Data Science Methods for Clean Energy Research (DSMCER)

Database Management Systems: Relational Databases & SQL

Caitlyn Wolf

UW Chemical Engineering

March 5th, 2019

Announcements

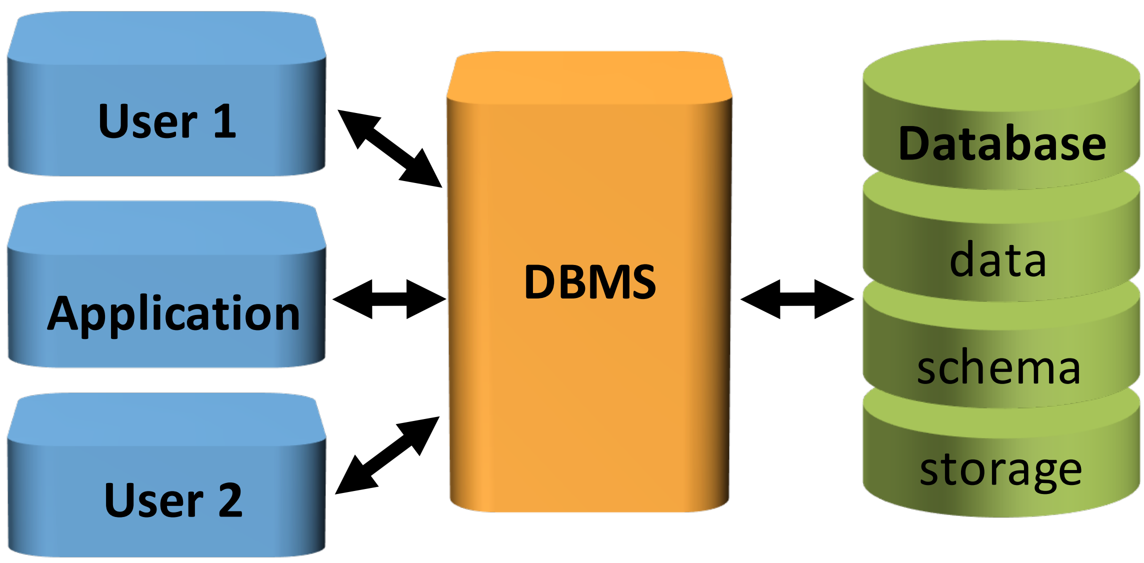
- Student stand-ups are this afternoon!

Outline

- Introduction to database management systems (DBMS’s)
  - What are they and why should I care?
- Data models
  - Hierarchical, Network, **Relational,** Entity-Relationship
- Database keys
- Database normalization
- SQL

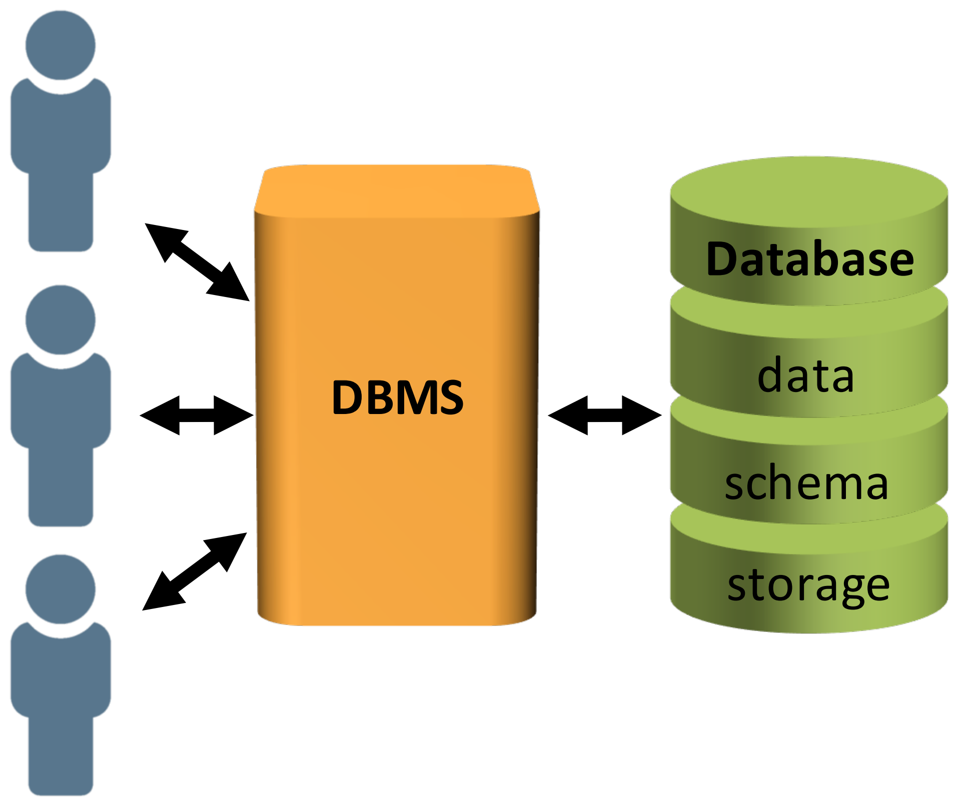
What is a database management system or DBMS?

A **database management system or DBMS** allows a user to efficiently build, access, modify and update a database of information.



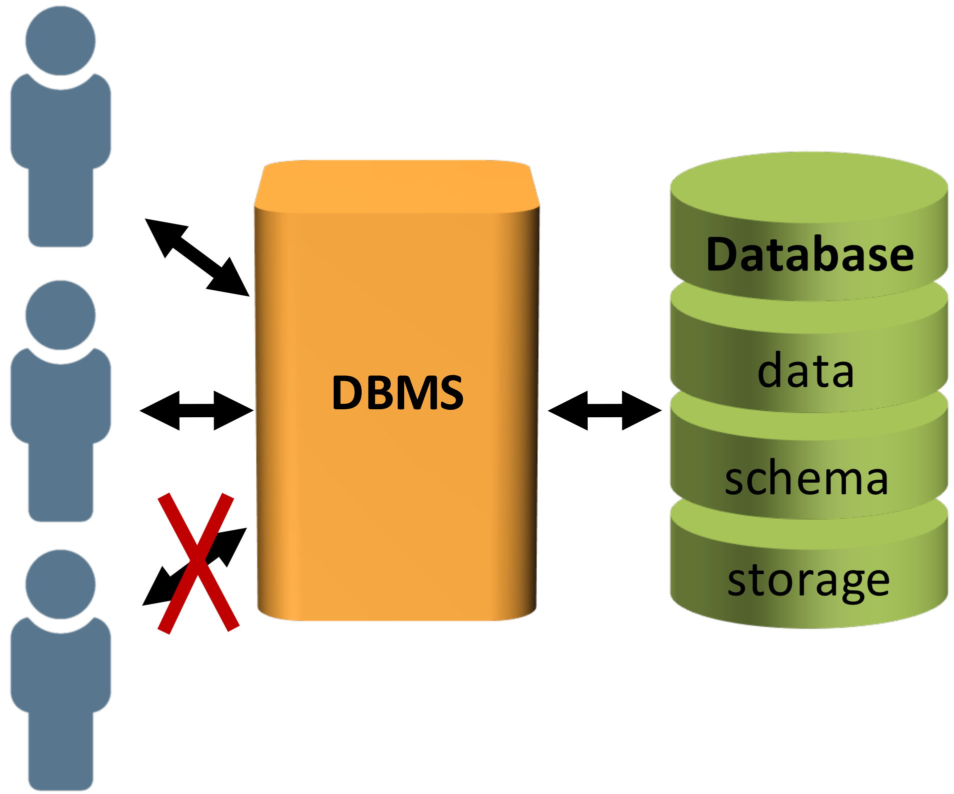
Advantages (and Disadvantages) of a DBMS

- Improved data access and sharing



Advantages (and Disadvantages) of a DBMS

- Improved data access and sharing
- Data administration and security



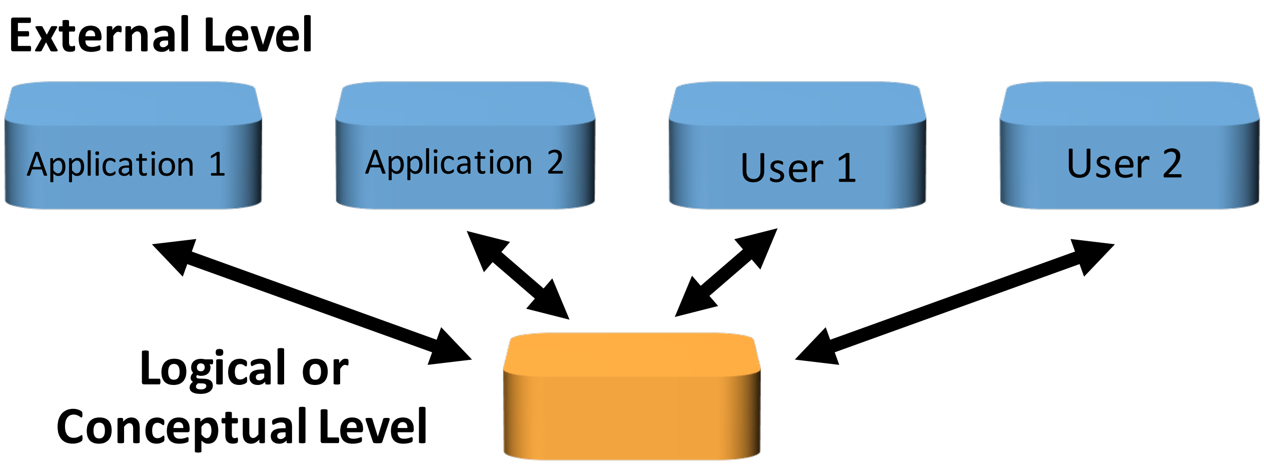
Advantages (and Disadvantages) of a DBMS

- Improved data access and sharing
- Data administration and security
- Data integrity (accuracy and consistency)

Advantages (and Disadvantages) of a DBMS

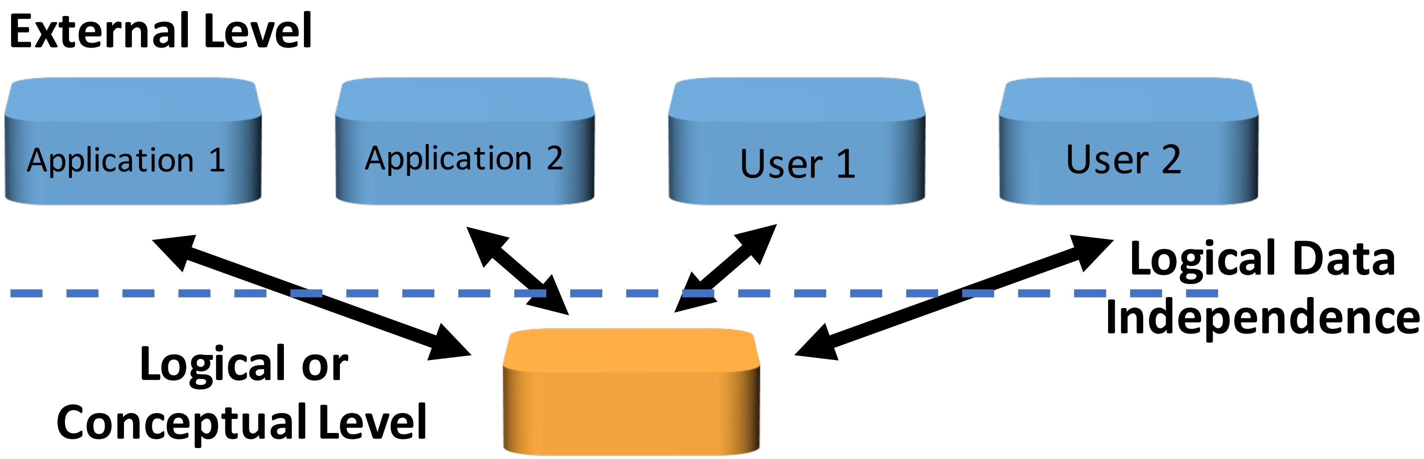
- Improved data access and sharing
- Data administration and security
- Data integrity (accuracy and consistency)
- Physical and logical data independence

Database Levels of Abstraction



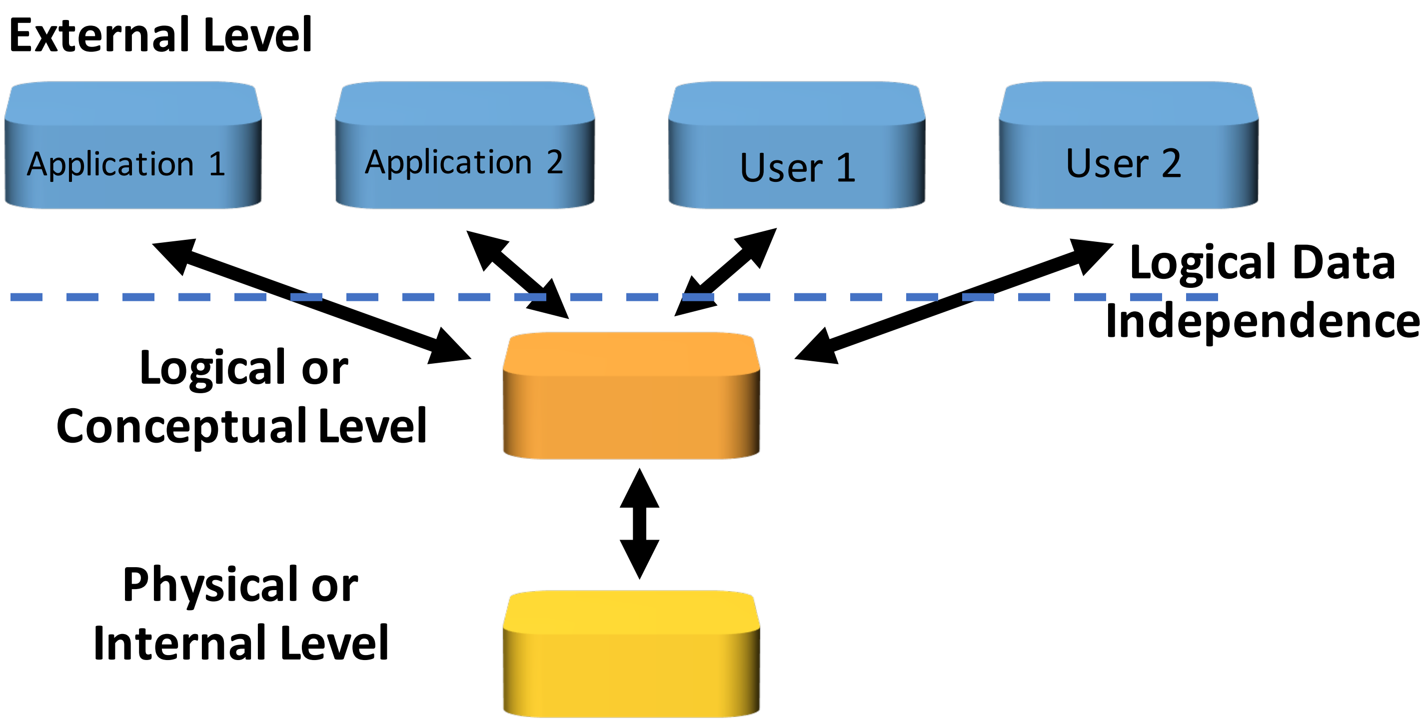
Reference: http://jcsites.juniata.edu/faculty/rhodes/dbms/dbarch.htm

Database Levels of Abstraction



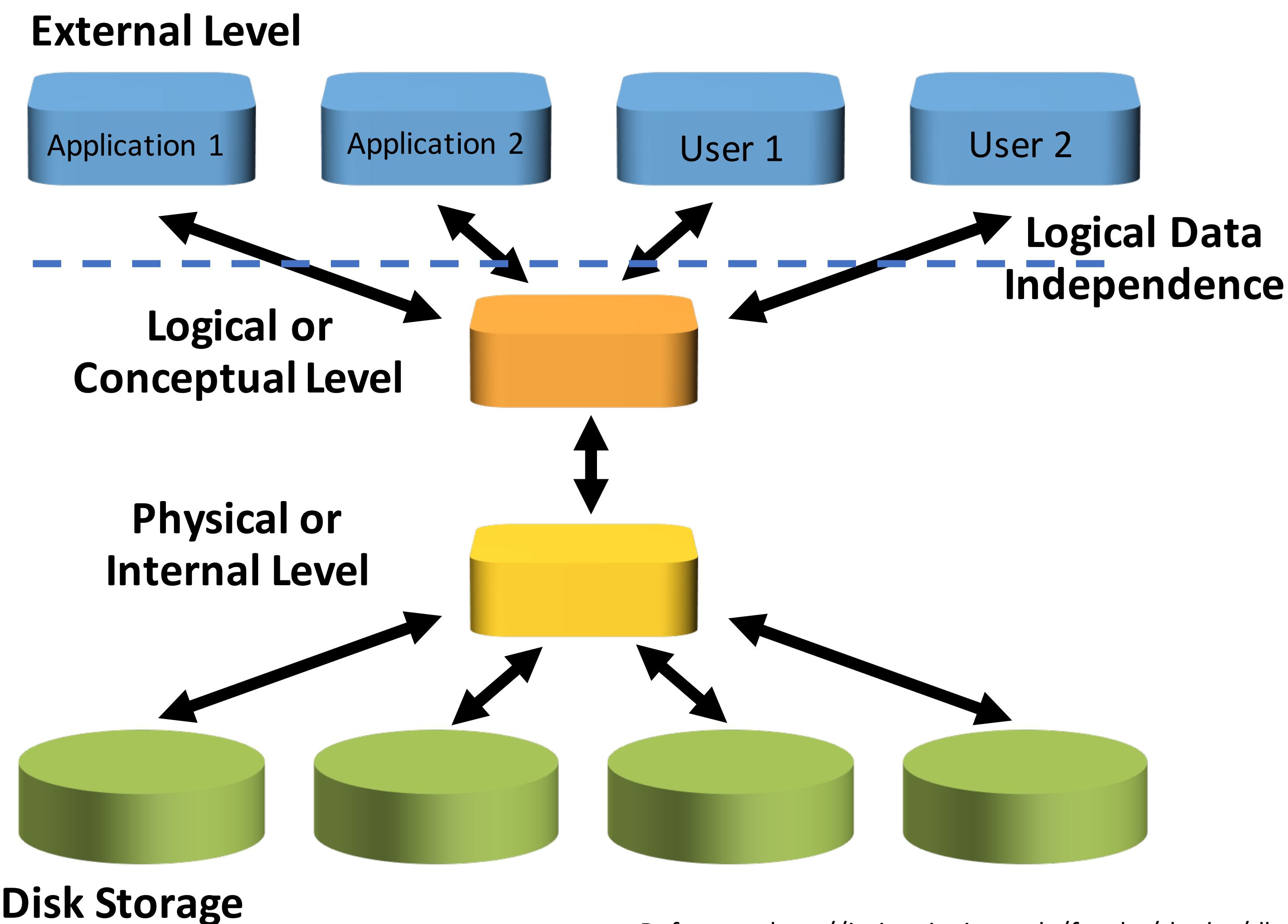
Reference: http://jcsites.juniata.edu/faculty/rhodes/dbms/dbarch.htm

Database Levels of Abstraction



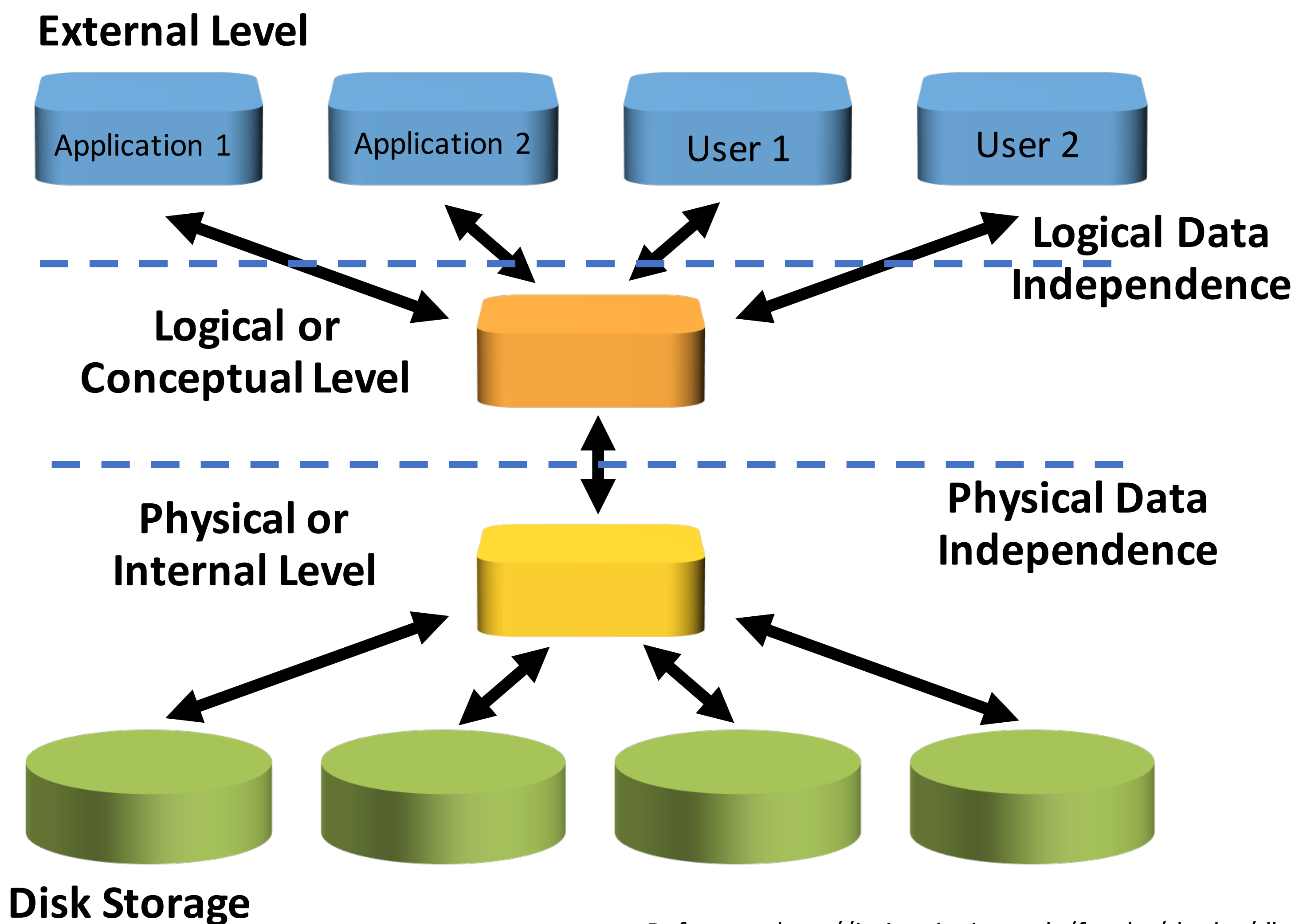
Reference: http://jcsites.juniata.edu/faculty/rhodes/dbms/dbarch.htm

Database Levels of Abstraction



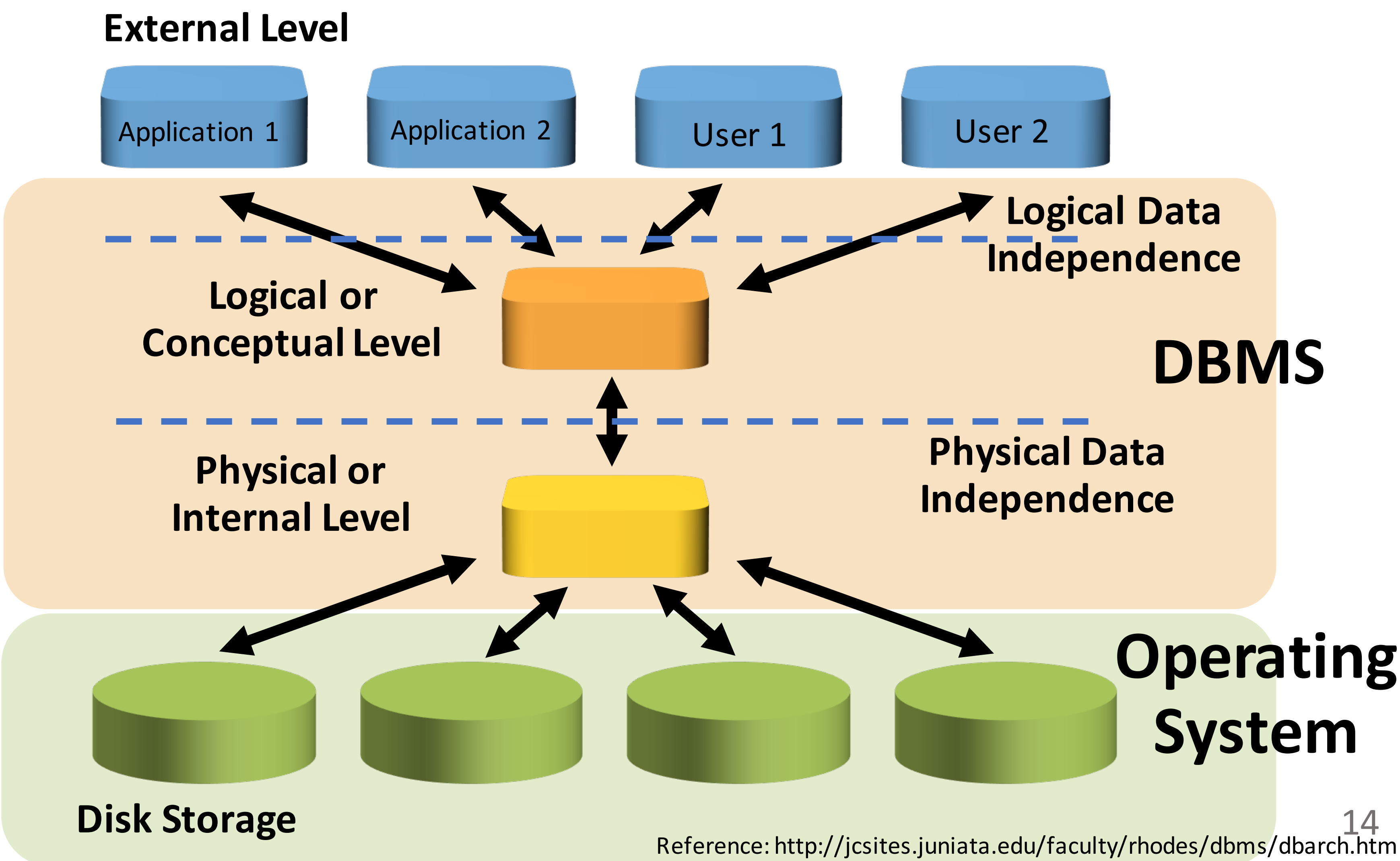
Reference: http://jcsites.juniata.edu/faculty/rhodes/dbms/dbarch.htm

Database Levels of Abstraction



Reference: http://jcsites.juniata.edu/faculty/rhodes/dbms/dbarch.htm

Database Levels of Abstraction



Reference: http://jcsites.juniata.edu/faculty/rhodes/dbms/dbarch.htm

Advantages (and Disadvantages) of a DBMS

- Improved data access and sharing
- Data administration and security
- Data integrity (accuracy and consistency)
- Physical and logical data independence
- Data loss protection

Advantages (and Disadvantages) of a DBMS

- Improved data access and sharing
- Data administration and security
- Data integrity (accuracy and consistency)
- Physical and logical data independence
- Data loss protection

- High cost

Advantages (and Disadvantages) of a DBMS

- Improved data access and sharing
- Data administration and security
- Data integrity (accuracy and consistency)
- Physical and logical data independence
- Data loss protection

- High cost
- Requires additional management and security

Advantages (and Disadvantages) of a DBMS

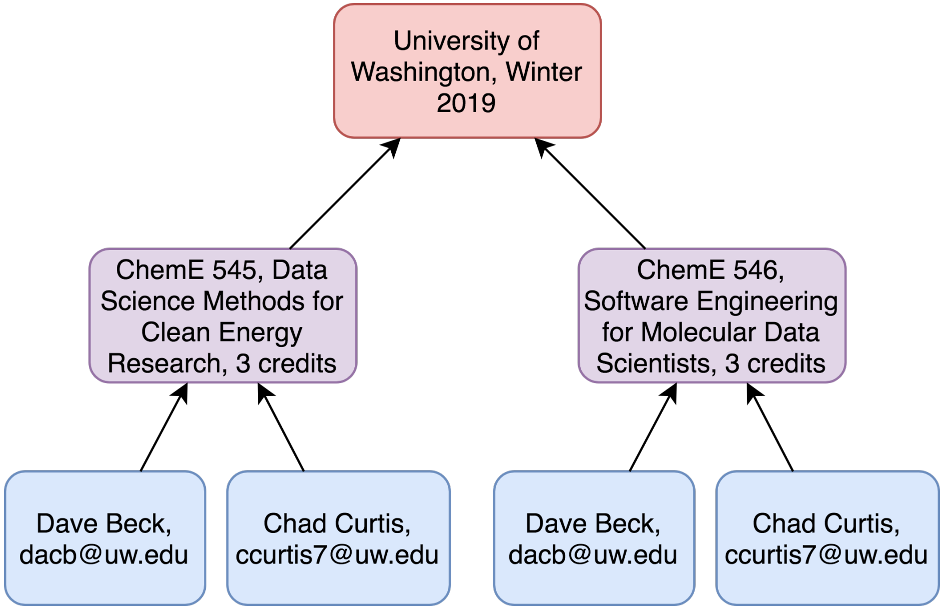
- Improved data access and sharing
- Data administration and security
- Data integrity (accuracy and consistency)
- Physical and logical data independence
- Data loss protection

- High cost
- Requires additional management and security
- More complex than general file systems

Data Models

Data Models: Hierarchical

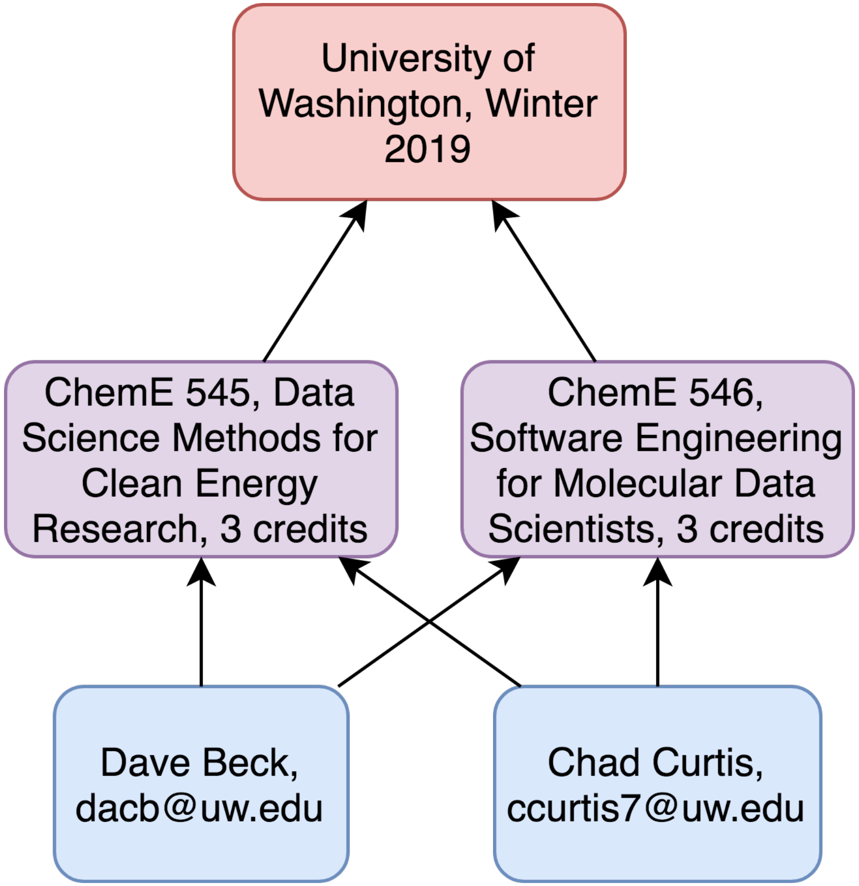
- Simple language to interact with and retrieve data from the system
- Some logical data independence is allowed
- Each "child" can only have one "parent" but "parents" can have multiple children.
- Leads to redundant data prone to inconsistencies upon updates
- Does not offer flexibility (e.g. instructor cannot exist if they are not teaching a course)
- Lacks physical data independence (i.e. changes in physical storage would break applications)



Additional Reading: Stonebraker, Michael, and Joey Hellerstein. "What goes around comes around." *Readings in Database Systems* 4 (2005): 1724-1735

Data Models: Network

- More flexible than a hierarchical model
- Limits redundant data by allowing many-many relationships
- Complex to work through and retrieve data from
- No physical data independence
- No logical data independence



Additional Reading: Stonebraker, Michael, and Joey Hellerstein. "What goes around comes around." *Readings in Database Systems* 4 (2005): 1724-1735

Data Models: Relational

- Offers physical AND logical data independence
- Flexible design
- Simplistic compared to the network model
- Language is difficult to understand (relational algebra & calculus)
- Inefficient data retrieval

Class ID	Number	Name	Credits
1	ChemE 545	DSMCER	3
2	ChemE 546	SEDS	3

Class ID	Instructor ID
1	1
1	2
2	1
2	2

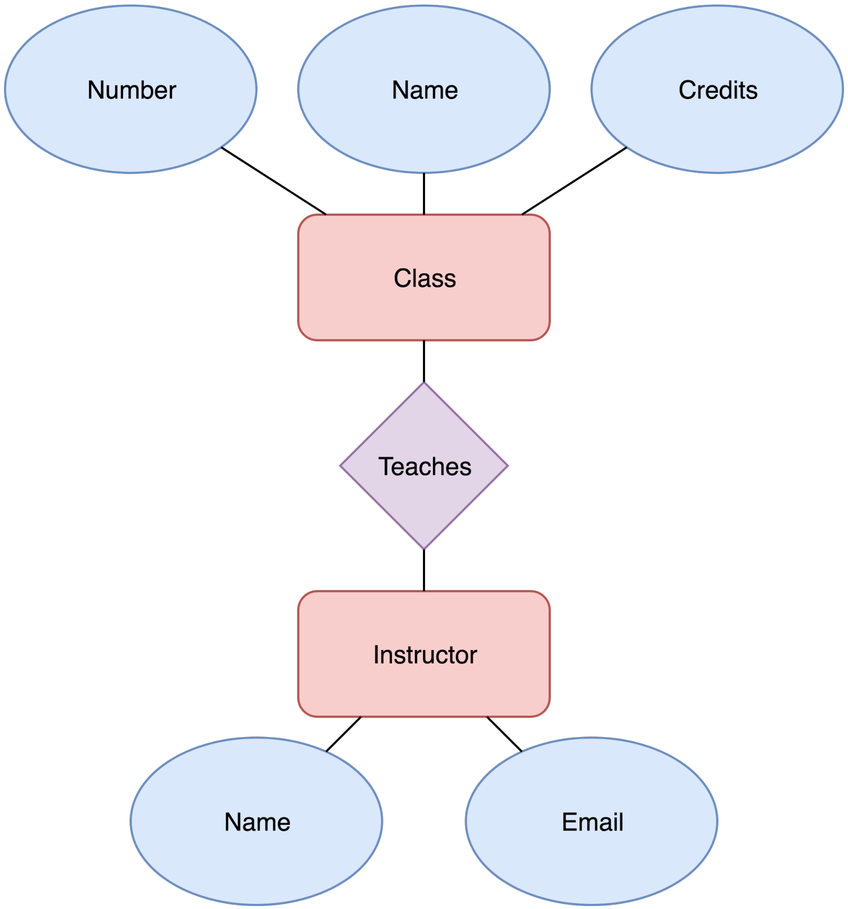
Instructor ID	Name	Email
1	Dave Beck	dacb@uw.edu
2	Chad Curtis	ccurtis7@uw.edu

In the 1980's, IBM had the final say in the debate between relational and network models....relational came out on top (see additional reading).

Additional Reading: Stonebraker, Michael, and Joey Hellerstein. "What goes around comes around." *Readings in Database Systems* 4 (2005): 1724-1735

Data Models: Entity-Relationship (E/R – Diagrams)

- Missing a language for interactions with the system
- Overshadowed by relational models
- Became successful as a **schema design tool for relational databases**



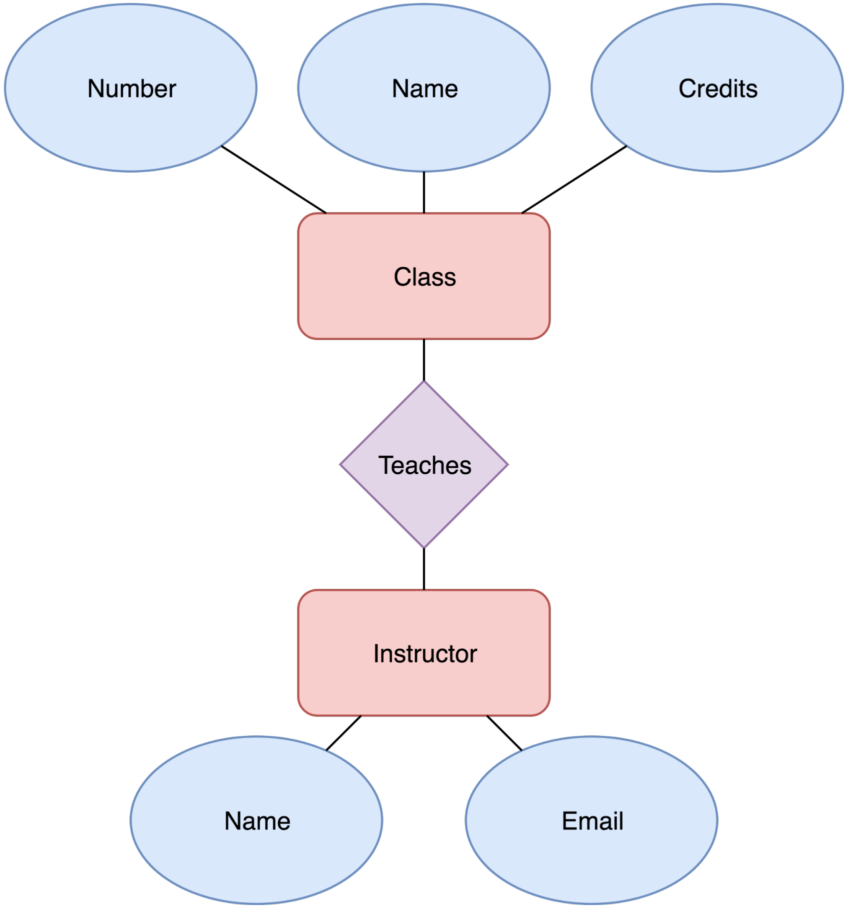
Additional Reading: Stonebraker, Michael, and Joey Hellerstein. "What goes around comes around." *Readings in Database Systems* 4 (2005): 1724-1735

Data Models: Entity-Relationship (E/R – Diagrams)

Class ID	Number	Name	Credits
1	ChemE 545	DSMCER	3
2	ChemE 546	SEDS	3

Class ID	Instructor ID
1	1
1	2
2	1
2	2

Instructor ID	Name	Email
1	Dave Beck	dacb@uw.edu
2	Chad Curtis	ccurtis7@uw.edu



Schema: roadmap of data organization in the database.

Database Keys

Database Keys

Class ID	Number	Name	Credits
1	ChemE 545	DSMCER	3
2	ChemE 546	SEDS	3

Instructor ID	Name	Email
1	Dave Beck	dacb@uw.edu
2	Chad Curtis	ccurtis7@uw.edu

Class ID	Instructor ID
1	1
1	2
2	1
2	2

Super, Candidate, Primary and Foreign Keys

- A key is used to uniquely identify a row, or tuple, in each table within a database.
- Keys are especially important for defining relationships between data.

Class ID	Number	Name	Credits
1	ChemE 545	DSMCER	3
2	ChemE 546	SEDS	3

Instructor ID	Name	Email
1	Dave Beck	dacb@uw.edu
2	Chad Curtis	ccurtis7@uw.edu

Class ID	Instructor ID
1	1
1	2
2	1
2	2

Super, Candidate, Primary and Foreign Keys

- A **super key** is an attribute, or set of attributes, that can determine all other attributes in a tuple.

What are the super keys?

Employee ID	Social Security No.	Name
1	123-45-ABCD	Caitlyn
2	123-45-DCBA	Dave
3	123-54-BACD	Torin
4	456-12-BCAD	Chad
5	456-98-BCDA	Ted
6	123-54-ABDC	Dave

- Employee ID
- Social Security Number
- Employee ID, Social Security Number
- Employee ID, Name
- Social Security Number, Name
- Employee ID, Social Security Number, Name

Super, Candidate, Primary and Foreign Keys

- A **super key** is an attribute, or set of attributes, that can determine all other attributes in a tuple.
- A **candidate key**, also called a minimal super key, is a key in which all attributes are required for uniquely identifying the tuple and remaining a key. This does **not** have to be a single attribute.

What are the candidate keys?

Employee ID	Social Security No.	Name
1	123-45-ABCD	Caitlyn
2	123-45-DCBA	Dave
3	123-54-BACD	Torin
4	456-12-BCAD	Chad
5	456-98-BCDA	Ted
6	123-54-ABDC	Dave

- Employee ID
- Social Security Number
- Employee ID, Social Security Number
- Employee ID, Name
- Social Security Number, Name
- Employee ID, Social Security Number, Name

What are the candidate keys?

Employee ID	Social Security No.	Name
1	123-45-ABCD	Caitlyn
2	123-45-DCBA	Dave
3	123-54-BACD	Torin
4	456-12-BCAD	Chad
5	456-98-BCDA	Ted
6	123-54-ABDC	Dave

- Employee ID
- Social Security Number
- ~~Employee ID, Social Security Number~~
- ~~Employee ID, Name~~
- ~~Social Security Number, Name~~
- ~~Employee ID, Social Security Number, Name~~

Super, Candidate, Primary and Foreign Keys

- A **super key** is an attribute, or set of attributes, that can determine all other attributes in a tuple.
- A **candidate key**, also called a minimal super key, is a key in which all attributes of the key are required for uniquely identifying the tuple and remaining a key. This does **not** have to be a single attribute.
- A **primary key** is one specific candidate key chosen in the database design to serve as the unique identifier for each tuple.

What is the primary key?

Employee ID	Social Security No.	Name
1	123-45-ABCD	Caitlyn
2	123-45-DCBA	Dave
3	123-54-BACD	Torin
4	456-12-BCAD	Chad
5	456-98-BCDA	Ted
6	123-54-ABDC	Dave

- Employee ID
- Social Security Number
- ~~Employee ID, Social Security Number~~
- ~~Employee ID, Name~~
- ~~Social Security Number, Name~~
- ~~Employee ID, Social Security Number, Name~~

What is the primary key?

Employee ID	Social Security No.	Name
1	123-45-ABCD	Caitlyn
2	123-45-DCBA	Dave
3	123-54-BACD	Torin
4	456-12-BCAD	Chad
5	456-98-BCDA	Ted
6	123-54-ABDC	Dave

- **Employee ID**
- ~~Social Security Number~~
- ~~Employee ID, Social Security Number~~
- ~~Employee ID, Name~~
- ~~Social Security Number, Name~~
- ~~Employee ID, Social Security Number, Name~~

Super, Candidate, Primary and Foreign Keys

- A **super key** is an attribute, or set of attributes, that can determine all other attributes in a tuple.
- A **candidate key**, also called a minimal super key, is a key in which all attributes of the key are required for uniquely identifying the tuple and remaining a key. This does **not** have to be a single attribute.
- A **primary key** is one specific candidate key chosen in the database design to serve as the unique identifier for each tuple.
- A **foreign key** references a primary key in another table and is used to define relationships between tuples in separate tables.

What are the primary and foreign keys?

Employee ID	Social Security No.	Name
1	123-45-ABCD	Caitlyn
2	123-45-DCBA	Dave
3	123-54-BACD	Torin
4	456-12-BCAD	Chad
5	456-98-BCDA	Ted
6	123-54-ABDC	Dave

Department ID	Employee ID
100	2
100	5
300	6
200	1
200	6
300	4
300	3

Department ID	Department Name	Floor Location
100	Sales	1
200	Marketing	2
300	Payroll	2

What are the primary and foreign keys?

Employee ID	Social Security No.	Name
1	123-45-ABCD	Caitlyn
2	123-45-DCBA	Dave
3	123-54-BACD	Torin
4	456-12-BCAD	Chad
5	456-98-BCDA	Ted
6	123-54-ABDC	Dave

Primary keys

Department ID	Department Name	Floor Location
100	Sales	1
200	Marketing	2
300	Payroll	2

Department ID	Employee ID
100	2
100	5
300	6
200	1
200	6
300	4
300	3

What are the primary and foreign keys?

Employee ID	Social Security No.	Name
1	123-45-ABCD	Caitlyn
2	123-45-DCBA	Dave
3	123-54-BACD	Torin
4	456-12-BCAD	Chad
5	456-98-BCDA	Ted
6	123-54-ABDC	Dave

Foreign keys

Department ID	Employee ID
100	2
100	5
300	6
200	1
200	6
300	4
300	3

Primary keys

Department ID	Department Name	Floor Location
100	Sales	1
200	Marketing	2
300	Payroll	2

In-class activity: Find the super and candidate keys

Class Name	Class Number	Instructor	Department
Data Science Methods for Clean Energy Research	545	Dave Beck	Chem E
Data Science Methods for Clean Energy Research	545	Dave Beck	MSE
Machine Learning	546	Kevin Jamieson	CSE
Colloidal Systems	556	John Berg	Chem E
Principles of Data Management	544	Dan Suciu	CSE
SEDS	546	Dave Beck	Chem E

In-class activity:
Find the super and candidate keys

Class Name	Class Number	Instructor	Department
Data Science Methods for Clean Energy Research	545	Dave Beck	Chem E
Data Science Methods for Clean Energy Research	545	Dave Beck	MSE
Machine Learning	546	Kevin Jamieson	CSE
Colloidal Systems	556	John Berg	Chem E
Principles of Data Management	544	Dan Suciu	CSE
SEDS	546	Dave Beck	Chem E

- **Class Name, Department**
- **Class Number, Department**
- Class Name, Instructor, Department
- Class Name, Class Number, Department
- Class Number, Instructor, Department
- Class Name, Class Number, Instructor, Department

Super keys

Candidate keys

Normalization

Database Normalization

- Normalization helps with schema design by providing a set of rules to limit data redundancy and support data integrity
- Rules get more strict as move forward through:
  - 1st Normal Form
  - 2nd Normal Form
  - 3rd Normal Form
  - Boyce Codd Normal Form (BCNF)
- For more information beyond what we will cover today, check out this reference guide:
  - https://www.studytonight.com/dbms/database-normalization.php

1^{st} Normal Form

- All tables should be 2-dimensional, i.e. not more than one value in each cell.

1st Normal Form

- All tables should be 2-dimensional, i.e. not more than one value in each cell.
- What do you see as a problem in this database:

Airline	Flight Number	Path	Plane Model	Pilot	Passenger	Passenger City	Passenger State
Alaska	36	SEA → MSP	A	Dave	Caitlyn	Seattle	Washington
American Airlines	36	SEA → MSP	B	Chad	Ted	Portland	Oregon
Alaska	7	MSP → JFK	C	Dave	Ted, Torin	Portland, San Diego	Oregon, California
Alaska	3367	MSP → JFK	C	Dave	Caitlyn, Torin	Seattle, San Diego	Washington, California
American Airlines	3367	JFK → SEA	D	Chad	Torin	San Diego	California

1st Normal Form

- All tables should be 2-dimensional, i.e. not more than one value in each cell.
- What do you see as a problem in this database:

Airline	Flight Number	Path	Plane Model	Pilot	Passenger	Passenger City	Passenger State
Alaska	36	SEA → MSP	A	Dave	Caitlyn	Seattle	Washington
American Airlines	36	SEA → MSP	B	Chad	Ted	Portland	Oregon
Alaska	7	MSP → JFK	C	Dave	Ted, Torin	Portland, San Diego	Oregon, California
Alaska	3367	MSP → JFK	C	Dave	Caitlyn, Torin	Seattle, San Diego	Washington, California
American Airlines	3367	JFK → SEA	D	Chad	Torin	San Diego	California

Multiple passengers are grouped together in common cells.

1st Normal Form

- All tables should be 2-dimensional, i.e. not more than one value in each cell.

Airline	Flight Number	Path	Plane Model	Pilot
Alaska	36	SEA → MSP	A	Dave
American Airlines	36	SEA → MSP	B	Chad
Alaska	7	MSP → JFK	C	Dave
Alaska	3367	MSP → JFK	C	Dave
American Airlines	3367	JFK → SEA	D	Chad

Airline	Flight Number	Passenger	Passenger City	Passenger State
Alaska	36	Caitlyn	Seattle	Washington
American Airlines	36	Ted	Portland	Oregon
Alaska	7	Ted	Portland	Oregon
Alaska	7	Torin	San Diego	California
Alaska	3367	Caitlyn	Seattle	Washington
Alaska	3367	Torin	San Diego	California
American Airlines	3367	Torin	San Diego	California

2^{nd} Normal Form

- 1^{st} normal form must be met and there should be no partial functional dependencies.
- AB→CDE, A→E

2^{nd} Normal Form

- 1^{st} normal form must be met and there should be no partial functional dependencies.
- AB→CDE, A→E

Airline	Flight Number	Path	Pilot	Plane Model
Alaska	36	SEA → MSP	Dave	A
American Airlines	36	SEA → MSP	Chad	B
Alaska	7	MSP → JFK	Dave	C
Alaska	3367	MSP → JFK	Dave	C
American Airlines	3367	JFK → SEA	Chad	D

2^{nd} Normal Form

- 1^{st} normal form must be met and there should be no partial functional dependencies.
- AB→CDE, A→E

Airline	Flight Number	Path	Pilot	Plane Model
Alaska	36	SEA → MSP	Dave	A
American Airlines	36	SEA → MSP	Chad	B
Alaska	7	MSP → JFK	Dave	C
Alaska	3367	MSP → JFK	Dave	C
American Airlines	3367	JFK → SEA	Chad	D

Airline, Flight Number → Path, Pilot, Plane Model

Airline → Pilot

2nd Normal Form

- 1st normal form must be met and there should be no partial functional dependencies.
- AB→CDE, A→E

Airline	Flight Number	Path	Plane Model
Alaska	36	SEA → MSP	A
American Airlines	36	SEA → MSP	B
Alaska	7	MSP → JFK	C
Alaska	3367	MSP → JFK	C
American Airlines	3367	JFK → SEA	D

Airline	Pilot
Alaska	Dave
American Airlines	Chad

2^{nd} Normal Form

- 1^{st} normal form must be met and there should be no partial functional dependencies.
- AB→CDE, A→E

2nd Normal Form

- 1st normal form must be met and there should be no partial functional dependencies.
- AB→CDE, A→E

Airline	Flight Number	Passenger	Passenger City	Passenger State
Alaska	36	Caitlyn	Seattle	Washington
American Airlines	36	Ted	Portland	Oregon
Alaska	7	Ted	Portland	Oregon
Alaska	7	Torin	Seattle	Washington
Alaska	3367	Caitlyn	Seattle	Washington
Alaska	3367	Torin	Seattle	Washington
American Airlines	3367	Torin	Seattle	Washington

2^{nd} Normal Form

- 1^{st} normal form must be met and there should be no partial functional dependencies.
- AB→CDE, A→E

Airline	Flight Number	Passenger	Passenger City	Passenger State
Alaska	36	Caitlyn	Seattle	Washington
American Airlines	36	Ted	Portland	Oregon
Alaska	7	Ted	Portland	Oregon
Alaska	7	Torin	Seattle	Washington
Alaska	3367	Caitlyn	Seattle	Washington
Alaska	3367	Torin	Seattle	Washington
American Airlines	3367	Torin	Seattle	Washington

Airline, Flight Number, Passenger → City, State

Passenger → City, State

2nd Normal Form

- 1st normal form must be met and there should be no partial functional dependencies.
- AB→CDE, A→E

Airline	Flight Number	Passenger
Alaska	36	Caitlyn
American Airlines	36	Ted
Alaska	7	Ted
Alaska	7	Torin
Alaska	3367	Caitlyn
Alaska	3367	Torin
American Airlines	3367	Torin

Passenger	Passenger City	Passenger State
Caitlyn	Seattle	Washington
Ted	Portland	Oregon
Torin	Seattle	Washington

3rd Normal Form

- 2nd normal form must be met and there should be no transitive functional dependencies.
- A → B, B → C

3rd Normal Form

- 2nd normal form must be met and there should be no transitive functional dependencies.
- A → B, B → C

Passenger	**Passenger City**	**Passenger State**
Caitlyn	Seattle	Washington
Ted	Portland	Oregon
Torin	Seattle	Washington

3rd Normal Form

- 2nd normal form must be met and there should be no transitive functional dependencies.
- A → B, B → C

Passenger	Passenger City	Passenger State
Caitlyn	Seattle	Washington
Ted	Portland	Oregon
Torin	Seattle	Washington

Passenger → City

City → State

3rd Normal Form

- 2nd normal form must be met and there should be no transitive functional dependencies.
- A → B, B → C

Passenger	Passenger City
Caitlyn	Seattle
Ted	Portland
Torin	Seattle

Passenger City	Passenger State
Seattle	Washington
Portland	Oregon

Boyce Codd Normal Form (BCNF)

- Must meet 3rd normal form, and every functional dependency must be trivial or represent a super key.
- AB → C, C → B

Boyce Codd Normal Form (BCNF)

- Must meet 3rd normal form, and every functional dependency must be trivial or represent a super key.
- AB → C, C → B

Airline	Flight Number	Path	Plane Model
Alaska	36	SEA → MSP	A
American Airlines	36	SEA → MSP	B
Alaska	7	MSP → JFK	C
Alaska	3367	MSP → JFK	C
American Airlines	3367	JFK → SEA	D

Boyce Codd Normal Form (BCNF)

- Must meet 3rd normal form, and every functional dependency must be trivial or represent a super key.
- AB → C, C → B

Airline	Flight Number	Path	Plane Model
Alaska	36	SEA → MSP	A
American Airlines	36	SEA → MSP	B
Alaska	7	MSP → JFK	C
Alaska	3367	MSP → JFK	C
American Airlines	3367	JFK → SEA	D

Airline, Flight Number, Path → Plane Model

Plane Model → Path

Boyce Codd Normal Form (BCNF)

- Must meet 3rd normal form, and every functional dependency must be trivial or represent a super key.
- AB → C, C → B

Airline	Flight Number	Path
Alaska	36	SEA → MSP
American Airlines	36	SEA → MSP
Alaska	7	MSP → JFK
Alaska	3367	MSP → JFK
American Airlines	3367	JFK → SEA

Path	Plane Model
SEA → MSP	A
SEA → MSP	B
MSP → JFK	C
JFK → SEA	D

Where we started

Airline	Flight Number	Path	Plane Model	Pilot	Passenger	Passenger City	Passenger State
Alaska	36	SEA → MSP	A	Dave	Caitlyn	Seattle	Washington
American Airlines	36	SEA → MSP	B	Chad	Ted	Portland	Oregon
Alaska	7	MSP → JFK	C	Dave	Ted, Torin	Portland, San Diego	Oregon, California
Alaska	3367	MSP → JFK	C	Dave	Caitlyn, Torin	Seattle, San Diego	Washington, California
American Airlines	3367	JFK → SEA	D	Chad	Torin	San Diego	California

Final design in BCNF

Airline	Flight Number	Path
Alaska	36	SEA → MSP
American Airlines	36	SEA → MSP
Alaska	7	MSP → JFK
Alaska	3367	MSP → JFK
American Airlines	3367	JFK → SEA

Airline	Pilot
Alaska	Dave
American Airlines	Chad

Path	Plane Model
SEA → MSP	A
SEA → MSP	B
MSP → JFK	C
JFK → SEA	D

Airline	Flight Number	Passenger
Alaska	36	Caitlyn
American Airlines	36	Ted
Alaska	7	Ted
Alaska	7	Torin
Alaska	3367	Caitlyn
Alaska	3367	Torin
American Airlines	3367	Torin

Passenger City	Passenger State
Seattle	Washington
Portland	Oregon

Passenger	Passenger City
Caitlyn	Seattle
Ted	Portland
Torin	Seattle

SQL:
Structured Query Language

SQL: Structured Query Language

- SQL can not only allow you to retrieve specific data from a relational database, it enables you to create, delete, and/or update information in the database.
- Today, we will focus on the basic structure for retrieving specific data from a database.
  - These will start with the keyword SELECT
- For more information, and a comprehensive SQL tutorial, check out this reference:
  - https://www.w3schools.com/sql/sql_intro.asp

SELECT

- You will always start your SQL queries with the following structure:

SELECT *feature 1, feature 2, feature 3…*

FROM *table;*

SELECT

Flights

Airline	Flight Number	Path
Alaska	36	SEA → MSP
American Airlines	36	SEA → MSP
Alaska	7	MSP → JFK
Alaska	3367	MSP → JFK
American Airlines	3367	JFK → SEA

SELECT Airline, Flight Number

FROM Flights;

SELECT

Flights

Airline	Flight Number	Path
Alaska	36	SEA → MSP
American Airlines	36	SEA → MSP
Alaska	7	MSP → JFK
Alaska	3367	MSP → JFK
American Airlines	3367	JFK → SEA

SELECT Airline, Flight Number

FROM Flights;

Results

Airline	Flight Number
Alaska	36
American Airlines	36
Alaska	7
Alaska	3367
American Airlines	3367

SELECT

Flights

Airline	Flight Number	Path
Alaska	36	SEA → MSP
American Airlines	36	SEA → MSP
Alaska	7	MSP → JFK
Alaska	3367	MSP → JFK
American Airlines	3367	JFK → SEA

```
SELECT *
FROM Flights;
```

SELECT

Flights

Airline	Flight Number	Path
Alaska	36	SEA → MSP
American Airlines	36	SEA → MSP
Alaska	7	MSP → JFK
Alaska	3367	MSP → JFK
American Airlines	3367	JFK → SEA

```
SELECT *
FROM Flights;
```

Results

Airline	Flight Number	Path
Alaska	36	SEA → MSP
American Airlines	36	SEA → MSP
Alaska	7	MSP → JFK
Alaska	3367	MSP → JFK
American Airlines	3367	JFK → SEA

SELECT DISTINCT

Flights

Airline	Flight Number	Path
Alaska	36	SEA → MSP
American Airlines	36	SEA → MSP
Alaska	7	MSP → JFK
Alaska	3367	MSP → JFK
American Airlines	3367	JFK → SEA

```
SELECT DISTINCT Airline
FROM Flights;
```

SELECT DISTINCT

Flights

Airline	Flight Number	Path
Alaska	36	SEA → MSP
American Airlines	36	SEA → MSP
Alaska	7	MSP → JFK
Alaska	3367	MSP → JFK
American Airlines	3367	JFK → SEA

```
SELECT DISTINCT Airline
FROM Flights;
```

Results

Airline
Alaska
American Airlines

WHERE

Flights

Airline	Flight Number	Path
Alaska	36	SEA → MSP
American Airlines	36	SEA → MSP
Alaska	7	MSP → JFK
Alaska	3367	MSP → JFK
American Airlines	3367	JFK → SEA

```
SELECT *
FROM Flights
WHERE Airline='Alaska';
```

WHERE

Flights

Airline	Flight Number	Path
Alaska	36	SEA → MSP
American Airlines	36	SEA → MSP
Alaska	7	MSP → JFK
Alaska	3367	MSP → JFK
American Airlines	3367	JFK → SEA

```
SELECT *
FROM Flights
WHERE Airline='Alaska';
```

Results

Airline	Flight Number	Path
Alaska	36	SEA → MSP
Alaska	7	MSP → JFK
Alaska	3367	MSP → JFK

WHERE

- You can use many different operators with the WHERE keyword to declare a condition:
    - =
    - <> or !=
    - >
    - <
    - >=
    - <=
- You can also string together many conditional statements using AND, OR, NOT

WHERE

Flights

Airline	Flight Number	Path
Alaska	36	SEA → MSP
American Airlines	36	SEA → MSP
Alaska	7	MSP → JFK
Alaska	3367	MSP → JFK
American Airlines	3367	JFK → SEA

```
SELECT *
FROM Flights
WHERE Airline='Alaska' AND
Path='SEA → MSP';
```

Results

Airline	Flight Number	Path
Alaska	36	SEA → MSP

JOINS

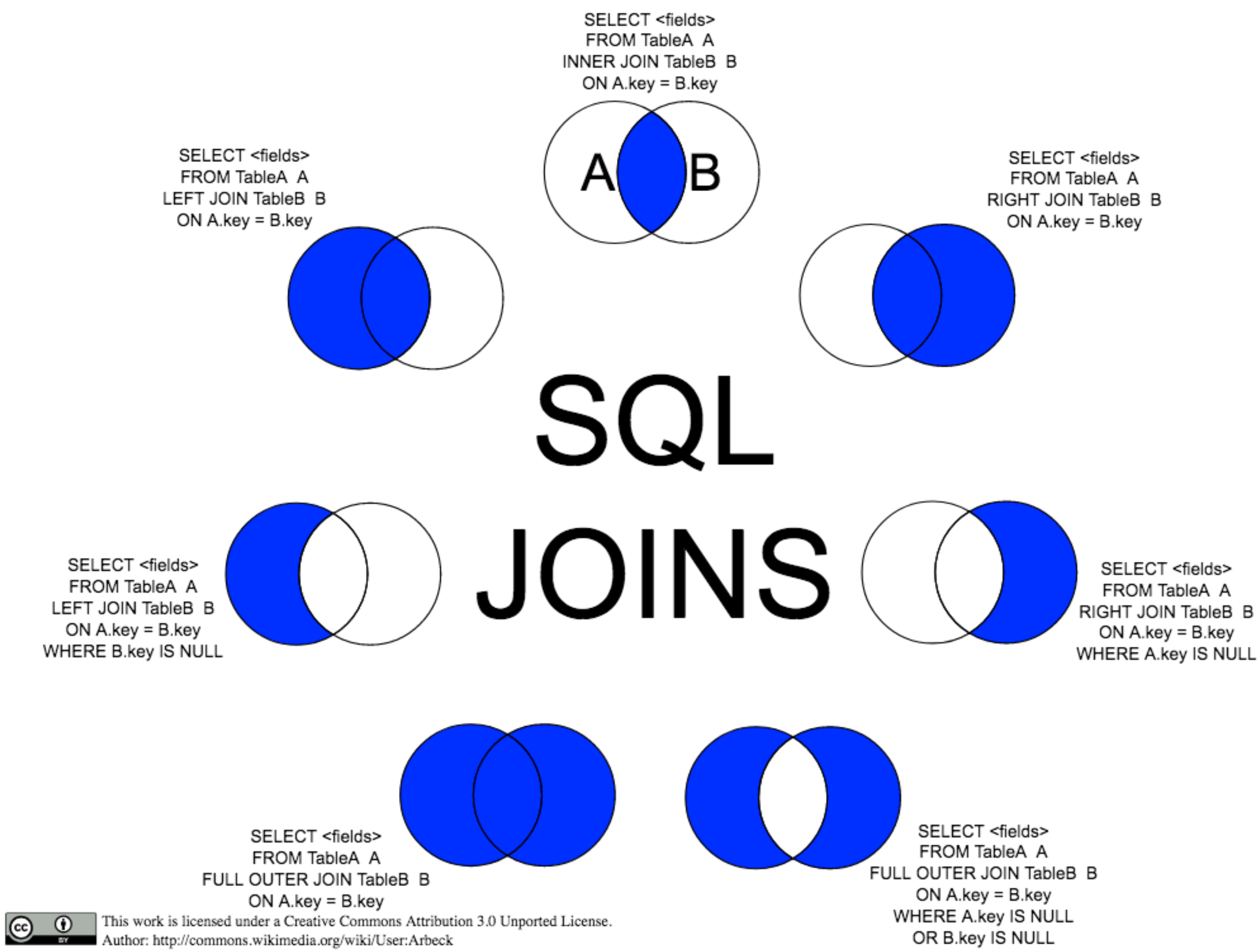
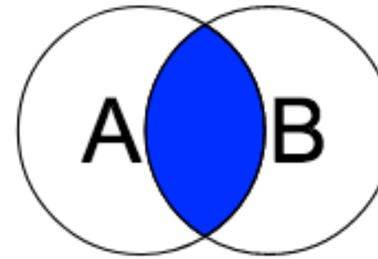


Image source: https://commons.wikimedia.org/wiki/File:SQL_Joins.svg

INNER JOIN

SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key



Passengers

Passenger	Passenger City
Caitlyn	Seattle
Ted	Portland
Torin	Seattle
Chad	New York

Homes

Passenger City	Passenger State
Seattle	Washington
Portland	Oregon
San Diego	California

SELECT Passengers.Passenger, Homes.PassengerState

FROM Passengers

INNER JOIN Homes ON Passengers.PassengerCity=Homes.PassengerCity;

Results

Passenger	Passenger State
Caitlyn	Washington
Ted	Oregon
Torin	Washington

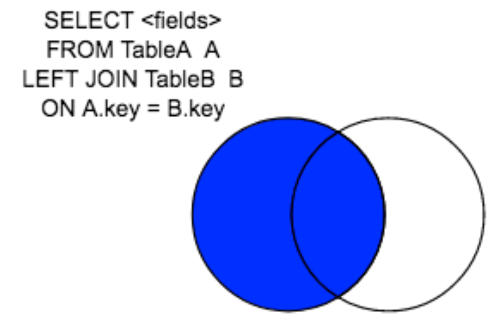
LEFT JOIN

Passengers

Passenger	Passenger City
Caitlyn	Seattle
Ted	Portland
Torin	Seattle
Chad	New York

Homes

Passenger City	Passenger State
Seattle	Washington
Portland	Oregon
San Diego	California



SELECT Passengers.Passenger, Homes.PassengerState

FROM Passengers

LEFT JOIN Homes ON Passengers.PassengerCity=Homes.PassengerCity;

Results

Passenger	Passenger State
Caitlyn	Washington
Ted	Oregon
Torin	Washington
Chad	NULL

Aggregate Functions

- COUNT, MAX, MIN, SUM, AVG are available to perform operations on your dataset

Passengers

Passenger	Passenger City
Caitlyn	Seattle
Ted	Portland
Torin	Seattle
Dave	Seattle
Chad	New York

SELECT COUNT(Passenger)

FROM Passengers

WHERE PassengerCity='Seattle';

Results

COUNT(Passenger)
3

GROUP BY

- You can also a GROUP BY keyword with an aggregate function

Passengers

Passenger	PassengerCity
Caitlyn	Seattle
Ted	Portland
Torin	Seattle
Dave	Seattle
Chad	New York

```
SELECT COUNT(Passenger), PassengerCity
FROM Passengers
GROUP BY PassengerCity
```

Results

COUNT(Passenger)	PassengerCity
3	Seattle
1	Portland
1	New York

Aliases

- You can give columns or tables temporary names using an alias

Passengers

Passenger	PassengerCity
Caitlyn	Seattle
Ted	Portland
Torin	Seattle
Dave	Seattle
Chad	New York

SELECT COUNT(Passenger) as Total, PassengerCity

FROM Passengers

GROUP BY PassengerCity

Results

Total	PassengerCity
3	Seattle
1	Portland
1	New York

Nesting with SELECT

- You can select from a table that is the result of another query

Passengers

Passenger	PassengerCity
Caitlyn	Seattle
Ted	Portland
Torin	Seattle
Dave	Seattle
Chad	New York

```
SELECT COUNT(Passenger) as Total, PassengerCity
FROM Passengers
GROUP BY PassengerCity
```

Results

Total	PassengerCity
3	Seattle
1	Portland
1	New York

Nesting with SELECT

- You can select from a table that is the result of another query

Passengers

Passenger	PassengerCity
Caitlyn	Seattle
Ted	Portland
Torin	Seattle
Dave	Seattle
Chad	New York

```
SELECT COUNT(Passenger) as Total, PassengerCity
FROM Passengers
GROUP BY PassengerCity
```

Results

Total	PassengerCity
3	Seattle
1	Portland
1	New York

```
SELECT MAX(Total), PassengerCity
FROM (
    SELECT COUNT(Passenger) as Total, PassengerCity
    FROM Passengers
    GROUP BY PassengerCity);
```

Results

Total	PassengerCity
3	Seattle

Practice writing SQL queries

The reference I provided also includes a practice relational database:

https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

Try writing each of the queries below and write down the answers:

1. How many customers are there in the database?
2. How many different first names are there among the employees?
3. How many different countries are the suppliers located in?
4. How many suppliers are located in France?
5. What is the most of one product item that was ever purchased on a single order?
6. Which country has the most suppliers? (Hint: explore a nested query)
7. Create a table of ProductName and CategoryName for products with a price less than 50.

Practice writing SQL queries

The reference I provided also includes a practice relational database:

https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

Try writing each of the queries below and write down the answers:

1. How many customers are there in the database? 91
2. How many different first names are there among the employees? 10
3. How many different countries are the suppliers located in? 17
4. How many suppliers are located in France? 3
5. What is the most of one product item that was ever purchased on a single order? 120
6. Which country has the most suppliers? (Hint: explore a nested query) USA
7. Create a table of ProductName and CategoryName for products with a price less than 50. 70 rows