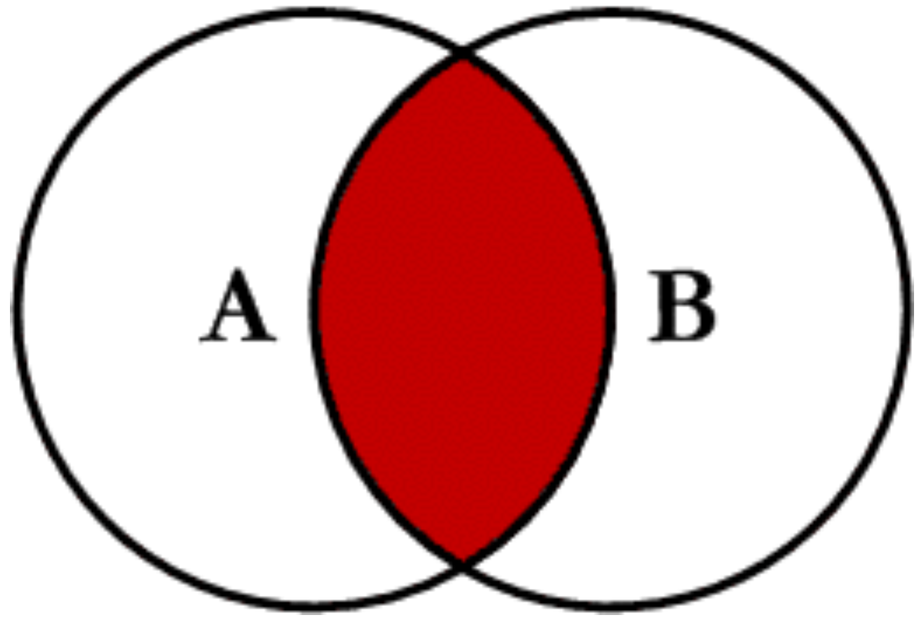
...or like this

```
SELECT *
FROM Student, Enrollment, School
WHERE Student.id = Enrollment.Student.ID
    AND Enrollment.SchoolID = School.id
    AND Enrollment.SchoolName = ‘Springfield Elementary’
```

This is the same, except it returns the count of students at Springfield Elementary

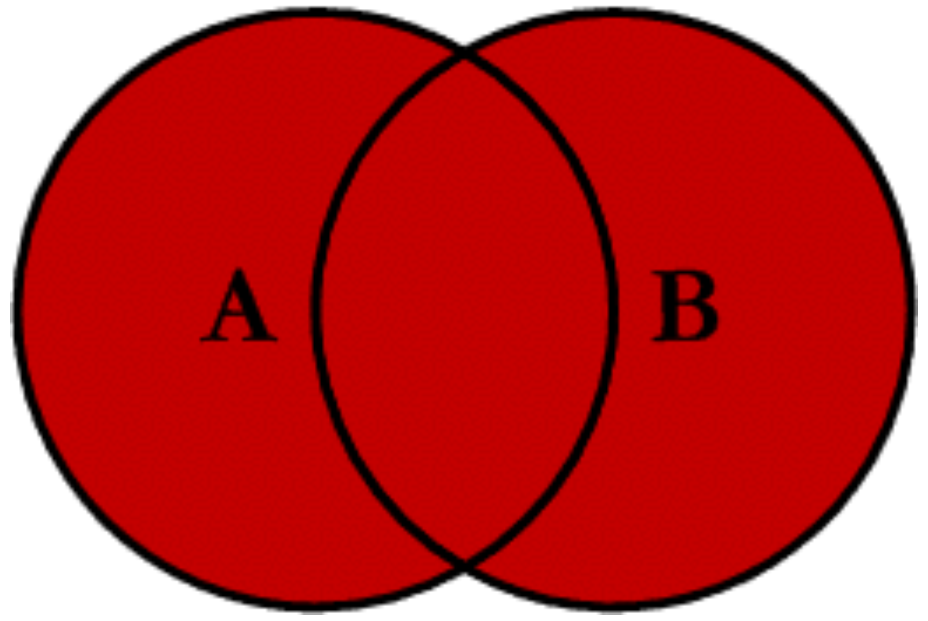
```
SELECT COUNT(*)
FROM Student, Enrollment, School
WHERE Student.id = Enrollment.Student.ID
    AND Enrollment.SchoolID = School.id
    AND Enrollment.SchoolName = ‘Springfield Elementary’
```

Inner Join



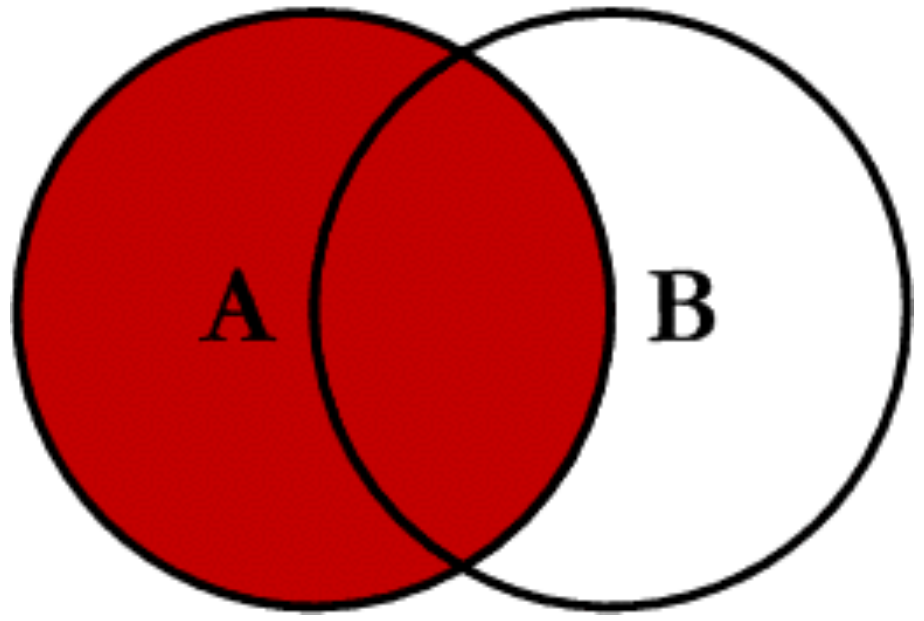
```
SELECT *
FROM A
INNER JOIN B
ON A.Key = B.Key
```

Outer Join



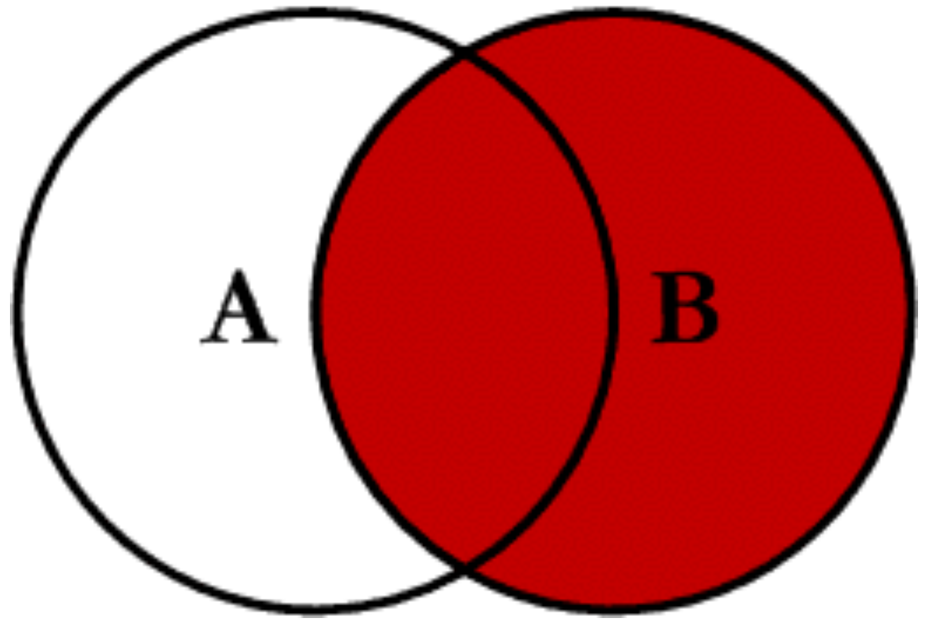
```
SELECT *
FROM A
FULL OUTER JOIN B
ON A.Key = B.Key
```

Left Join



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

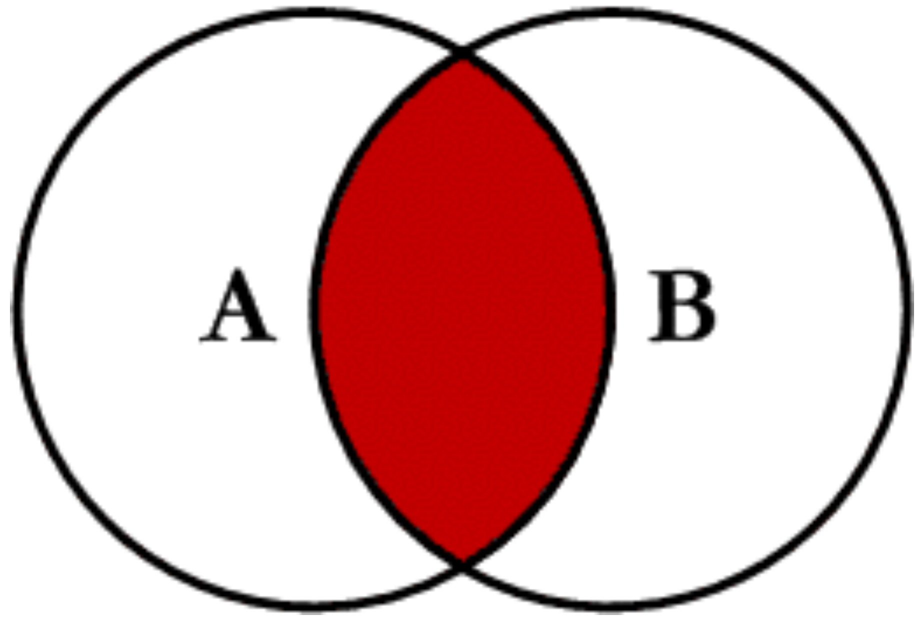
Right Join



```
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

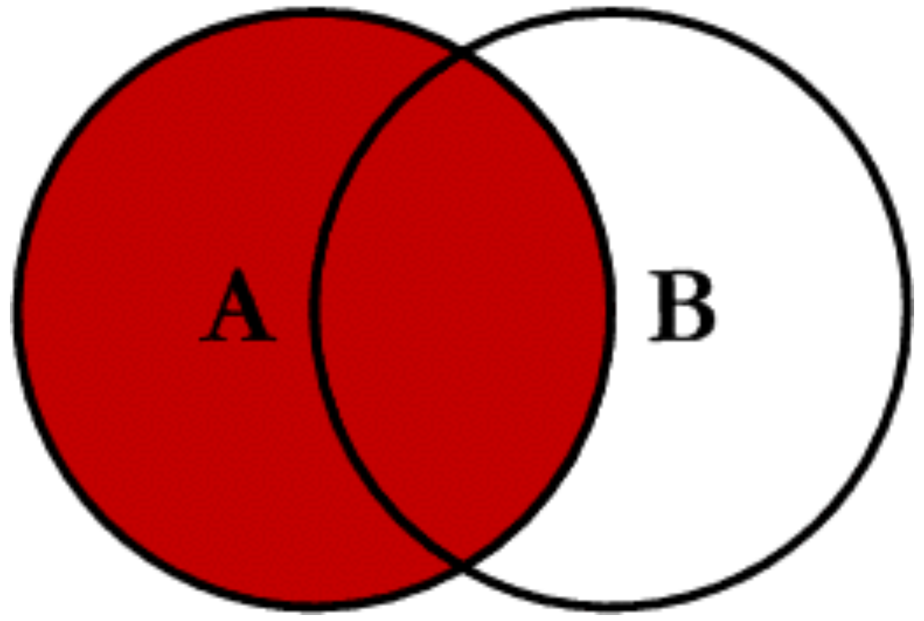
pets.name	owners.name
Mittens	Spok
Rufus	Geordi
Fireball	Janeway

PETS

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

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

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

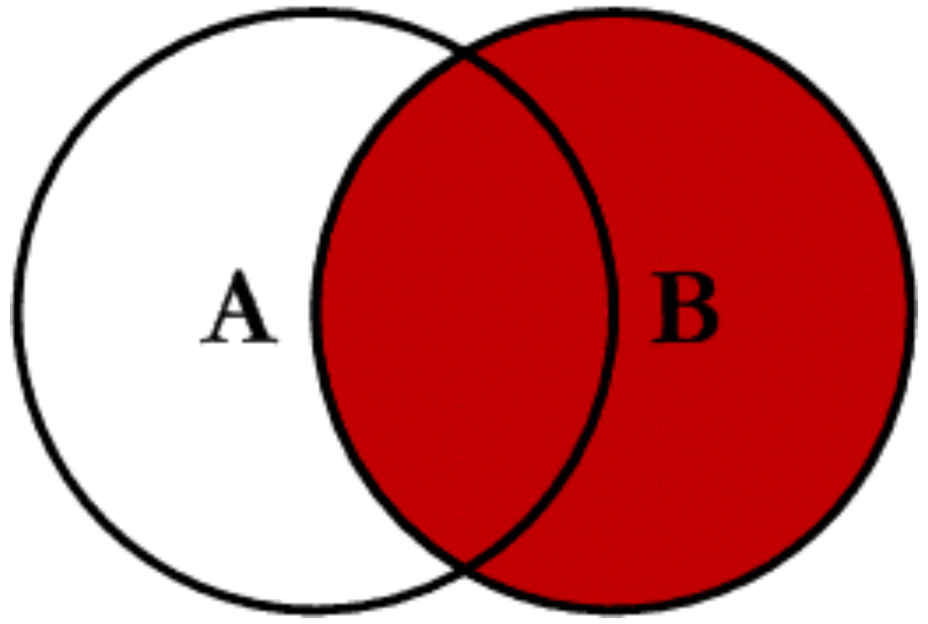
PETS

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

OWNERS

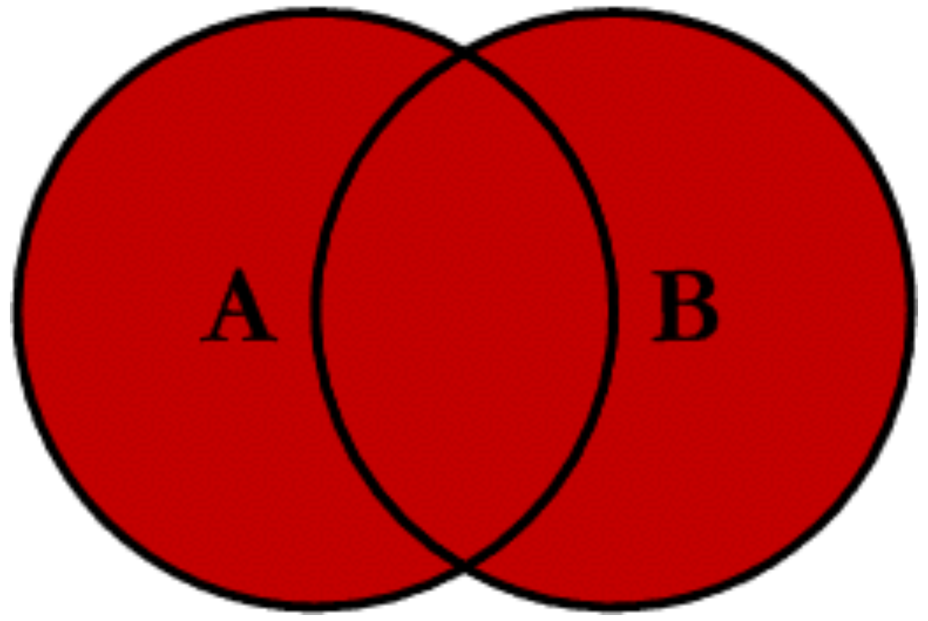
ID	name
1	Geordi
2	Janeway
3	Data
4	Spok

Right Join



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

Outer Join



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

OWNERS

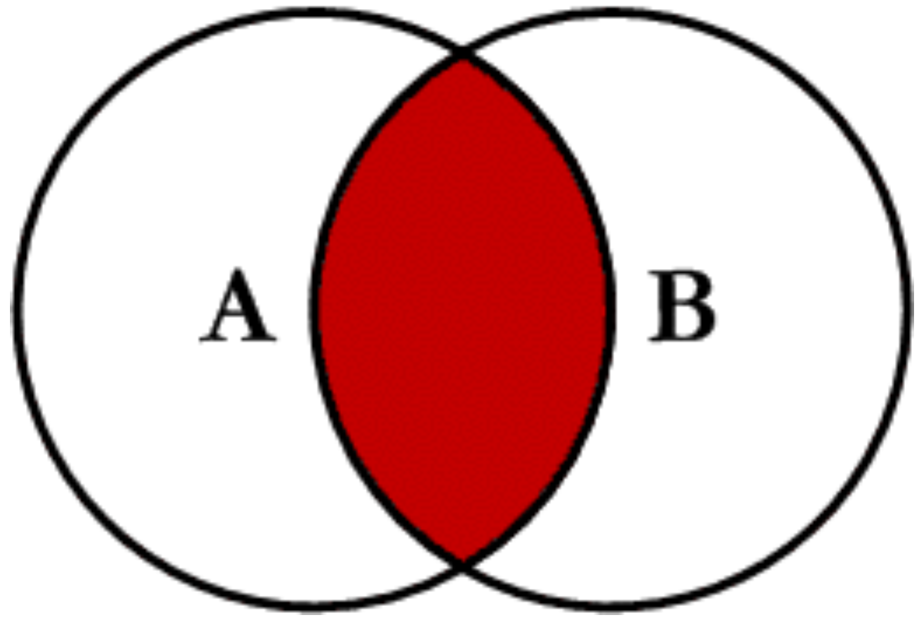
ID	name
1	Geordi
2	Janeway
3	Data
4	Spok

PETS

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

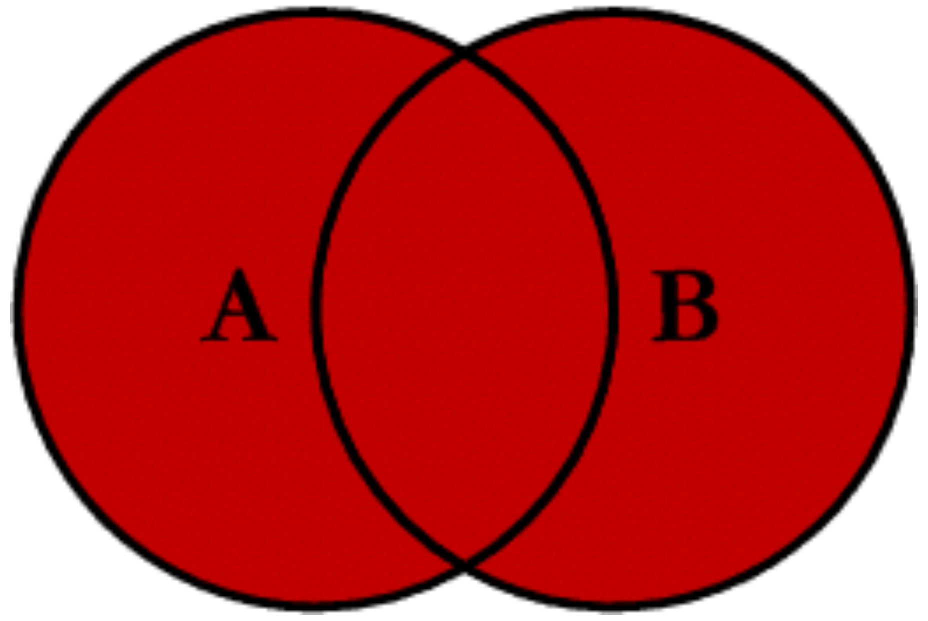
pets.name	owners.name
Mittens	Spok
Carol	null
Rufus	Geordi
Fireball	Janeway
null	Data

Inner Join



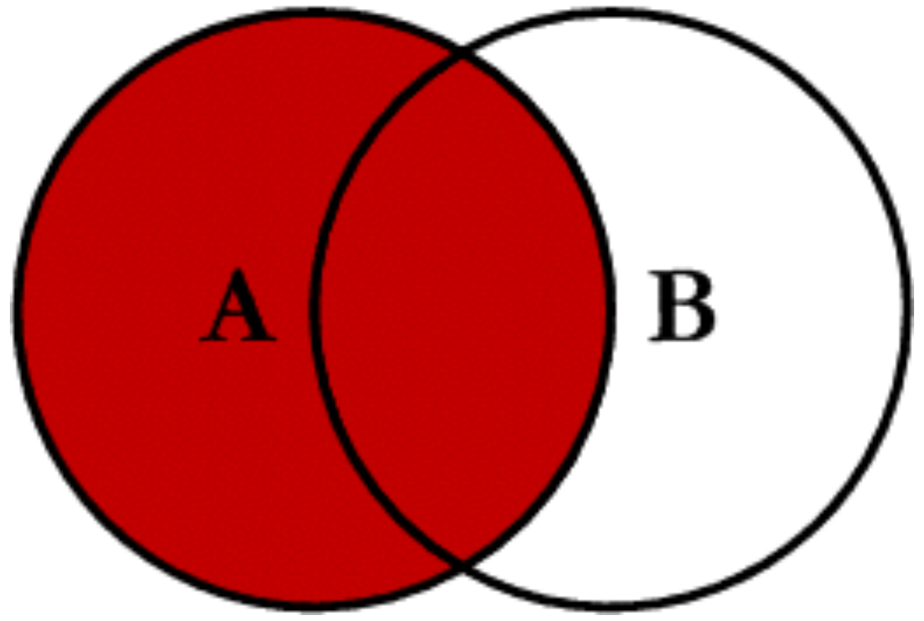
```
SELECT *
FROM A
INNER JOIN B
ON A.Key = B.Key
```

Outer Join



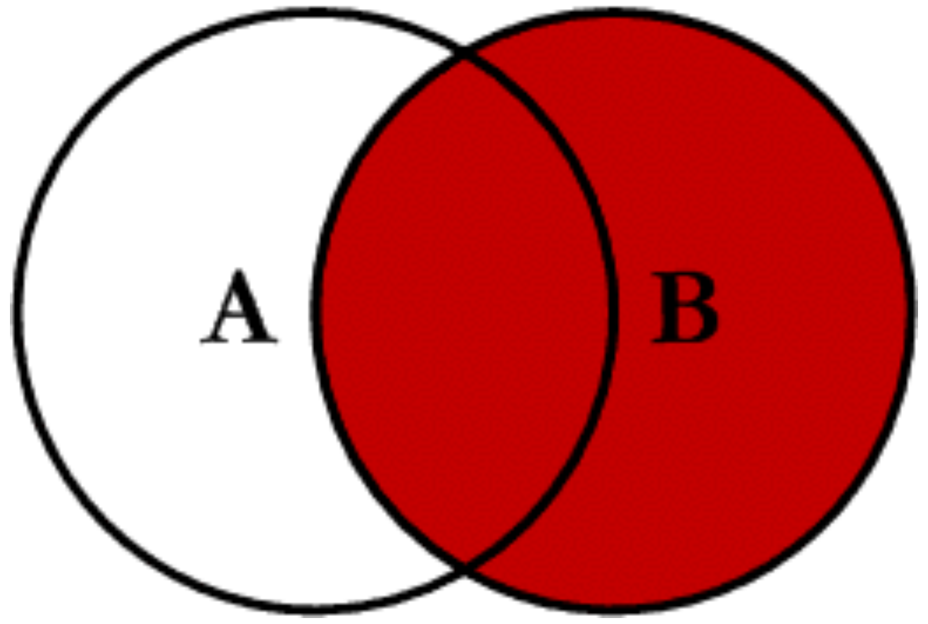
```
SELECT *
FROM A
FULL OUTER JOIN B
ON A.Key = B.Key
```

Left Join



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

Right Join



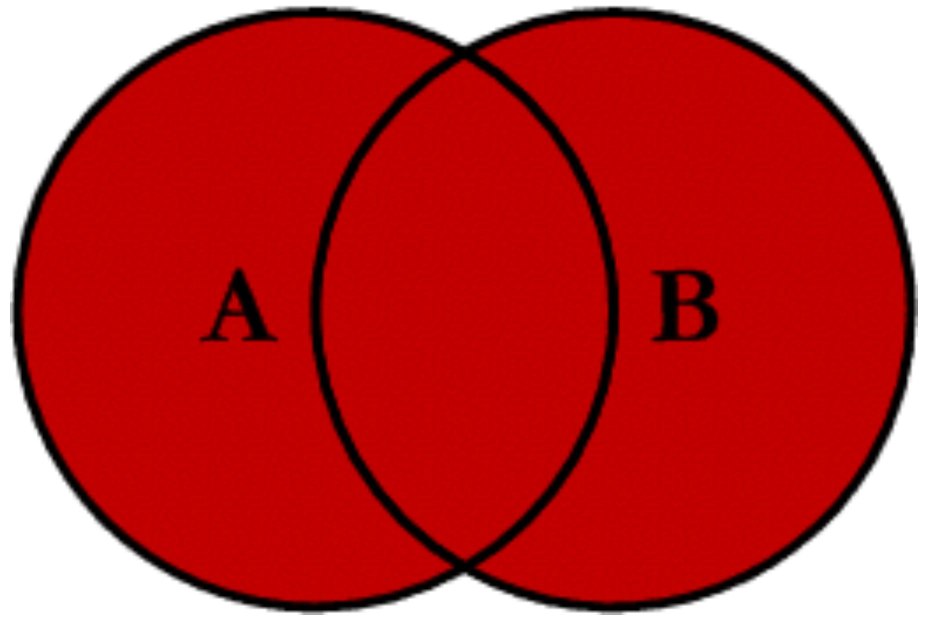
```
SELECT *
FROM A
RIGHT JOIN B
ON A.Key = B.Key
```

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

OWNERS

ID	name
1	Geordi
2	Janeway
3	Data
4	Spock

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

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

RDBMS vs NoSQL

- **A DBMS doesn't *have* to be relational**
  - Remember, DBMS is just an application that intelligently stores data and can answer requests to manage that data
- **Lately, many "NoSQL" or non-relational DBMSs have been gaining popularity**
  - Graph databases (e.g. GraphQL)
  - Document databases (e.g. MongoDB)
  - Hybrids (e.g. PostgreSQL)
- **RDBMSs still remain the #1 DB option for now**

Enough Theory. Examples!

All 20 Year Old Students

Students

ID	Name	Age	Gender	Address
1	Nick D.	20	M	2
2	Andy D.	28	M	2
3	Beth M.	23	F	1
4	Lisa N.	20	F	4

20 Year Old Students

ID	Name	Age
1	Nick D.	20
4	Lisa N.	20

```
SELECT ID, Name, Age
FROM Students
WHERE Age = 20;
```

Students

ID	Name	Age	Gender	Address
1	Nick D.	20	M	2
2	Andy D.	28	M	2
3	Beth M.	23	F	1
4	Lisa N.	20	F	4

```
SELECT Students.ID, Name, Street, Zip, City
  FROM Students
JOIN Addresses
  ON Students.Address = Addresses.ID
```

Addresses

ID	Street	Zip	City	State
1	423 Main St.	60647	Chicago	IL
2	13 Main St.	60655	Barrington	IL
3	15 Main St.	60651	Elsewhere	IL
4	14 Main St.	60650	Chicago	IL

Students with Addresses

Student.ID	Name	Street	Zip	City
1	Nick D.	13 Main St.	60655	Barrington
2	Andy D.	13 Main St.	60655	Barrington
3	Beth M.	423 Main St.	60647	Chicago
4	Lisa N.	14 Main St.	60650	Chicago

Students

ID	Name	Age	Gender	Address
1	Nick D.	20	M	2
2	Andy D.	28	M	2
3	Beth M.	23	F	1
4	Lisa N.	20	F	4

```
SELECT Student.ID, Name, Street, Zip, City
  FROM Students
JOIN Addresses
  ON Students.Address = Addresses.ID
WHERE Adresses.City = 'chicago';
```

Addresses

ID	Street	Zip	City	State
1	423 Main St.	60647	Chicago	IL
2	13 Main St.	60655	Barrington	IL
3	15 Main St.	60651	Elsewhere	IL
4	14 Main St.	60650	Chicago	IL

Students with Addresses

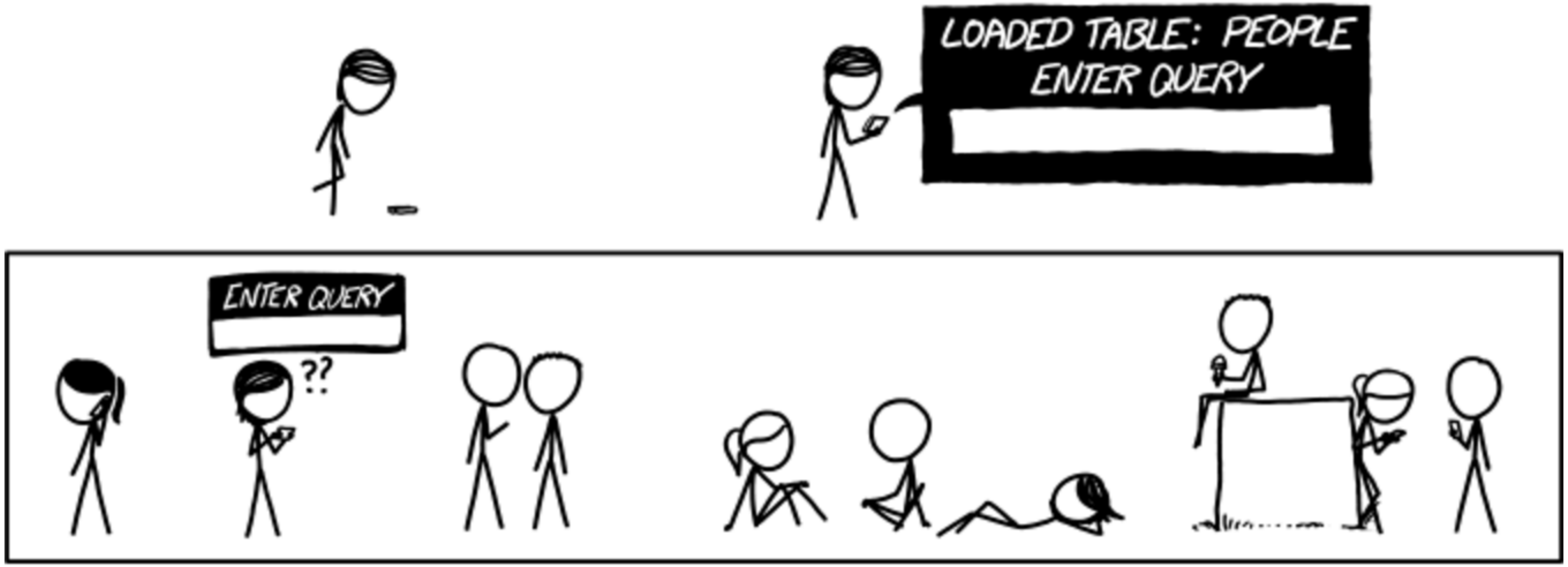
Student.ID	Name	Street	Zip	City
3	Beth M.	423 Main St.	60647	Chicago
4	Lisa N.	14 Main St.	60650	Chicago

Some Common SQL Keywords

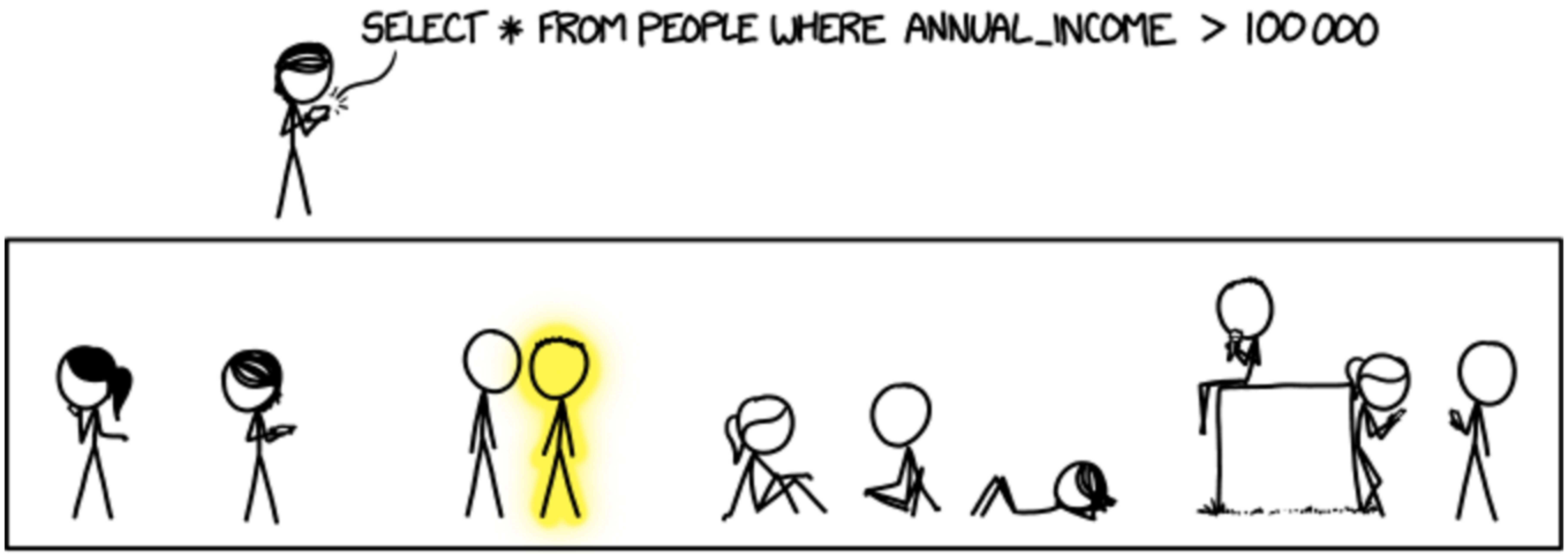
Keyword	Action
SELECT	Which COLUMNS to include in output table (shrinks the result horizontally!)
FROM	Which TABLE to pull data from
JOIN	Another TABLE to glue / concatenate to the output
ON	What COLUMNS must match when joining two tables
WHERE	Which ROWS to include in the output table (shrinks the result vertically!)
LIKE	often used with 'where', e.g. WHERE name LIKE "%don%" returns names containing 'don'
AS	SELECT name as some_alias FROM Students allows you to alias a column to refer to it later.

Some common SQL functions

Keyword	Action
COUNT	SELECT COUNT(col_name) from Students // => how many records in col_name of Student table
MAX	SELECT MAX(col_name) from Students // => largest value in col_name of Students table

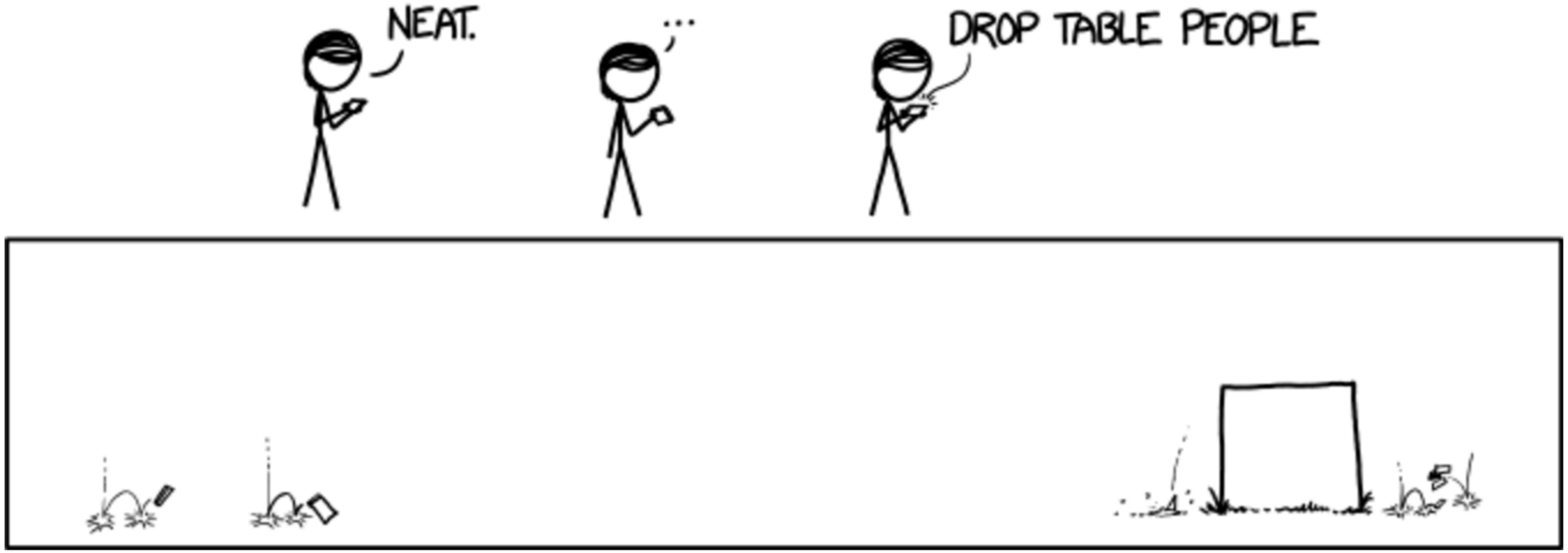

LOADED TABLE: PEOPLE
ENTER QUERY
ENTER QUERY
??

SELECT * FROM PEOPLE WHERE AGE > 30



SELECT * FROM PEOPLE WHERE ANNUAL_INCOME > 100000

SELECT * FROM PEOPLE WHERE AFRAID_OF_FLYING = TRUE


NEAT.
...
DROP TABLE PEOPLE

WORKSHOP