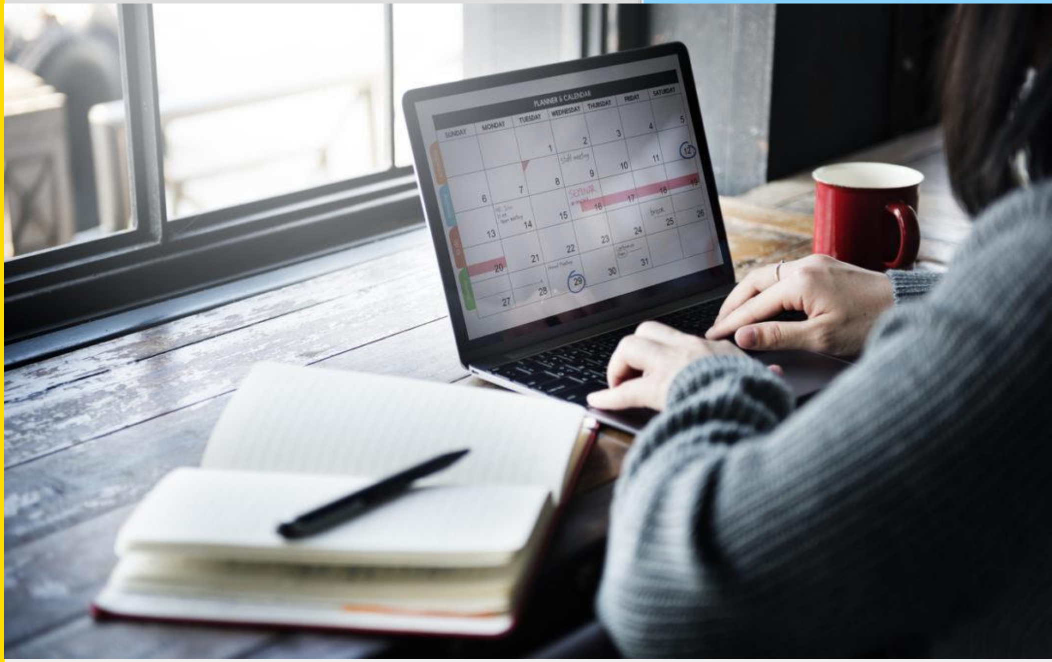
Welcome to ZZEN9313 Big Data Management



- We'll be starting at 7:30 pm
- In preparation for this webinar please check that your microphone is muted.
- We'll be taking questions by chat

This webinar will be recorded and made available for your ongoing reference.

UNSW SYDNEY

Week 1's Tutor: Daria Schumm,
d.schumm@student.unsw.edu.au

Week 1's QA Sessions:

Wednesday 7—8 pm

Thursday 8—9 pm

Friday 7—8 pm

Saturday 8—9 pm

Part 1: Introduction to Big Data

What is Big Data?

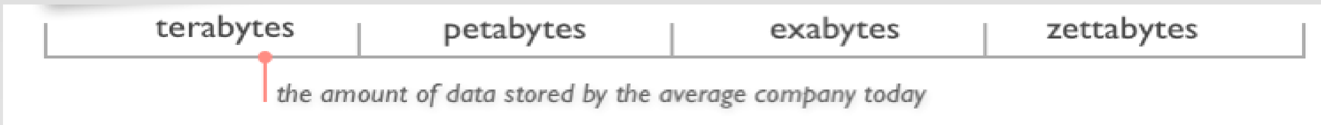
- No standard definition! here is from Wikipedia:
  - Big data is a field that treats ways to analyze, systematically extract information from, or otherwise deal with data sets that are too large or complex to be dealt with by traditional data-processing application software
  - Challenges include capturing data, data storage, data analysis, search, sharing, transfer, visualization, querying, updating, information privacy, and data source..
  - The term "big data" often refers simply to the use of predictive analytics, user behavior analytics, or certain other advanced data analytics methods that extract value from big data, and seldom to a particular size of data set.

Big Data Characteristics: 3V

- The Vs of big data were often referred to as the "three Vs"
  - **Volume**: In a big data environment, the amounts of data collected and processed are much larger than those stored in typical relational databases.
  - **Variety**: Big data consists of a rich variety of data types.
  - **Velocity**: Big data arrives to the organization at high speeds and from multiple sources simultaneously.

Volume (Scale)

- In the big data era, huge amount of data is being generated every day



Recent Twitter statistics

Quick Twitter Statistics

Total Number of Monetizable Daily Active Users:

217 million (source)

Last updated: 21/02/22

Total Number of Tweets Sent per Day:

500 million (source)

Last updated: 21/02/22

Q4 2022 Total Twitter Revenue:

$1.57 billion (source)

Last updated: 21/02/22

The number of US Adults Who Use Twitter:

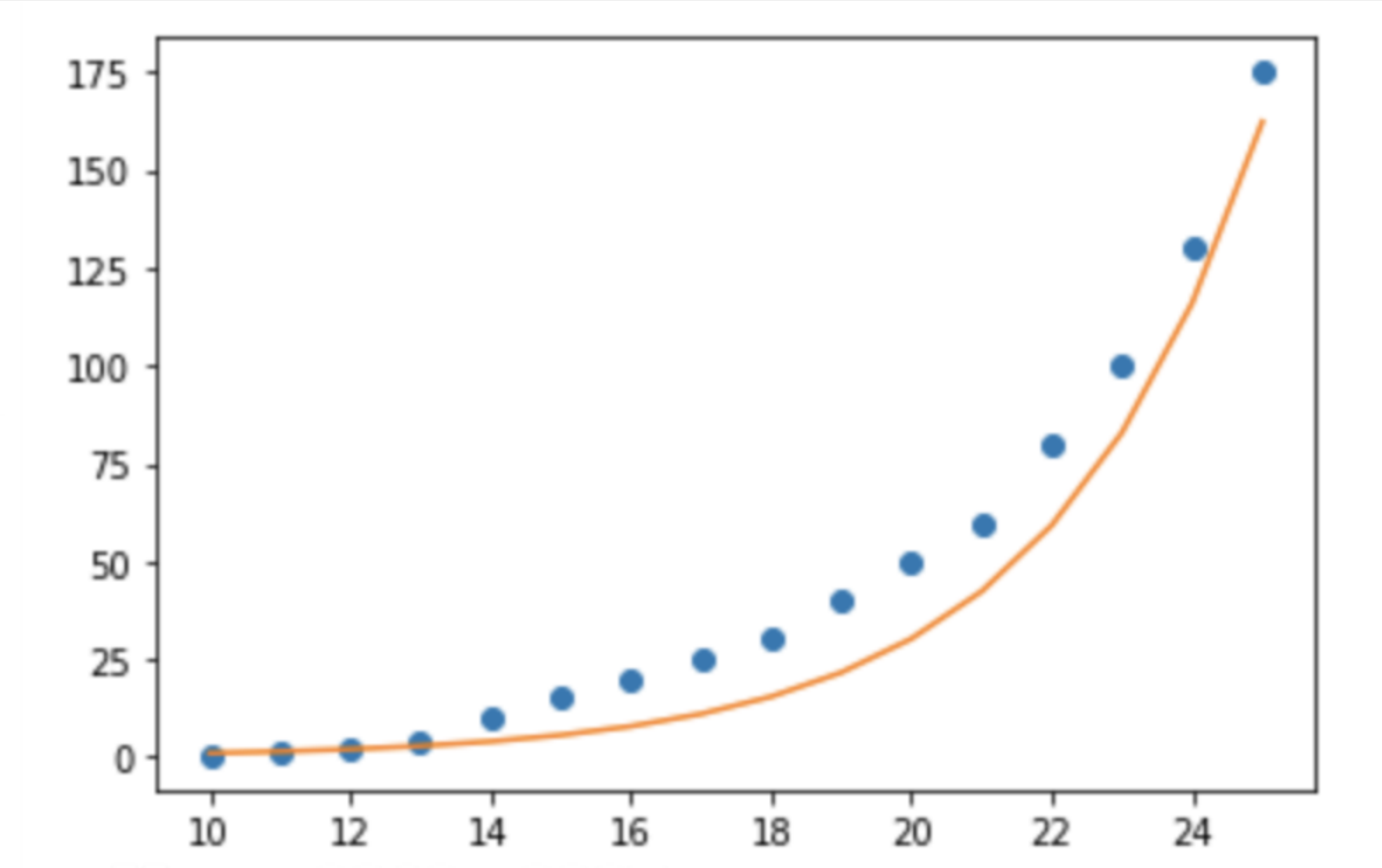
23% (source)

Last updated: 21/02/22

https://www.omnicoreagency.com/twitter-statistics/

Volume (Scale)

- Data volume is increasing exponentially (40% increase per year)



Data amount in Zetabytes from 2010 to 2025

[A forecast by IDC & SeaGate. Image by Sven Balnojan.]

Variety (Complexity)

- Different Types:
  - Relational Data (Tables/Transaction/Legacy Data)
  - Text Data (Web)
  - Semi-structured Data (XML)
  - Spatial Data
  - Temporal Data
  - Graph Data
    - Social Network, Semantic Web (RDF), …
  - One application can be generating/collecting many types of data
- Different Sources:
  - Movie reviews from IMDB and Rotten Tomatoes
  - Product reviews from different provider websites

To extract knowledge→ all these types of data need to linked together

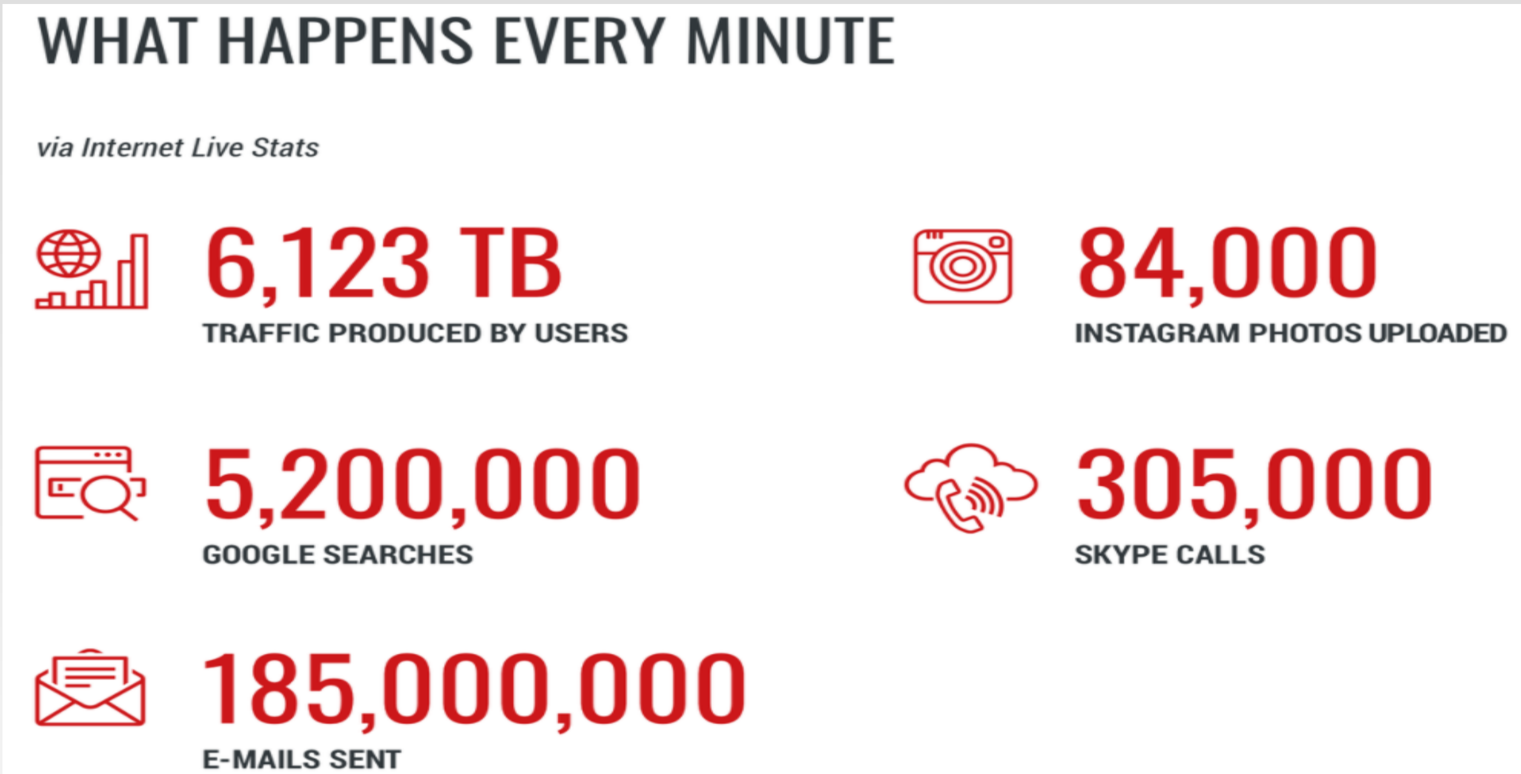
A Global View of Linked Big Data

Diversified social network

Velocity (Speed)

- Velocity essentially measures how fast the data is coming in.
- Data is being generated fast and need to be processed fast
  - Late decisions -> missing opportunities
- It is usually met in online data analytics, for example
  - **E-Promotions**: based on your current location, your purchase history, what you like -> send promotions right now for store next to you
  - **Healthcare monitoring**: sensors monitoring your activities and body -> any abnormal measurements require immediate reaction
  - **Pandemic management and response**: contact tracing for new infected COIVD-19 cases and future hotspots prediction to slow down the spread of infectious diseases

Velocity in Real-world



- The statistics for 1 second in many applications. http://www.internetlivestats.com/one-second/

Extended Big Data Characteristics: 6V

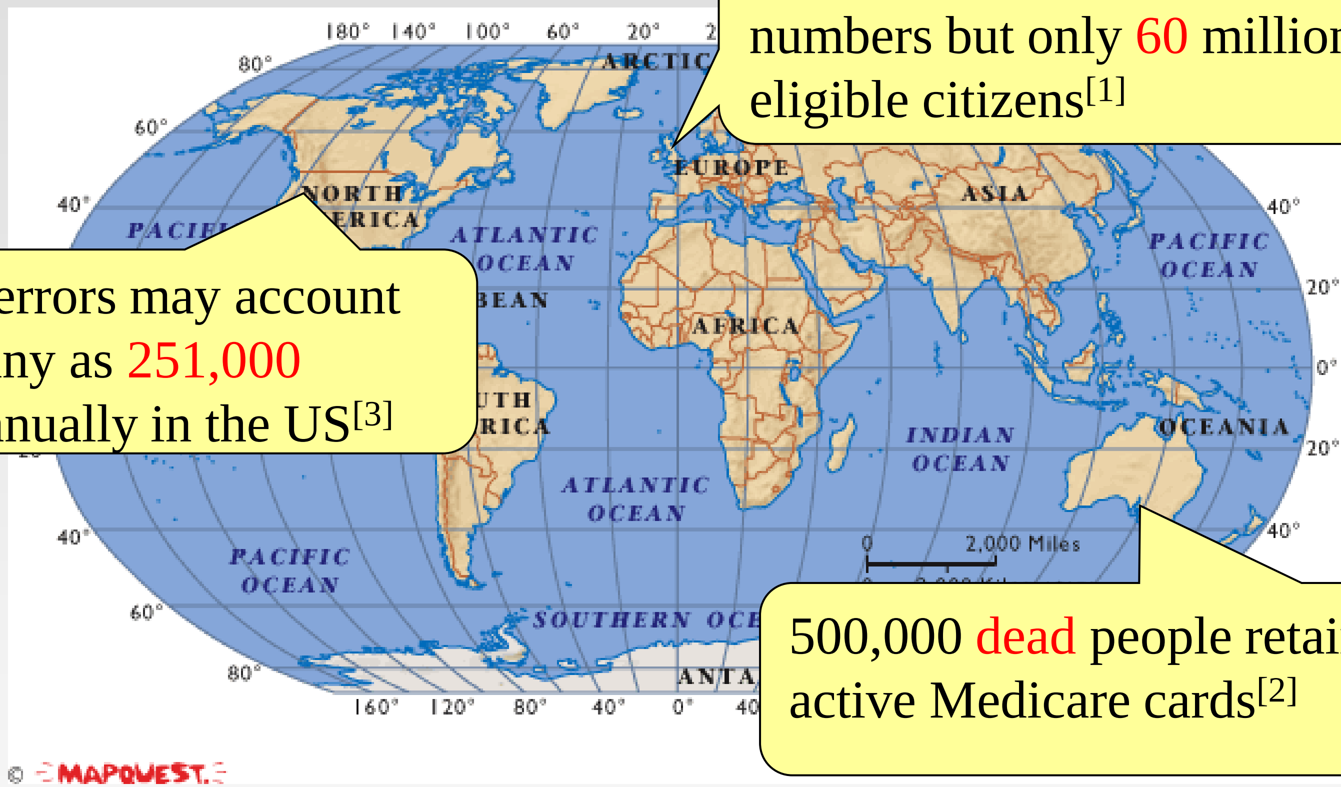
- Volume: In a big data environment, the amounts of data collected and processed are much larger than those stored in typical relational databases.
- Variety: Big data consists of a rich variety of data types.
- Velocity: Big data arrives to the organization at high speeds and from multiple sources simultaneously.

- Veracity: Data quality issues are particularly challenging in a big data context.
- Value: Ultimately, big data is meaningless if it does not provide value toward some meaningful goal.
- Visibility/Visualization/Variability/Validity/….

Veracity (Quality & Trust)

- Data = quantity + quality
- When we talk about big data, we typically mean its quantity:
  - What capacity of a system provides to cope with the sheer size of the data?
  - Is a query feasible on big data within our available resources?
  - How can we make our queries tractable on big data?
  - . . .
- Can we trust the answers to our queries?
  - Dirty data routinely lead to misleading financial reports, strategic business planning decision -> loss of revenue, credibility and customers, disastrous consequences
- *The study of data quality is as important as data quantity*

Data in real-life is often dirty



81 million National Insurance numbers but only 60 million eligible citizens[1]

Medical errors may account for as many as 251,000 deaths annually in the US[3]

500,000 dead people retain active Medicare cards[2]

[1] https://publications.parliament.uk/pa/cm200001/cmhansrd/vo010327/debtext/10327-21.htm
[2] https://www.privacy.org.au/Campaigns/ID_cards/MedicareSmartcard.html
[3] Your Health Care May Kill You: Medical Errors. https://pubmed.ncbi.nlm.nih.gov/28186008/

Value

- Big data is meaningless if it does not provide value toward some meaningful goal

Other V's

- Visibility: the state of being able to see or be seen is implied.
  - Big Data – visibility = Black Hole?

- Visualization: Making all that vast amount of data comprehensible in a manner that is easy to understand and read.

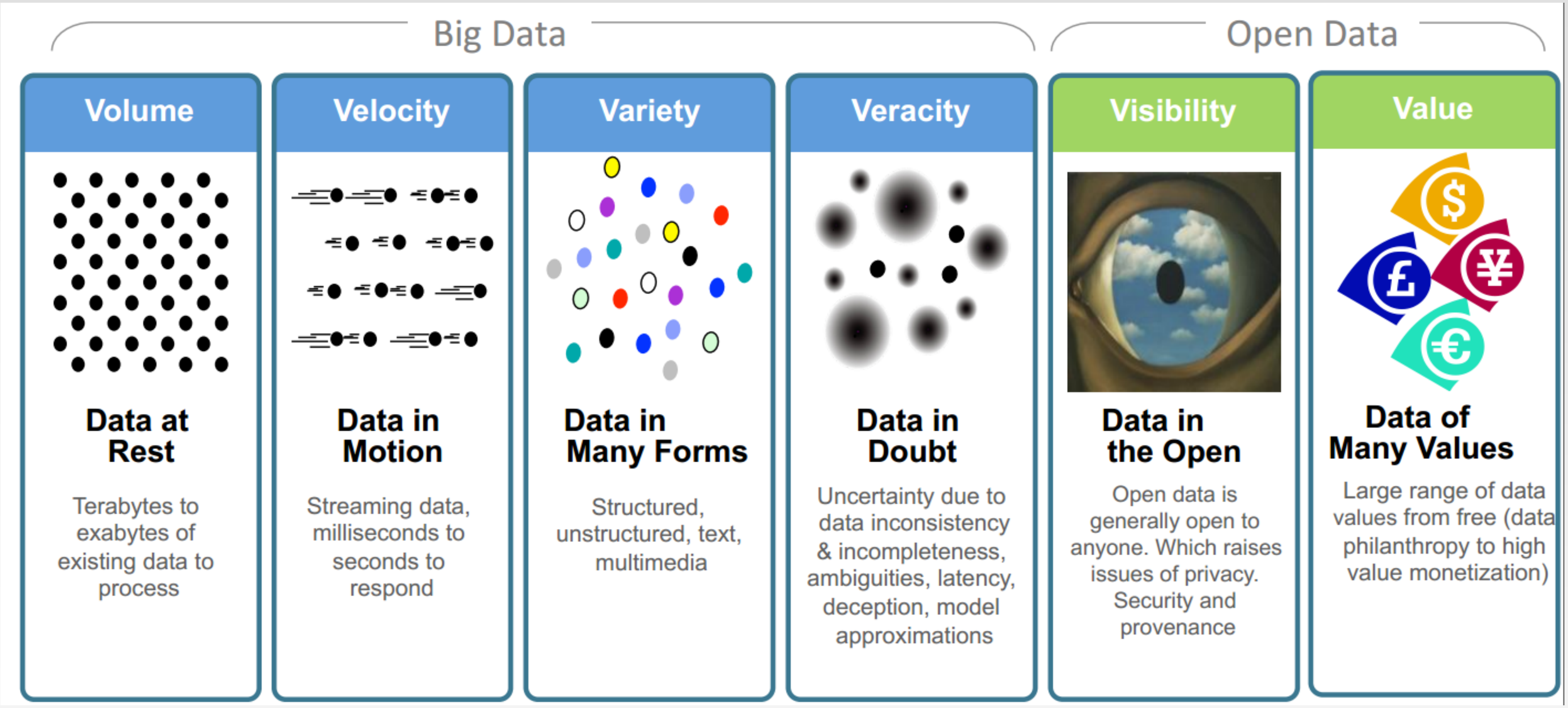
A visualization of Divvy bike rides across Chicago

- Big data visualization tools:

Other V's

- Variability
  - Variability refers to data whose meaning is constantly changing. This is particularly the case when gathering data relies on language processing.
  - It defines the need to get meaningful data considering all possible circumstances.
- Viscosity
  - This term is sometimes used to describe the latency or lag time in the data relative to the event being described. We found that this is just as easily understood as an element of Velocity.
- Volatility
  - Big data volatility refers to how long is data valid and how long should it be stored. You need to determine at what point is data no longer relevant to the current analysis.
- More V's in the future …
  - How many v's are there in big data? http://www.clc-ent.com/TBDE/Docs/vs.pdf

Big Data: 6V in Summary



Transforming Energy and Utilities through Big Data & Analytics. By Anders Quitzau@IBM

Big Data Open-Source Tools

https://datafloq.com/big-data-open-source-tools/os-home/

Part 2: Introduction to RDBMS

Structure of a Relational Database

- Relation model was introduced in 1970 by Dr. E. F. Codd (of IBM)
- A relational database basically consists of a number of tables, each of which represents a relation instance, or simply relation.
- Each table contains a number of rows and a number of columns.
- Each row represents a tuple of a relation.
- Each column corresponds to an attribute of a relation.
- Each table specifies the current value of the relation and hence is called a *relation instance* (tuples can be stored in any order).

attributes
(or columns)

customer_name	***customer_street***	***customer_city***
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

tuples
(or rows)

customer

Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: *instructor* relation with unordered tuples

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Database Schema

- The schema of a table/relation is the the structure of the table/relation, called a *relation schema*.
- The structure of a relational database is specified by a database schema, which contains a set of relation schemas.
- Each relation schema consists of a relation name and a number of attributes.
- Each attribute has a particular attribute type (similar to data types in programming), that is, a domain of values.

LicenseNumber	Name	Expiry
15063558	Harry Pogson	2020-12-20
22531555	Selina Chua	2021-06-07
13541235	Yu Hou	2020-07-07

```
CREATE TABLE Citizens {
    LicenseNumber TEXT PRIMARY_KEY,
    Name TEXT,
    Expiry DATE
}
```

Schema

Attribute Types

- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
  - E.g. the value of an attribute can be an account number, but cannot be a set of account numbers
- The special value *null* is a member of every domain

Keys – Why and What

- Each tuple in a relation/table needs to be uniquely identified, e.g., a tuple may represent a student record, a module, or an employee record
- Simply put, a key is an attribute that identifies a unique tuple of each possible relation *r(R)*, e.g., K = {*ID*}.
- A key can be a combination of attributes, say K $\subseteq$ R = (A_1, A_2, …, A_n). *K* is called a **superkey** of R, if values for K are sufficient to uniquely identify a unique tuple of each possible relation *r(R)*.
  - E.g., K= {*ID, name*}.
  - By "possible *r* " we mean a relation *r* that could possibly exist, given the relation schema, rather than just what you currently have in the database.

    E.g., K={*name*} is not a superkey
  - Both K= {*ID*} and K= {*ID, name*} can be superkeys

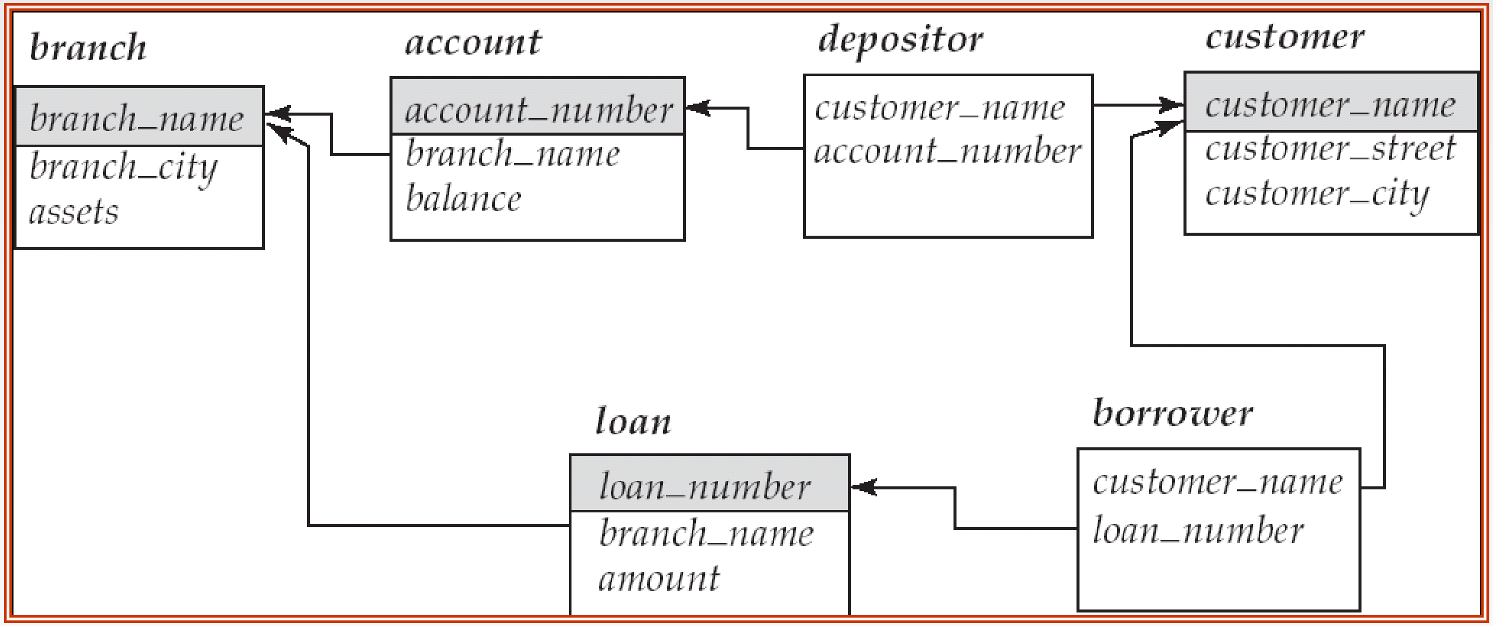
Keys (Cont.)

- *K* is a **candidate key** if *K* is minimal
  Example: K= {*ID*} is a candidate key for *instructor*, since it is a superkey and no subset of it is a superkey.
- **Primary key:** a candidate key chosen as the principal means of identifying tuples within a relation
  - Should choose an attribute whose value never, or very rarely, changes.
    - Both national insurance number and email address is unique, which one should be used as the primary key?
  - We normally underline the primary key

  For instance, *instructor(<u>ID</u>, name, dept_name, salary)*

Foreign Keys

- The attribute of a relation schema attribute is called a **foreign key** if it corresponds to the primary key of another relation schema.
  - E.g. *customer_name* and *account_number* attributes in *depositor* are foreign keys that are the primary keys of *customer* and *account* respectively.
  - Only values occurring in the primary key attribute of the **referenced relation** may occur in the foreign key attribute of the **referencing relation**.
- **Schema diagram** – show the connections between relation schemas



Database

- A database consists of multiple relations
- Information about an enterprise is broken up into parts

  instructor
  student
  advisor

- Bad design:

  univ (*instructor -ID, name, dept_name, salary, student_Id*, ..)

  results in
  - repetition of information (e.g., two students have the same instructor)
  - the need for null values (e.g., represent an student with no advisor)
- Normalization theory deals with how to design "good" relational schemas

Step 1: Design a Database Schema

- The design is done either on paper or using a visual design tool. The goal is to design a set of good relation schemas.

instructor(<u>ID</u>, name, dept_name, salary)

department(<u>dept_name</u>, building, budget)

student(<u>ID</u>, name, dept_name, tot_credit)

advisor(<u>i_ID</u>, <u>s_ID</u>)

Step 2: Create the Structure of a Database from a Set of Relation Schemas Using a Database Definition Language (DDL)

instructor

ID	*name*	*dept_name*	*salary*

department

dept_name	*building*	*budget*

student

ID	*name*	*dept_name*	*tot_cred*

Step 3: Load Relational Databases with Initial Data (Cont.)

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Kaze	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Step 3: Load Relational Databases with Initial Data (Cont.)

department

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Step 3: Load Relational Databases with Initial Data (Cont.)

student

ID	*name*	*dept_name*	*tot_cred*
10128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

Step 4: Query Relational Databases

Query over a single table.

Make simple queries.

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Kaze	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

List of instructors in the Physics department.

Step 4: Query Relational Databases (Cont.)

Query over a single table.

Make simple queries.

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Kaze	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

List of instructors in the Physics department.

Step 4: Query Relational Databases (Cont.)

Query over a single table.

Make aggregate queries.

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Kaze	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

How many instructors are in the `Comp. Science department?

Step 4: Query Relational Databases (Cont.)

Query over a single table.

Make aggregate queries.

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Kaze	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

How many instructors are in the `Comp. Science department?

⇒ 3

Step 4: Query Relational Databases (Cont.)

Query over a single table.

Make aggregate queries.

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Kaze	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Which instructor earns the highest salary.

Step 4: Query Relational Databases (Cont.)

Query over a single table.

Make aggregate queries.

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Kaze	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Which instructor earns the highest salary.

Einstein

Step 4: Query Relational Databases (Cont.)

Query over multiple tables by joining them.

Which instructors work in the Watson building?

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Kaze	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

department

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Step 4: Query Relational Databases (Cont.)

Query over multiple tables by joining them.

Which instructors work in the Watson building?

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Kaze	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

department

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Step 4: Query Relational Databases (Cont.)

Query composability and nested queries.

The result of a query is a table itself. Hence another query can be made over the result table of the query.

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Kaze	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

List the name of all instructors who work in either the Comp. Sci. department or the Physics department.

First you make a query to get a table of all instructors in the Comp. Sci. department.

Step 4: Query Relational Databases (Cont.)

Query composability and nested queries.

The result of a query is a table itself. Hence another query can be made over the result table of the query.

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Kaze	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

List the name of all instructors who work in either the Comp. Sci. department or the Physics department.

First you make a query to get a table of all instructors in the Comp. Sci. department.

Step 4: Query Relational Databases (Cont.)

Query composability and nested queries.

The result of a query is a table itself. Hence another query can be made over the result table of the query.

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Kaze	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

List the name of all instructors who work in either the Comp. Sci. department or the Physics department.

First you make a query to get a table of all instructors in the Comp. Sci. department.

Next you make another query to get a table of all instructors in the Physics department.

Step 4: Query Relational Databases (Cont.)

Query composability and nested queries.

The result of a query is a table itself. Hence another query can be made over the result table of the query.

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Kaze	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

List the name of all instructors who work in either the Comp. Sci. department or the Physics department.

First you make a query to get a table of all instructors in the Comp. Sci. department.

Next you make another query to get a table of all instructors in the Physics department.

Step 4: Query Relational Databases (Cont.)

Query composability and nested queries.

The result of a query is a table itself. Hence another query can be made over the result table of the query.

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
22222	Einstein	Physics	95000
33456	Gold	Physics	87000
45565	Kaze	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

List the name of all instructors who work in either the Comp. Sci. department or the Physics department.

First you make a query to get a table of all instructors in the Comp. Sci. department.

Next you make another query to get a table of all instructors in the Physics department.

Finally you make a query over the two previously generated tables to get a table of instructors in either of these two departments.

Step 5: Modify Relational Databases

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Kaze	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

You can add a new row into a table.

You can delete a row from a table.

You can add a new column into a table.

You can delete a column row from a table.

You can modify the value of a cell.

Criteria for Good Design

- Desired properties of a relational schema
  - Achieve representation power, i.e., the relations in the schema should contain all the information conveyed by the attributes and constraints
  - Reduce redundancy
  - Avoid loss of information.

Redundancy means that when we **mutate** a database (i.e. insert a row, update a row, or delete a row) we may have to update it in multiple places.

FirstName	LastName	PhoneType	PhoneNumber
Jane	Johnson	Work	55578900
Jane	Johnson	Home	55563472
Peter	Simons	Work	55524111
Peter	Simons	Mobile	0455999111

E.G. What happens if "Peter" wants to update his name to "Paul"?

Schema Decomposition

FirstName	LastName	PhoneType	PhoneNumber
Jane	Johnson	Work	55578900
Jane	Johnson	Home	55563472
Peter	Simons	Work	55524111
Peter	Simons	Mobile	0455999111

FirstName	LastName
Jane	Johnson
Peter	Simons

PhoneType	PhoneNumber
Work	55578900
Home	55563472
Work	55524111
Mobile	0455999111

Now we have removed redundancy, however, these tables are not linked properly yet - so we need to link them.

Linking Tables

Now we need to link these tables together somehow

FirstName	LastName
Jane	Johnson
Peter	Simons

Primary Key

PersonId	FirstName	LastName
1	Jane	Johnson
2	Peter	Simons

PhoneType	PhoneNumber
Work	55578900
Home	55563472
Work	55524111
Mobile	0455999111

Foreign Key

PersonId	PhoneType	PhoneNumber
1	Work	55578900
1	Home	55563472
2	Work	55524111
2	Mobile	0455999111

Adding a primary key to the table

PhoneId	PersonId	PhoneType	PhoneNumber
1	1	Work	55578900
2	1	Home	55563472
3	2	Work	55524111
4	2	Mobile	0455999111

A Lossy Decomposition

ID	name	street	city	salary
⋮				
57766	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

employee

ID	name
⋮	
57766	Kim
98776	Kim
⋮	

name	street	city	salary
⋮			
Kim	Main	Perryridge	75000
Kim	North	Hampton	67000
⋮			

natural join

ID	name	street	city	salary
⋮				
57766	Kim	Main	Perryridge	75000
57766	Kim	North	Hampton	67000
98776	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

Database Normalization

- Database normalization is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity
- In this course we will explore different levels of normalised databases:
  - First normal form
  - Second normal form
  - Third normal form
- BCNF, 4NF, and 5NF can be explored further in independent learning

1NF: First Normal Form

- All values are atomic. There are no duplicate rows
- To normalize into 1NF:
  - For composite attributes or divisible attributes, make each component as a column
    - E.g., divide the course ID ZZEN9313 into two columns
  - For multivalued attributes:
    - Remove the multivalued attribute that violates 1NF and place it in a separate relation together with the primary key.
    - Or, if the max number of values is known, then we can replace the violate attribute by the max number atomic attributes. This incurs null values.

ID	name	phone
001	Jim	512-555-1234
002	Taylor	512-555-4567 512-555-6789

ID	name
001	Jim
002	Taylor

ID	phone
001	512-555-1234
002	512-555-4567
002	512-555-6789

ID	name	phone1	Phone2
001	Jim	512-555-1234	null
002	Taylor	512-555-4567	512-555-6789

2NF: Second Normal Form

- Is in 1NF Has no partial dependencies
- Partial dependency: When a non-key attribute depends on part of the primary key
- Only an issue for composite primary keys: If each candidate key of R contains a single attribute, R is in 2NF
- Normalization into 2NF:
  - Associate non-prime attributes with only the part of the candidate key on which they are functionally dependent
  - (*i_ID, s_ID*) is the candidate key; $i\_ID \rightarrow meeting\_place$

i_ID	s_ID	meeting_place
001	1234	Room 001
002	2342	Room 002
001	3465	Room 001
002	1234	Room 002
003	8992	Room 003

⇒

i_ID	s_ID
001	1234
002	2342
001	3465
002	1234
003	8992

\+

i_ID	meeting_place
001	Room 001
002	Room 002
003	Room 003

3NF: Third Normal Form

- Is in 2NF
- Has no non-key attributes transitively dependent on the primary key
  - Transitive dependency: Indirect relationship between columns in a table that causes a functional dependency
  - If non-key column C changes, which would cause another non-key column B to change, there is a transitive dependency

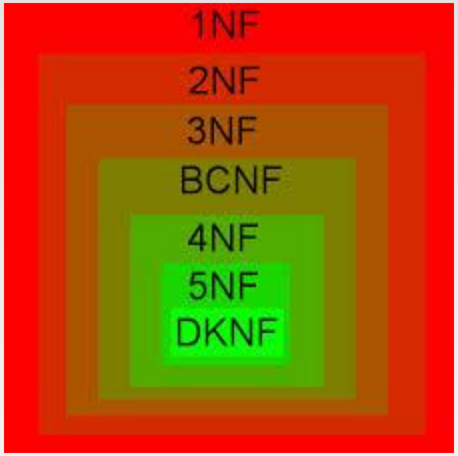
employee_id	name	suburb	postcode
123	John	seaview downs	5049
345	Ajeet	lismore	2480
567	Lora	seaview downs	5049
891	Lilly	tugan	4224

employee_id	name	suburb
123	John	seaview downs
345	Ajeet	lismore
567	Lora	seaview downs
891	Lilly	tugan

suburb	postcode
seaview downs	5049
lismore	2480
tugan	4224

Since { employee_id => suburb } and { suburb => postcode }, we have a transitive dependency and need to separate out

Relationships of Different Forms



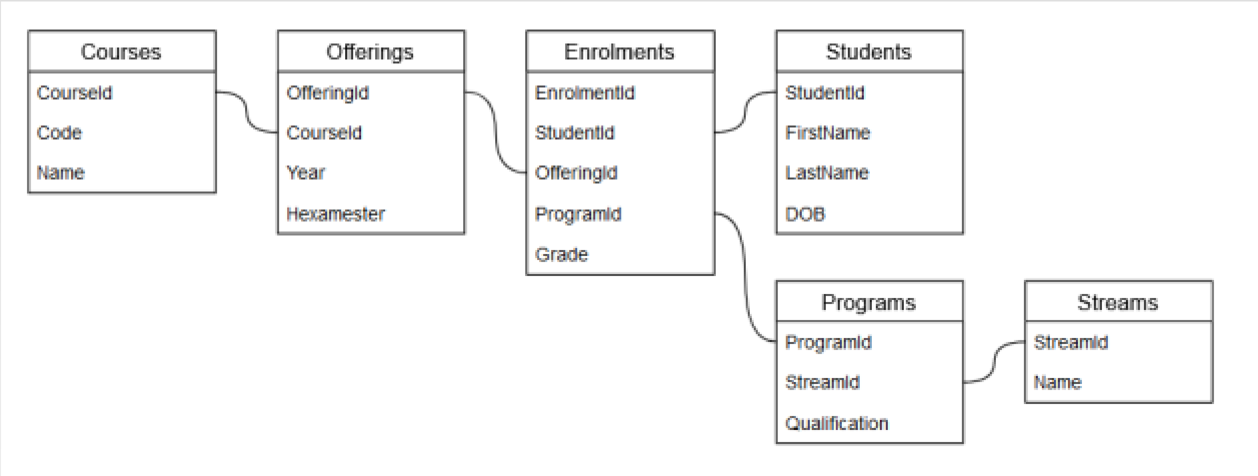
- First Normal Form
  - All attributes are atomic
- Second Normal Form
  - All attributes depend on every whole key
- Third Normal Form
  - All attributes depend on nothing but the key
- Boyce-Codd Normal Form
  - No redundancy based on functional dependency, but not functional dependency preserving

Data Warehouse

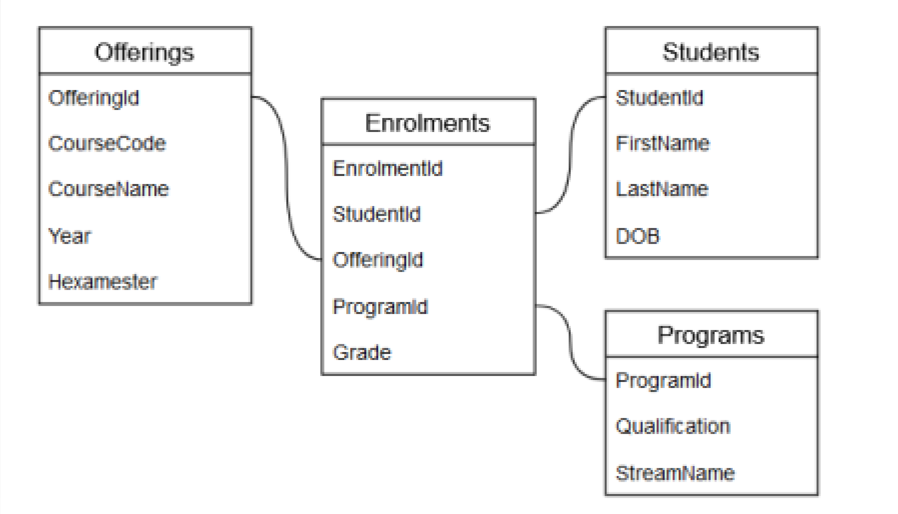
- Operational databases are optimised for OLTP (Online Transactional Processing)
- Data warehouse databases are optimised for OLAP (Online Analytical Processing)
- The typical process is that at some intermittent cadence (e.g. daily), data is copied from the operational database into the data warehouse.
- Analysis is done on the data warehouse "offline" then. Since this database is offline and often entails processing huge amounts of data, there are two things to consider:
  - Normalisation comes at a cost for large queries
  - Since it's offline, there are limited insert/update/deletions, so the risk of anomalies is low
- This tends to motivate us to use de-normalised structures for data warehouses

Denormalisation

- Operational Database



- Data Warehouse Database



Denormalisation

- Denormalisation is essentially the reverse process of normalisation: Reducing the number of tables and increasing the amount of redundancy.

- For data warehouses, a common method of denormalisation is to produce star-schemas. Star schemas contain a fact table and dimension tables that essentially mean you never have to do table joins that are more than 1 separated

Assessment 1

- This assignment is due Monday 5pm in week 2.
- The assignment is split up into two parts:
  - 1. focuses on building a normalised database from a very raw table of data. Bulding occurs in the form of drawing a diagram.
  - 2. focuses on converting an operational database into a star schema

End of Week 1