Introduction to Databases

CEE412 / CET522

Transportation Data Management and Visualization

WINTER 2020

Announcement

- Homework 1 due on next Wednesday (Jan 22)
- Questions about Homework 1?
- Review of Quiz 1

- Clarifications
  - Class time: 8:30-9:50AM, Wen. and Fri.
  - Final exam time

Database

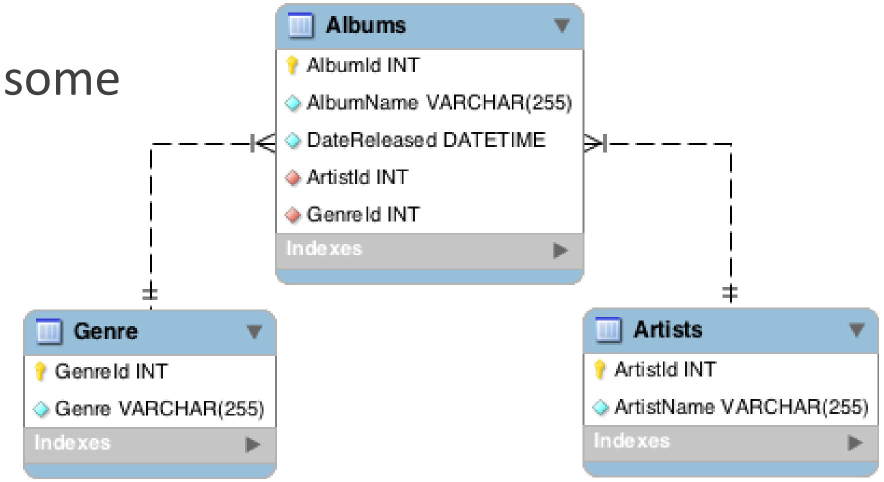
Database = DB

- A collection of information that exists over a long period of time

Key features of database

- Schema
  - The design or structure of the file system, describing the relation between different elements in the system and the hierarchy of data.
- Query
  - A query is a command issued by the DB user that performs some operation on the data contained in the DB.
- Large storage space
- Control access

Refer to the readings posted on Canvas



Albums
AlbumId INT
AlbumName VARCHAR(255)
DateReleased DATETIME
ArtistId INT
GenreId INT
Indexes

Genre
GenreId INT
Genre VARCHAR(255)
Indexes

Artists
ArtistId INT
ArtistName VARCHAR(255)
Indexes

https://database.guide/what-is-a-database-schema/

Database Management System

Database Management System = DBMS

A DBMS is a powerful tool for creating and managing a large amount of data efficiently and allowing it to persist over a long period of time:

- A collection of files that store the data
- A program that accesses and updates those files for you

Database Management System

Relational databases present a user with a view of data organized as tables called relations.

Image Credit: Antony Theobald via Flickr

Port	Arrive	Depart	Unit	Driver
DFW	7:15AM	9:20AM	BG3388	1220
DFW	2:00PM	4:10PM	AB3391	1001
LAX	9:50PM	1:00AM	AB7782	2231

Port	City	State	Growth	Passengers
ATL	Atlanta	GA	7.1%	3,200,000
DFW	Dallas	TX	0.4%	2,000,000
LAX	Los Angeles	CA	5.3%	1,900,000

Relational DBMS Example

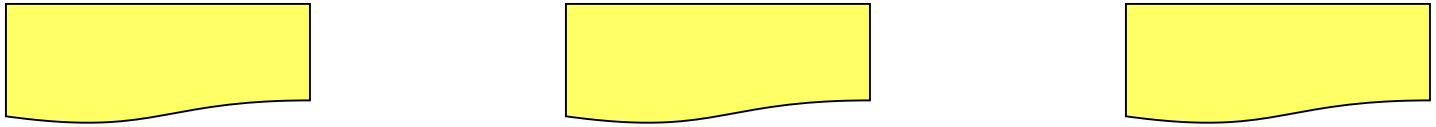
An example that shows why we need databases:

Suppose we are building a system to store the information about:

- students
- courses
- professors
- who takes what, who teaches what

Relational DBMS Example

If we do not use a DBMS, how do we do it?



students.txt courses.txt professors.txt

Use text files,

Then write C or Java programs to implement specific tasks.

Relational DBMS Example

Once the programs are developed, students can use them to register courses

Suppose student “John” wants to register for “CEE412”, then the program(s) need to conduct the following operations:

- Read ‘students.txt
- Read ‘courses.txt’
- Find & update the record for “John” in students.txt
- Find & update the record for “CEE412” in courses.txt
- Write “students.txt”
- Write “courses.txt”

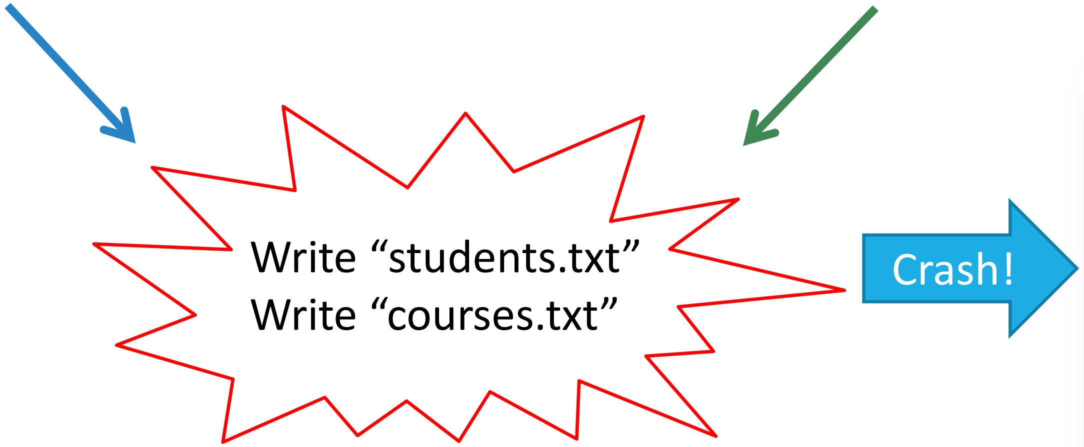
Relational DBMS Example

John:

- Read ‘students.txt
- Read ‘courses.txt’
- Find & update the record “John”
- Find & update the record “CEE412”

Kris:

- Read ‘students.txt
- Read ‘courses.txt’
- Find & update the record “Kris”
- Find & update the record “CEE412”



Write “students.txt”
Write “courses.txt”

Crash!

Image Credit: Ruin Raider via Flickr

Relational DBMS Example

If we have large data sets (say 50GB), what problems could we run into?

When our data are simultaneously accessed by many users, we need locks to protect them. It is not an simple issue.

- A lock is like a signal that controls traffic. The signal is intended to prevent collision, only one user can "write" at a given time.

These issues are related to how big the data is and how often we access the data.

Relational DBMS Example

So, we need a Relational DBMS

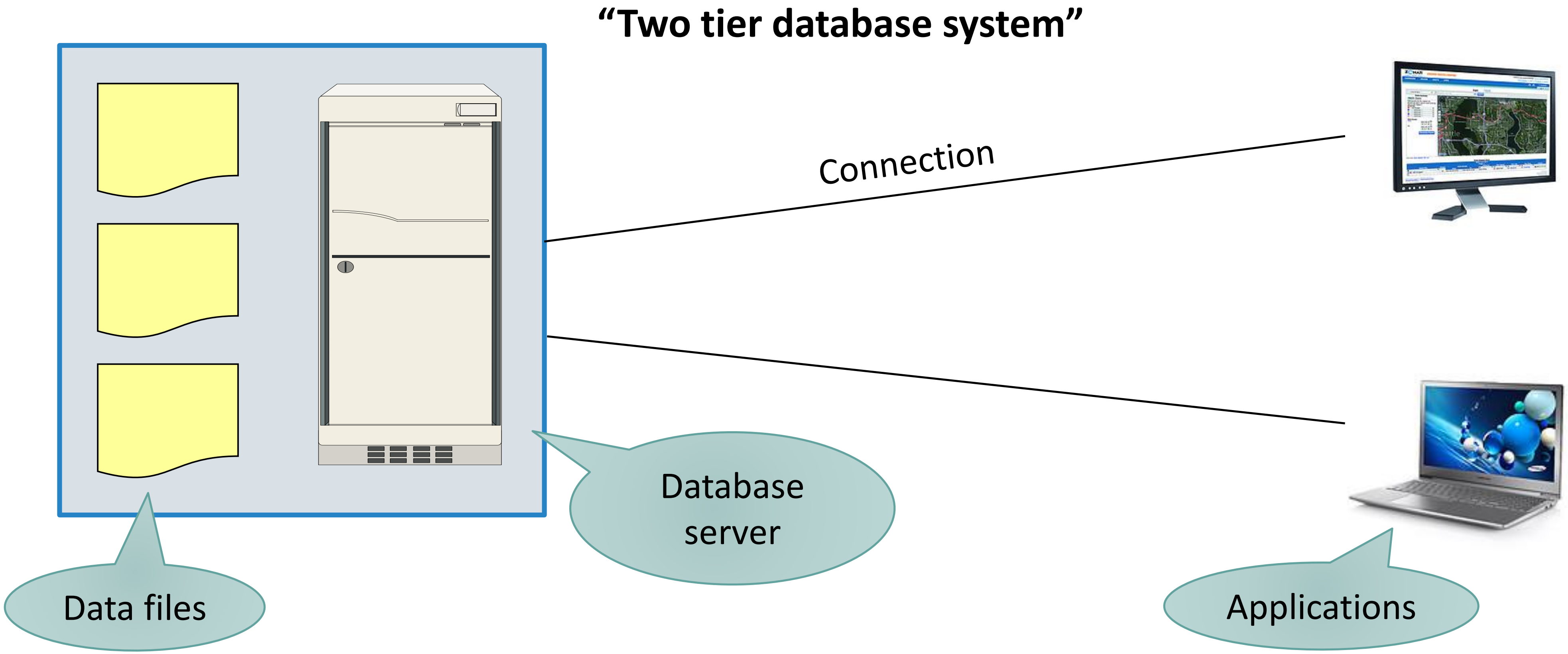


Image credit: TruckPR via Flickr, SamsungTomorrow via Flickr

Relational Database Example

Tables:

Students

SID	Name	Category
1312345	John	Undergrad
1623456	Mary	Grad
…	…	…

Takes

SID	CID
123-45-6789	CEE412
234-56-7890	CEE591
…	…

Professors

PID	Name	Category
98765	Yinhai	Professor
87654	Devyn	Asst. Professor
…	…	…

Courses

CID	Name	Enrollment	Professor
CEE412	Transportation Data Management	45	98765
CEE416	Urban Transportation Planning	35	87654
CEE591	Freight Transportation	11	76543
…	…	…	…

- Still implemented as files, but behind the scenes can be quite complex

What Else?

Other issues to consider:

- What happens if we lose power or communications during an update?
- What if a user makes a mistake in data entry?
- How to improve performance for large files or many concurrent users?
- How do we manage access and user credentials?
- Others…

Functionality of a DBMS

What are the capabilities that a DBMS provides to users?

- Persistent storage: durable, independent, flexible
- Programming interface allow data management through query language
- Transaction management: the ACID test

Lets go through these in order…

Persistent Storage

Persistent storage, why do we care?

Loss of connection or power:

- Data is fine. Any active transactions are either rolled back or permanent, never in between

A process which generates or collects data over time:

- Store it permanently as it is generated, rather than store in memory and output text files at intervals

Query Processing

A DBMS user uses SQL, (Structured Query Language) which has two components:

- Data Definition Language - DDL
- Data Manipulation Language - DML

→ Query language

Behind the scenes the DBMS has:

- Query optimizer
- Query engine
- Storage management
- Transaction Management (concurrency, recovery)

Query Processing

Problem:

- Given a database with multiple tables and relationships, find all courses that “John” takes.

How to answer this question?

What happens behind the scene?

Query processor figures out how to answer the query efficiently

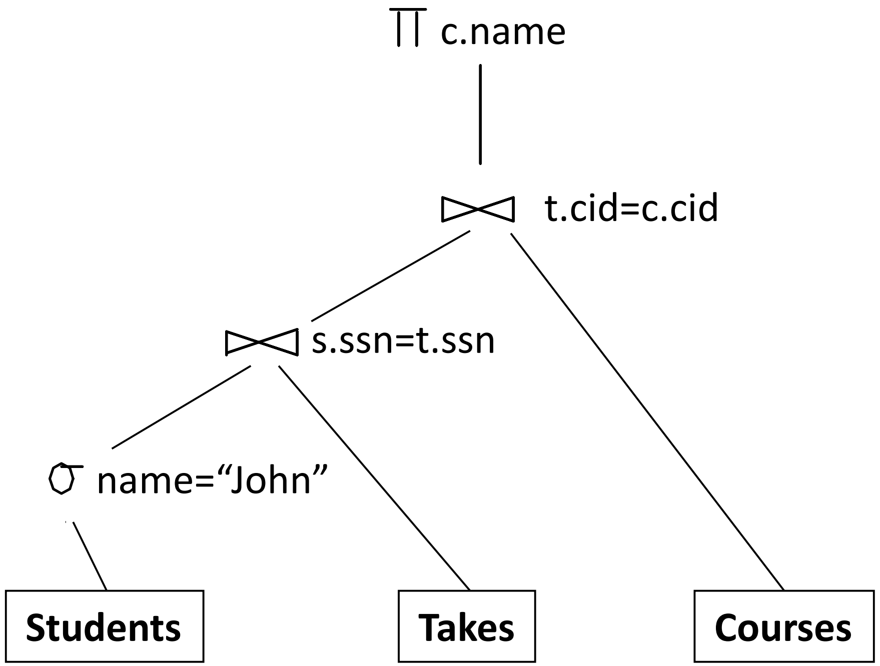
A query is parsed, preprocessed, and optimized by a query compiler

Query Processing

DECLARATIVE SQL query → IMPERATIVE query execution plan

```
SELECT   C.name
FROM     Students S, Takes T, Courses C
WHERE    S.name = “John” and
         S.ssn = T.ssn and T.cid = C.cid
```

The optimizer chooses the best execution plan for a query



Transaction Processing

A transaction is a unit of work, typically one or more database operations, that must be executed atomically and in apparent isolation from other transactions.

Examples are:

- Withdraw money from ATM
- Trade stocks on line
- Register a class through myUW
- …

Transaction Processing

The ACID Properties of Transactions:

Properly implemented transactions are commonly said to meet the "ACID test":

- "A" stands for "atomicity" - All or nothing
- "C" stands for "consistency" - All rules maintained
- "I" stands for "isolation" - All transactions executed apparently independently
- "D" stands for "durability" - Robust to power loss, failure of various types

Functionality of a DBMS

Two things to remember:

Client-server architecture

- Can be slow
- Cumbersome connection
- But good for the data

It is just someone else's C program

- We may be impressed by its speed
- But it can be frustratingly slow

Functionality of a DBMS

Big commercial database vendors:

- Oracle
- IBM (with DB2)
- Microsoft (SQL Server)
- Sybase

Some free and open-source database systems:

- MySQL
- PostgreSQL
- SQLite

Data Warehousing

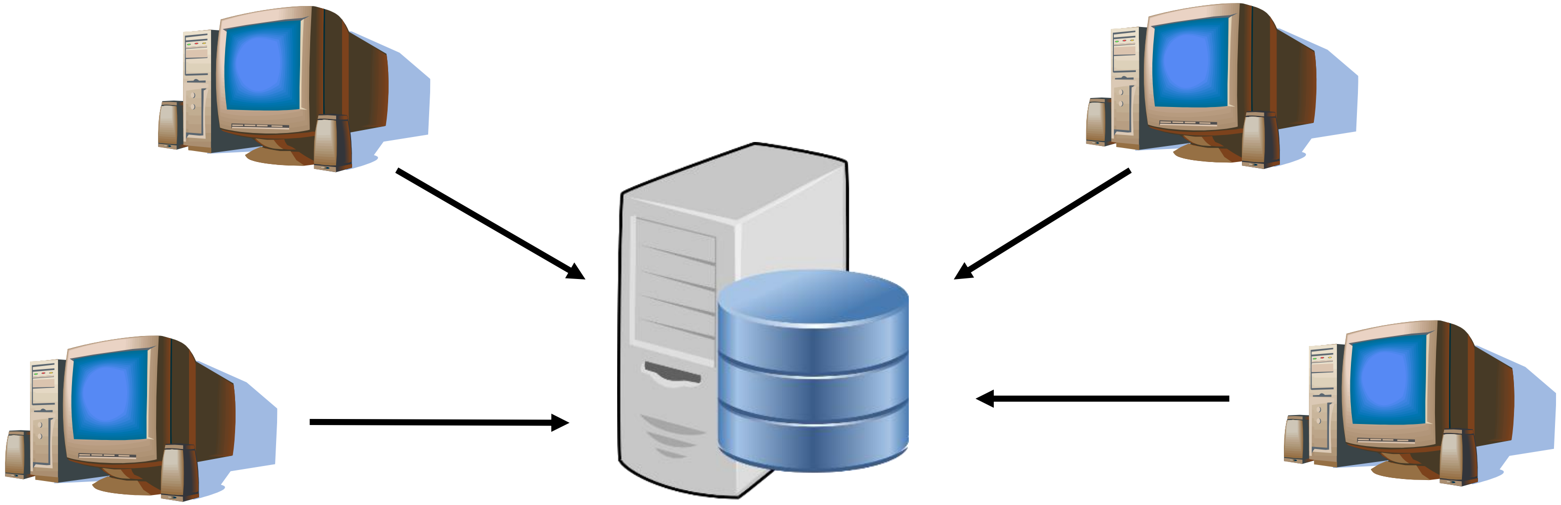
A common term, what is a data warehouse?

A data warehouse is a copy of transaction data specifically structured for querying and reporting

Why do we need data warehouses?

- Decision making support
- Information inquiry and sharing
- Data analysis

Data Warehousing



Data Warehouse

Each database may use different terms and DBMS

All data is integrated into centralized repository

A data warehouse is updated as the legacy databases change, but not necessarily in real-time

DBMS Applications

Where are DBMS used?

- Backend for traditional “database” applications
- Backend for large websites
- Backend for web services

Examples for transportation applications?

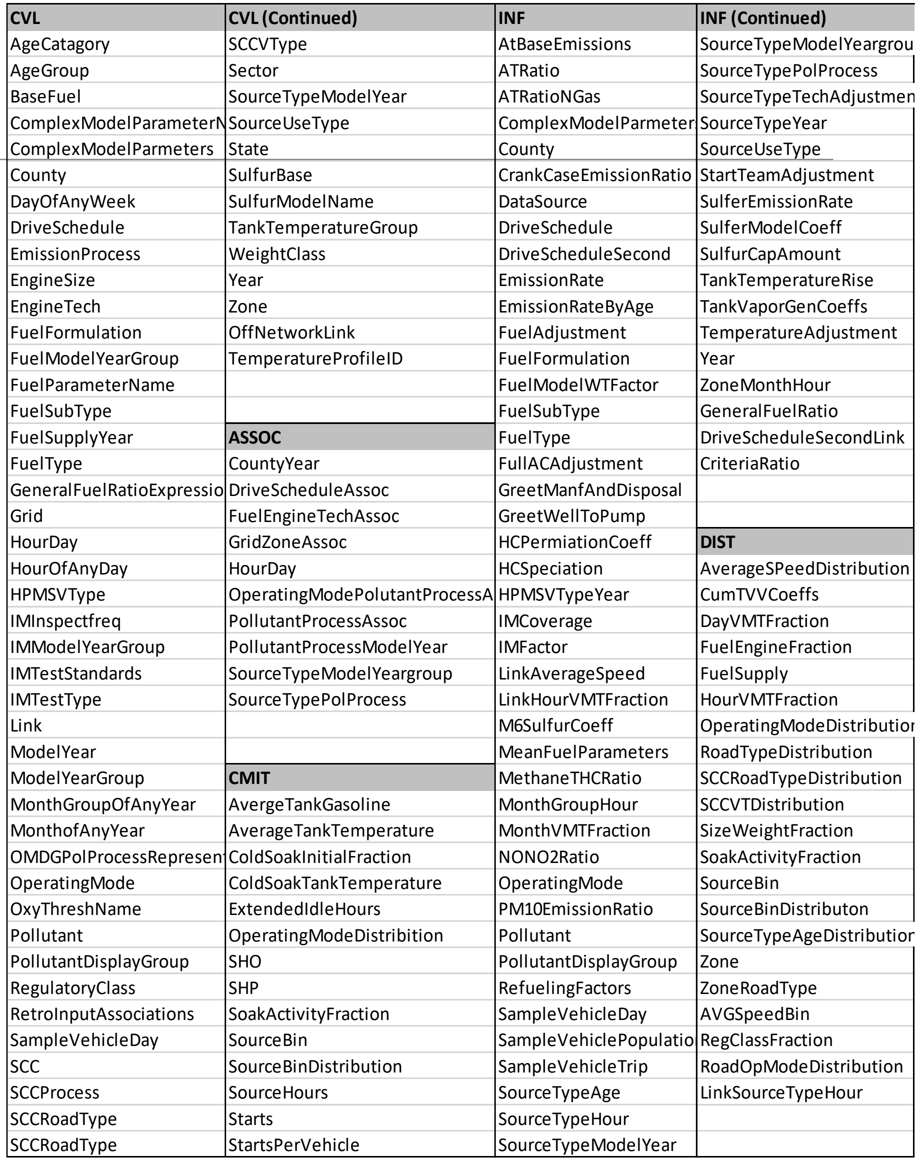
DBMS Applications

Environmental Protection Agency (EPA)'s MOVES (Motor Vehicle Emission Simulator)

- This Java application relies on a large collection of data with many relationships, and is supported by a MySQL database.
- Data generated over multiple years by multiple organizations, with many important relationships

Given the set of tables on the right:

- How (besides a relational database) could we build an application based on this data?
- How does the relational model make it easier to maintain these datasets?
- How could we understand it unless there was a carefully designed and documented data model?



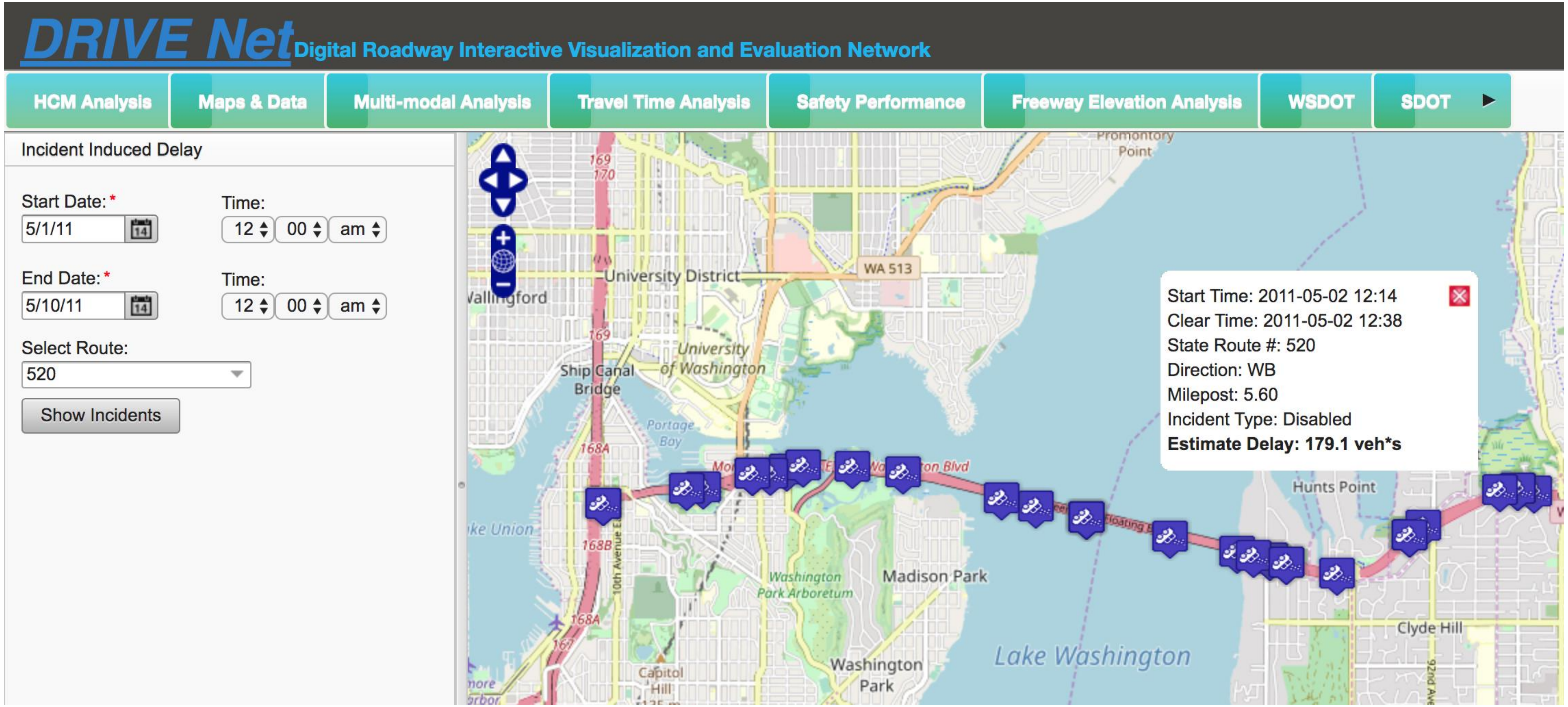
CVL	CVL (Continued)	INF	INF (Continued)
AgeCatagory	SCCVType	AtBaseEmissions	SourceTypeModelYeargrou
AgeGroup	Sector	ATRatio	SourceTypePolProcess
BaseFuel	SourceTypeModelYear	ATRatioNGas	SourceTypeTechAdjustmen
ComplexModelParameterN	SourceUseType	ComplexModelParmeter	SourceTypeYear
ComplexModelParmeters	State	County	SourceUseType
County	SulfurBase	CrankCaseEmissionRatio	StartTeamAdjustment
DayOfAnyWeek	SulfurModelName	DataSource	SulferEmissionRate
DriveSchedule	TankTemperatureGroup	DriveSchedule	SulferModelCoeff
EmissionProcess	WeightClass	DriveScheduleSecond	SulfurCapAmount
EngineSize	Year	EmissionRate	TankTemperatureRise
EngineTech	Zone	EmissionRateByAge	TankVaporGenCoeffs
FuelFormulation	OffNetworkLink	FuelAdjustment	TemperatureAdjustment
FuelModelYearGroup	TemperatureProfileID	FuelFormulation	Year
FuelParameterName		FuelModelWTFactor	ZoneMonthHour
FuelSubType		FuelSubType	GeneralFuelRatio
FuelSupplyYear	**ASSOC**	FuelType	DriveScheduleSecondLink
FuelType	CountyYear	FullACAdjustment	CriteriaRatio
GeneralFuelRatioExpressio	DriveScheduleAssoc	GreetManfAndDisposal	
Grid	FuelEngineTechAssoc	GreetWellToPump	
HourDay	GridZoneAssoc	HCPermiationCoeff	**DIST**
HourOfAnyDay	HourDay	HCSpeciation	AverageSPeedDistribution
HPMSVType	OperatingModePolutantProcessA	HPMSVTypeYear	CumTVVCoeffs
IMInspectfreq	PollutantProcessAssoc	IMCoverage	DayVMTFraction
IMModelYearGroup	PollutantProcessModelYear	IMFactor	FuelEngineFraction
IMTestStandards	SourceTypeModelYeargroup	LinkAverageSpeed	FuelSupply
IMTestType	SourceTypePolProcess	LinkHourVMTFraction	HourVMTFraction
Link		M6SulfurCoeff	OperatingModeDistributior
ModelYear		MeanFuelParameters	RoadTypeDistribution
ModelYearGroup	**CMIT**	MethaneTHCRatio	SCCRoadTypeDistribution
MonthGroupOfAnyYear	AvergeTankGasoline	MonthGroupHour	SCCVTDistribution
MonthofAnyYear	AverageTankTemperature	MonthVMTFraction	SizeWeightFraction
OMDGPolProcessRepresent	ColdSoakInitialFraction	NONO2Ratio	SoakActivityFraction
OperatingMode	ColdSoakTankTemperature	OperatingMode	SourceBin
OxyThreshName	ExtendedIdleHours	PM10EmissionRatio	SourceBinDistributon
Pollutant	OperatingModeDistribition	Pollutant	SourceTypeAgeDistributior
PollutantDisplayGroup	SHO	PollutantDisplayGroup	Zone
RegulatoryClass	SHP	RefuelingFactors	ZoneRoadType
RetroInputAssociations	SoakActivityFraction	SampleVehicleDay	AVGSpeedBin
SampleVehicleDay	SourceBin	SampleVehiclePopulatio	RegClassFraction
SCC	SourceBinDistribution	SampleVehicleTrip	RoadOpModeDistribution
SCCProcess	SourceHours	SourceTypeAge	LinkSourceTypeHour
SCCRoadType	Starts	SourceTypeHour	
SCCRoadType	StartsPerVehicle	SourceTypeModelYear	

Database Design

Procedure:

- Requirements modeling (conceptual, pictures)
  - Decide what entities should be part of the application and how they should be linked.
- Schema design and implementation (logical, physical)
  - Decide a set of tables, attributes.
  - Define the tables in the database system.
  - Populate database (insert tuples).
- Write application programs using the DBMS
  - Way easier with the data management taken care of.

Example: Incident Induced Delay



Incident Induced Delay

Start Date:* 5/1/11 Time: 12 00 am

End Date:* 5/10/11 Time: 12 00 am

Select Route: 520

Show Incidents

Start Time: 2011-05-02 12:14
Clear Time: 2011-05-02 12:38
State Route #: 520
Direction: WB
Milepost: 5.60
Incident Type: Disabled
Estimate Delay: 179.1 veh*s

Example: Incident Induced Delay

Application to quantify the delay attributable to incidents

At minimum I need:

- Incident/Accident data: time, location, type, etc.
- Traffic volume and speed: location, time, volume, speed, etc.

I need relationships between different traffic data and accidents

In this case, the relationships are defined by location and time

Example: Incident Induced Delay

How does this design look?

Incidents

Date	Time	Route	Milepost
12/22/2012	8:22:35.00	005	171.30
11/31/2012	8:12:22.00	005	156.90
10/11/2012	11:58:21.00	167	19.05

Detectors

ID_Number	Route	Direction	Lane	Milepost
10031	005	Increasing	1	145.1
10032	005	Increasing	2	145.1
10033	005	Increasing	3	145.1

Loop_Data

Date	Time	ID_Number	Occupancy	Volume	Speed
12/1/2012	13:22:35.00	10031	3.3	3	68.18
12/1/2012	4:12:22.55	10031	3.0	2	50.68
12/1/2012	22:58:21.21	10031	8.0	6	56.25

Consider the form of the data and how it relates to the requirements of the application…

Database Design

Considerations for database design:

- Agree on structure of the database before deciding on a particular implementation
- Store data in an optimal way

Consider issues such as:

- What entities to model
- How entities are related
- What constraints exist in the domain
- How to achieve good designs

Methods for Database Design

Entity/Relationship model (E/R):

- More relational in nature

Object Definition Language (ODL):

- Closer in spirit to object-oriented models
- Not covered in this class

Both can be translated (semi-automatically) to relational schemas

Database Jargon

Entity:

- An entity is a single object about which data can be stored. It is the "subject" of a table. Entities and their interrelationships are modeled through the use of entity-relationship diagrams.

Attribute:

- A single data item related to a database object. The database schema associates one or more attributes with each database entity. Attribute is also known as field or column.

Database Jargon

Set:

- A collection of objects, known as the elements of the set, specified in such a way that we can tell in principle whether or not a given object belongs to it. Order and repetition of elements within the set are irrelevant so, for example, {1, 2, 3} = {3, 2, 1} = {1, 3, 1, 2, 2}.

List:

- A data structure holding many values, possibly of different types, which is usually accessed sequentially, working from the head to the end of the tail - an "ordered list".

Are lists {1, 3, 5} and {3, 5, 1} identical?

Database Jargon

Metadata:

- Metadata is literally "data about data." This term refers to information about data itself.

Schema:

- A collection of metadata that describes the relations in a database. It can be simply described as the "layout" of a database or the blueprint that outlines the way data is organized into tables.

Tuple:

- A single record, a row in a table or relation when implemented

Database Jargon

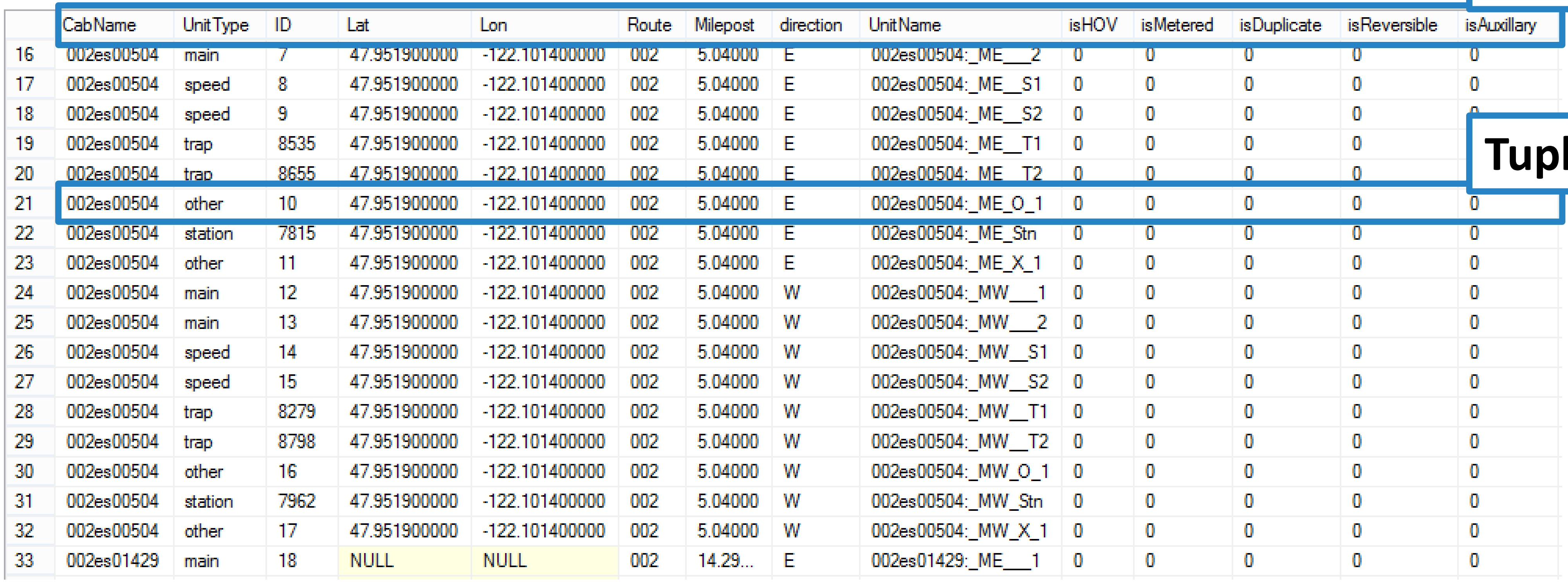
Relation:

- An unordered collection of tuples in some domain, a database table when implemented

Relationship:

- Not the same as a Relation
- Describes the relationship between entity sets or relations

Database Jargon



Attributes

Tuple

	CabName	UnitType	ID	Lat	Lon	Route	Milepost	direction	UnitName	isHOV	isMetered	isDuplicate	isReversible	isAuxillary
16	002es00504	main	7	47.951900000	-122.101400000	002	5.04000	E	002es00504:_ME___2	0	0	0	0	0
17	002es00504	speed	8	47.951900000	-122.101400000	002	5.04000	E	002es00504:_ME__S1	0	0	0	0	0
18	002es00504	speed	9	47.951900000	-122.101400000	002	5.04000	E	002es00504:_ME__S2	0	0	0	0	0
19	002es00504	trap	8535	47.951900000	-122.101400000	002	5.04000	E	002es00504:_ME__T1	0	0	0	0	[illegible]
20	002es00504	trap	8655	47.951900000	-122.101400000	002	5.04000	E	002es00504:_ME__T2	0	0	0	0	[illegible]
21	002es00504	other	10	47.951900000	-122.101400000	002	5.04000	E	002es00504:_ME_O_1	0	0	0	0	0
22	002es00504	station	7815	47.951900000	-122.101400000	002	5.04000	E	002es00504:_ME_Stn	0	0	0	0	0
23	002es00504	other	11	47.951900000	-122.101400000	002	5.04000	E	002es00504:_ME_X_1	0	0	0	0	0
24	002es00504	main	12	47.951900000	-122.101400000	002	5.04000	W	002es00504:_MW___1	0	0	0	0	0
25	002es00504	main	13	47.951900000	-122.101400000	002	5.04000	W	002es00504:_MW___2	0	0	0	0	0
26	002es00504	speed	14	47.951900000	-122.101400000	002	5.04000	W	002es00504:_MW__S1	0	0	0	0	0
27	002es00504	speed	15	47.951900000	-122.101400000	002	5.04000	W	002es00504:_MW__S2	0	0	0	0	0
28	002es00504	trap	8279	47.951900000	-122.101400000	002	5.04000	W	002es00504:_MW__T1	0	0	0	0	0
29	002es00504	trap	8798	47.951900000	-122.101400000	002	5.04000	W	002es00504:_MW__T2	0	0	0	0	0
30	002es00504	other	16	47.951900000	-122.101400000	002	5.04000	W	002es00504:_MW_O_1	0	0	0	0	0
31	002es00504	station	7962	47.951900000	-122.101400000	002	5.04000	W	002es00504:_MW_Stn	0	0	0	0	0
32	002es00504	other	17	47.951900000	-122.101400000	002	5.04000	W	002es00504:_MW_X_1	0	0	0	0	0
33	002es01429	main	18	NULL	NULL	002	14.29...	E	002es01429:_ME___1	0	0	0	0	0

Database Jargon

Entities:

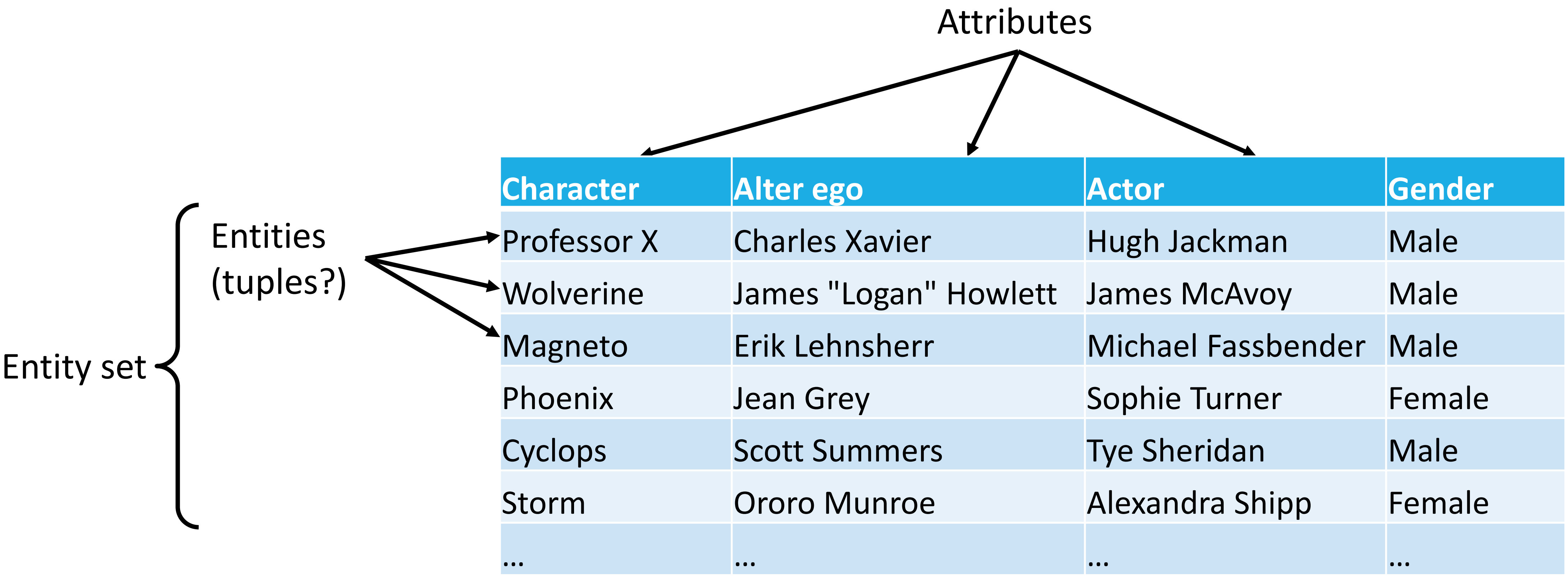
Uniquely identifiable "things" about which we want to store information

Attributes:

Information we want to store about entities, properties of entities

Name
Actor
Gender
Birthdate
Etc.

Database Jargon



Attributes

Entity set

Entities (tuples?)

Character	Alter ego	Actor	Gender
Professor X	Charles Xavier	Hugh Jackman	Male
Wolverine	James "Logan" Howlett	James McAvoy	Male
Magneto	Erik Lehnsherr	Michael Fassbender	Male
Phoenix	Jean Grey	Sophie Turner	Female
Cyclops	Scott Summers	Tye Sheridan	Male
Storm	Ororo Munroe	Alexandra Shipp	Female
…	…	…	…

“Tuple” and “Entity” are not interchangeable terms. Why?

Tuple = record or table row, Entity = conceptual thing or event