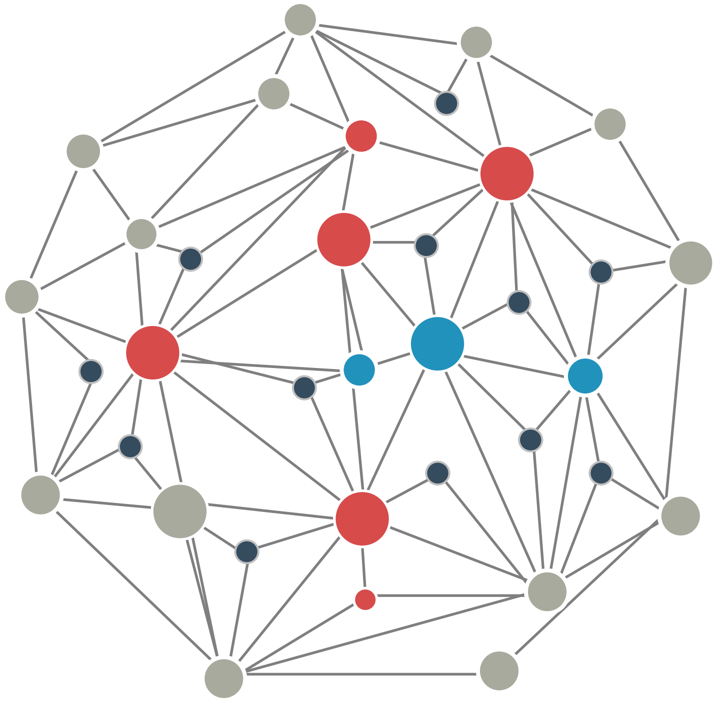
2021-2022 ZJE Database and Software Technology 2

Database

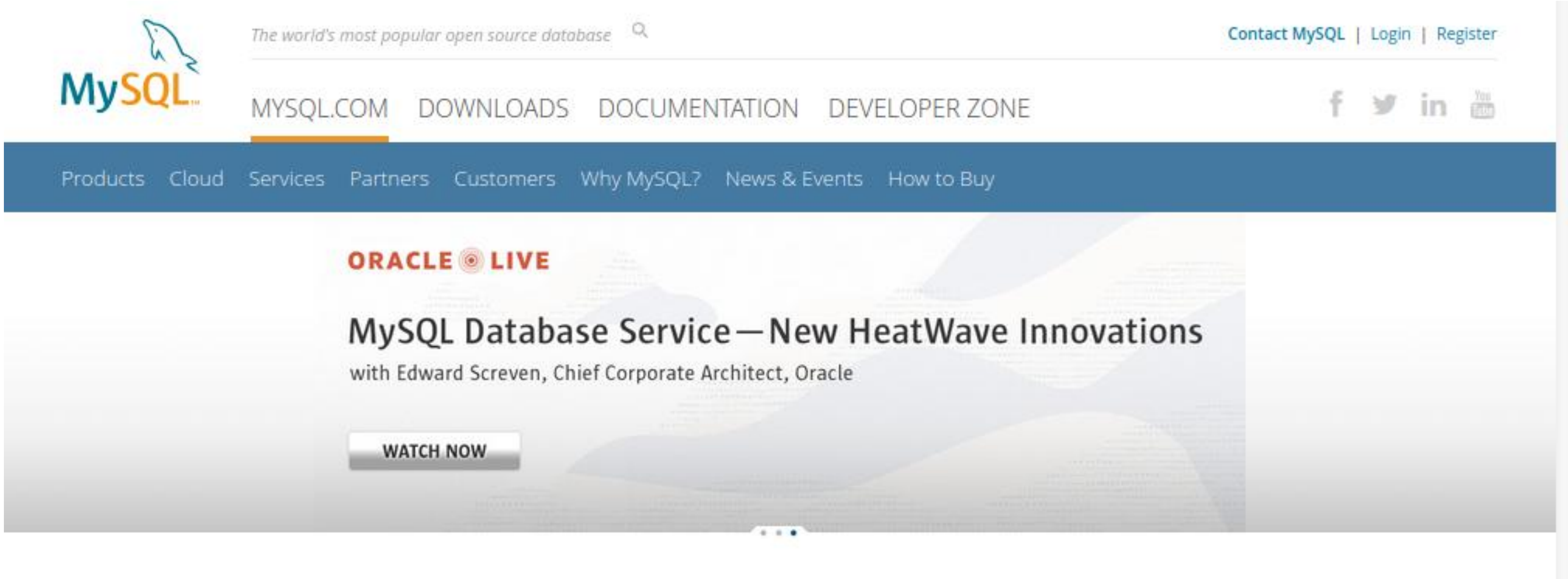
Zhaoyuan Fang

zhaoyuanfang@intl.zju.edu.cn



Course set-up

- **Let’s install MySQL!**
- **Go to : https://www.mysql.com**



MySQL Database Service—New HeatWave Innovations

with Edward Screven, Chief Corporate Architect, Oracle

Course set-up

Tips:

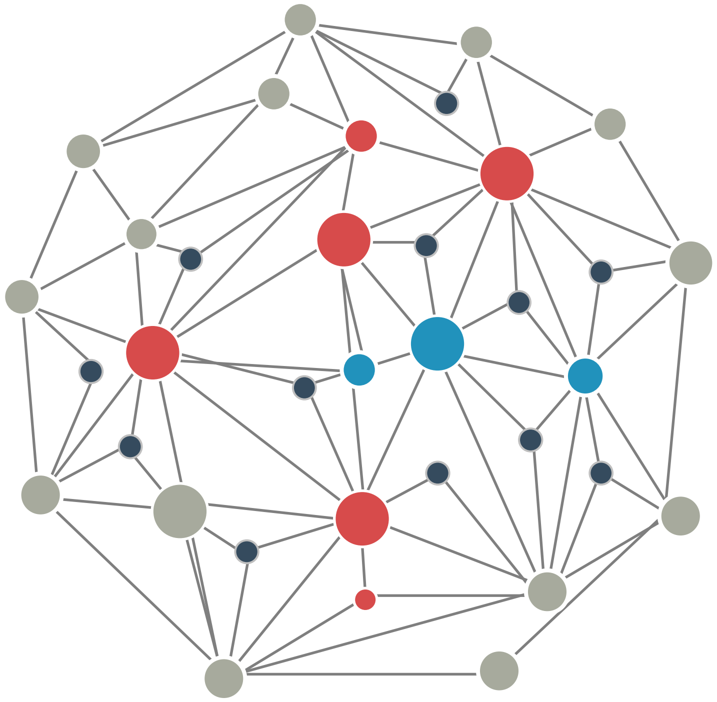
- **Follow the installation GUI**
- **Set a simple root password (e.g. 111)**
- **Type “show databases;”**

DST2 – Week 1

Introduction to relational database

Zhaoyuan Fang

zhaoyuanfang@intl.zju.edu.cn



General Info

database

Prof. Weng

Prof. Fang

Prof. Chen

Course Info – Database part

- Goal:
  - Apply the concepts of relational data model in database design. Use Structured Query Language (SQL) to operate a relational database.
  - Apply the theory of database structure optimization. Balance non-redundancy and query efficiency by functional dependency analysis and design normalization.
  - Able to manage database transaction, data storage, indexing, and query optimization.

Course Info – Database part

- Lecture – Database part
  - 6 weeks, every Tuesday 9:00 am – 11:50 am, total break at 15~20 mins
  - We will combine Lecture/tutorial/practical all in this 2hr30mins

- ICA
  - Mini-project 2: We will use Blackboard system, individual work is required.
    - Deadline: 10:00pm, Feb 13th, 2023 (Mon)
    - What to submit: your SQL code as well as a short report
    - Provisional marks and feedback date: Mar 6th, 2023

Course Info – Database part

- Week 01: Introduction to relational databases
- Week 02: Fundamentals of Structured Query Language (SQL)
- Week 03: Advanced Structured Query Language (SQL)
- Week 04: Database design and E-R modelling
- Week 05: Database normalization, indexing and query optimization
- Week 06: Transaction management and introduction to MySQL

Week 1 Learning Objectives – Part 1+2+3

- Understand the rationale for using database
- Differentiate data vs information vs knowledge
- See how modern databases evolved from file systems
- Describe the main functions of a database management system (DBMS)
- Structure of Relational Databases
- Fundamental Relational-Algebra-Operations
- Additional Relational-Algebra-Operations
- Extended Relational-Algebra-Operations
- Null Values
- Modification of the Database
- Install course set-up environment (MySQL)

Week 1 Learning Objectives – Part 1

- Understand the rationale for using database
- Differentiate data vs information vs knowledge
- See how modern databases evolved from file systems
- Describe the main functions of a database management system (DBMS)

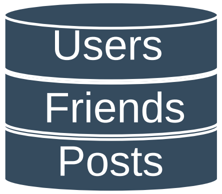
Databases are everywhere: applications in daily life

A day in Susan’s life

See how many databases she interacts with everyday

In the morning, Susan checks her wechat, weibo accounts

- Where is the data about the friends and groups stored?
- Where are the “likes” stored and what would they be used for?



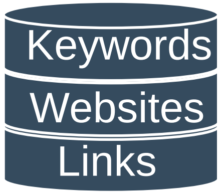
Users

Friends

Posts

At study, susan uses Baidu to search for useful information

- Where is those information stored?
- How the information is queried upon a searching?



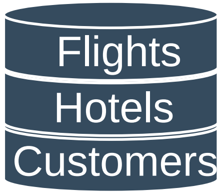
Keywords

Websites

Links

After study, she plans for a trip and buys airline tickets and hotel reservations online

- Where are the airline and hotel data from ?
- What customer data would be kept by the website?
- Where would the customer data be stored?



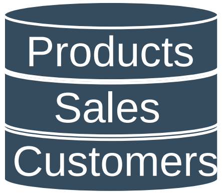
Flights

Hotels

Customers

Then she makes a few online purchases

- Where are the product and transaction data stored?
- Where does the system get the data to generate product “recommendations” to the customer?
- Where would credit card information be stored?



Products

Sales

Customers

Databases are everywhere: applications in biomedical sciences

Repositories

NCBI (GenBank)	http://www.ncbi.nlm.nih.gov/
EMBL-EBI	http://www.ebi.ac.uk/
DDBJ	http://www.ddbj.nig.ac.jp/
INSDC	http://www.insdc.org/
RefSeq	http://www.ncbi.nlm.nih.gov/RefSeq/
VEGA	http://vega.sanger.ac.uk/
CCDS	http://www.ncbi.nlm.nih.gov/CCDS/

Genome Browsers

Ensembl	http://www.ensembl.org/
UCSC Genome Browser	http://genome.ucsc.edu/
NCBI MapViewer	http://www.ncbi.nlm.nih.gov/mapview/
1000 Genomes	http://www.1000genomes.org/

Subject-specific Database

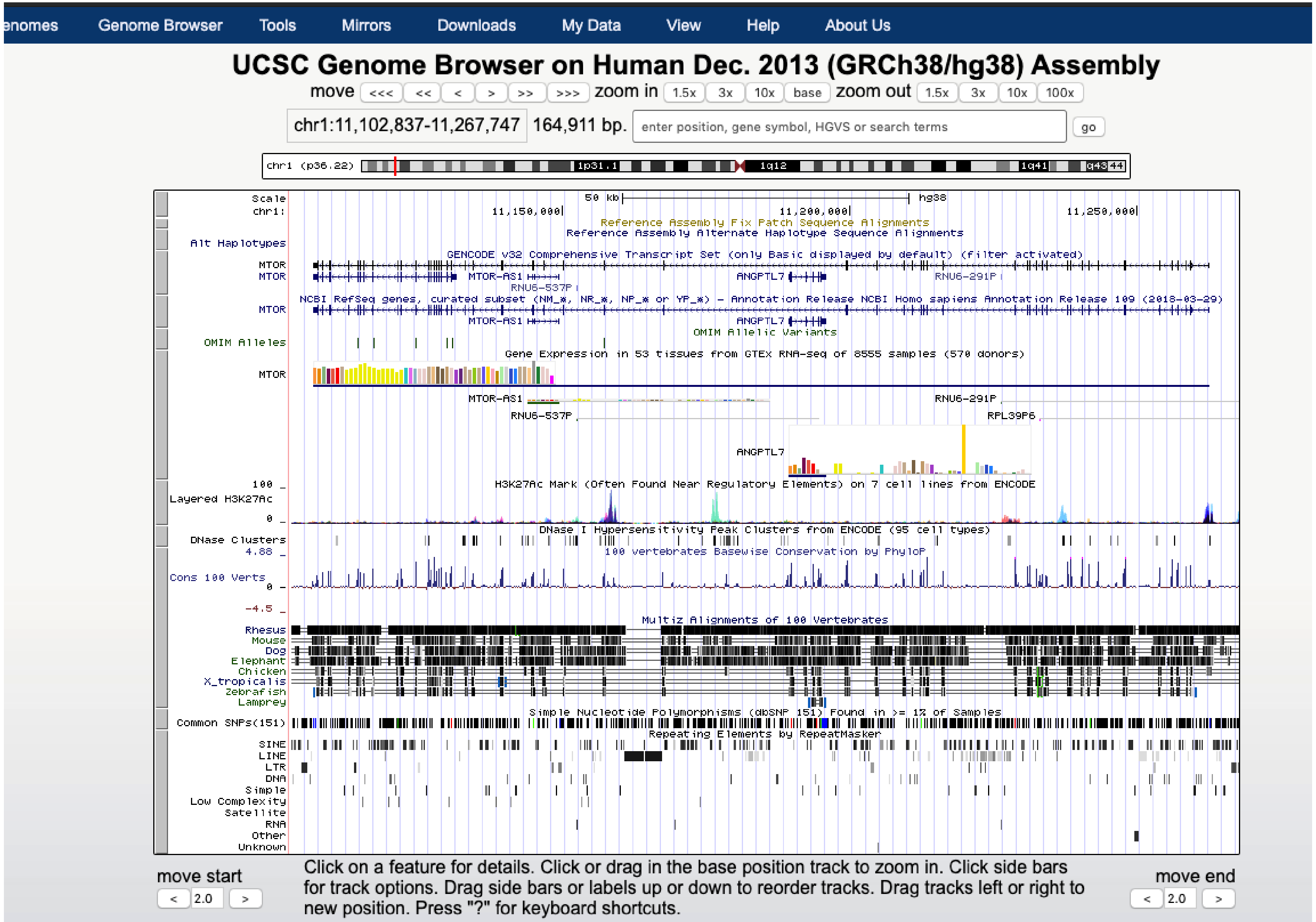
PDB, Protein 3D structures	http://www.rcsb.org/
Pfam, protein families	http://pfam.sanger.ac.uk/
GEO, Gene Expression	http://www.ncbi.nlm.nih.gov/geo/
TCGA, cancer genome atlas	https://portal.gdc.cancer.gov/
ArrayExpress, transcriptomics	http://www.ebi.ac.uk/microarray-as/ae/
dbGaP	http://www.ncbi.nlm.nih.gov/sites/entrez?db=gap
HuGE Navigator	http://www.hugenavigator.net/

Species & Taxa specific databases

Rat	http://rgd.mcw.edu/
Mouse	http://www.informatics.jax.org/
ZFIN, Zebrafish	http://zfin.org FlyBase,
Drosophila	http://flybase.org/ VectorBase,
human disease	http://www.vectorbase.org/
C. elegans	http://www.wormbase.org
crop grasses	http://www.gramene.org
Arabidopsis	http://www.arabidopsis.org/
Yeast	http://www.yeastgenome.org/
Microbial	http://img.jgi.doe.gov/ EcoliHub,
E. coli	http://www.ecolicommunity.org/ VBRC,
Viral	http://athena.bioc.uvic.ca/

And many more!

Example - UCSC Genome Browser (a huge database)


enomes
Genome Browser
Tools
Mirrors
Downloads
My Data
View
Help
About Us
move <<< << < > >> >>>
zoom in 1.5x 3x 10x base
zoom out 1.5x 3x 10x 100x
go
chr1 (p36.22)
1p31.1
1q12
1q41
q43
44
Scale
chr1:
50 kb
hg38
11,150,000
11,200,000
11,250,000
Reference Assembly Fix Patch Sequence Alignments
Reference Assembly Alternate Haplotype Sequence Alignments
Alt Haplotypes
GENCODE v32 Comprehensive Transcript Set (only Basic displayed by default) (filter activated)
MTOR
MTOR
MTOR-AS1
ANGPTL7
RNU6-291P
RNU6-537P
NCBI RefSeq genes, curated subset (NM_*, NR_*, NP_* or YP_*) - Annotation Release NCBI Homo sapiens Annotation Release 109 (2018-03-29)
MTOR
ANGPTL7
MTOR-AS1
OMIM Allelic Variants
OMIM Alleles
Gene Expression in 53 tissues from GTEx RNA-seq of 8555 samples (570 donors)
MTOR
MTOR-AS1
RNU6-291P
RNU6-537P
RPL39P6
ANGPTL7
100 _
H3K27Ac Mark (Often Found Near Regulatory Elements) on 7 cell lines from ENCODE
Layered H3K27Ac
0 _
DNase I Hypersensitivity Peak Clusters from ENCODE (95 cell types)
DNase Clusters
4.88 _
100 vertebrates Basewise Conservation by PhyloP
Cons 100 Verts
0
-4.5 _
Multiz Alignments of 100 Vertebrates
Rhesus
Mouse
Dog
Elephant
Chicken
X_tropicalis
Zebrafish
Lamprey
Simple Nucleotide Polymorphisms (dbSNP 151) Found in >= 1% of Samples
Common SNPs(151)
Repeating Elements by RepeatMasker
SINE
LINE
LTR
DNA
Simple
Low Complexity
Satellite
RNA
Other
Unknown
move start
< 2.0 >
move end
< 2.0 >

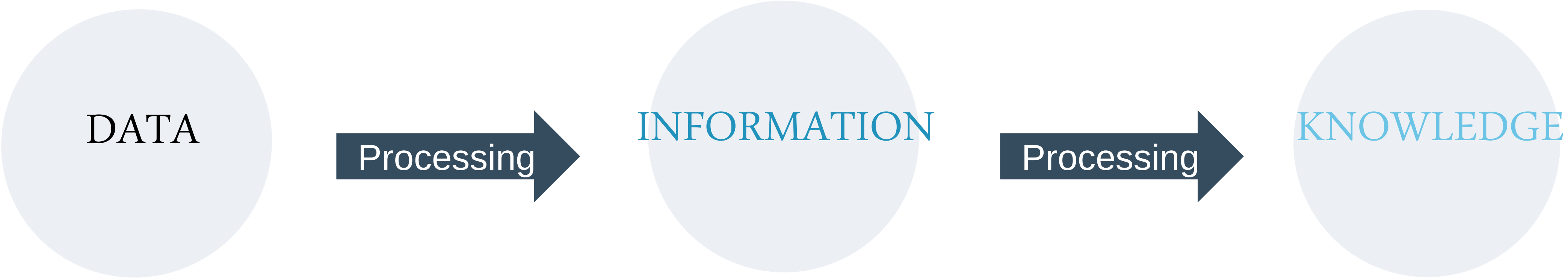
UCSC Genome Browser on Human Dec. 2013 (GRCh38/hg38) Assembly

chr1:11,102,837-11,267,747 164,911 bp. enter position, gene symbol, HGVS or search terms

Click on a feature for details. Click or drag in the base position track to zoom in. Click side bars for track options. Drag side bars or labels up or down to reorder tracks. Drag tracks left or right to new position. Press "?" for keyboard shortcuts.

Live demo:
http://genome.ucsc.edu/

What is Data? Data vs Information vs Knowledge



DATA

Processing

INFORMATION

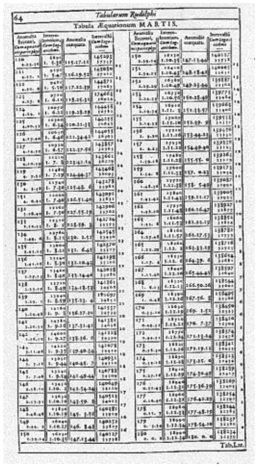
Processing

KNOWLEDGE

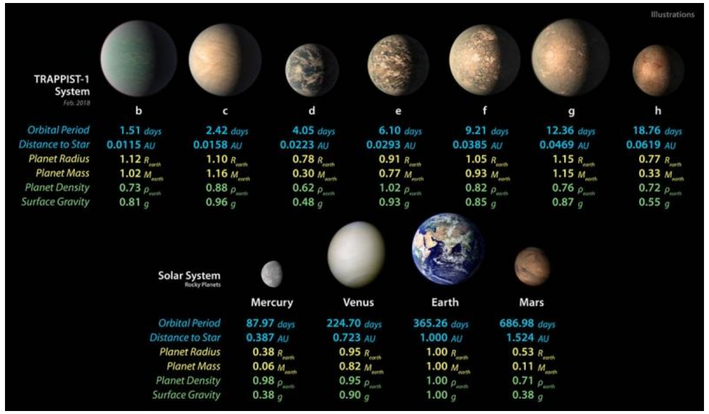
Raw facts
(number/text/figure/sequence…)

Reveal its meaning

information and facts
about a specific subject



Astronomical data of planets



Illustrations

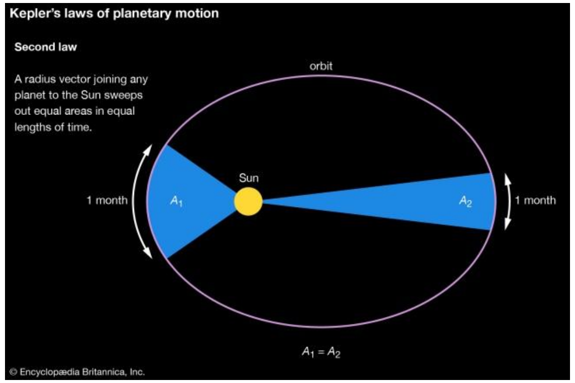
TRAPPIST-1 System
Feb. 2018

	b	c	d	e	f	g	h
Orbital Period	1.51 days	2.42 days	4.05 days	6.10 days	9.21 days	12.36 days	18.76 days
Distance to Star	0.0115 AU	0.0158 AU	0.0223 AU	0.0293 AU	0.0385 AU	0.0469 AU	0.0619 AU
Planet Radius	1.12 R_{earth}	1.10 R_{earth}	0.78 R_{earth}	0.91 R_{earth}	1.05 R_{earth}	1.15 R_{earth}	0.77 R_{earth}
Planet Mass	1.02 M_{earth}	1.16 M_{earth}	0.30 M_{earth}	0.77 M_{earth}	0.93 M_{earth}	1.15 M_{earth}	0.33 M_{earth}
Planet Density	0.73 ρ_{earth}	0.88 ρ_{earth}	0.62 ρ_{earth}	1.02 ρ_{earth}	0.82 ρ_{earth}	0.76 ρ_{earth}	0.72 ρ_{earth}
Surface Gravity	0.81 g	0.96 g	0.48 g	0.93 g	0.85 g	0.87 g	0.55 g

Solar System
Rocky Planets

	Mercury	Venus	Earth	Mars
Orbital Period	87.97 days	224.70 days	365.26 days	686.98 days
Distance to Star	0.387 AU	0.723 AU	1.000 AU	1.524 AU
Planet Radius	0.38 R_{earth}	0.95 R_{earth}	1.00 R_{earth}	0.53 R_{earth}
Planet Mass	0.06 M_{earth}	0.82 M_{earth}	1.00 M_{earth}	0.11 M_{earth}
Planet Density	0.98 ρ_{earth}	0.95 ρ_{earth}	1.00 ρ_{earth}	0.71 ρ_{earth}
Surface Gravity	0.38 g	0.90 g	1.00 g	0.38 g

Motion information for planets



Kepler's laws of planetary motion

Second law

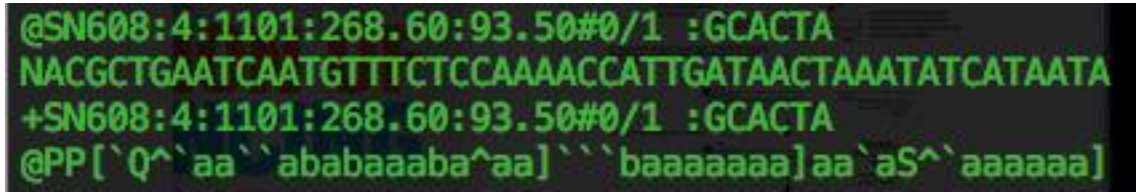
A radius vector joining any planet to the Sun sweeps out equal areas in equal lengths of time.

© Encyclopædia Britannica, Inc.

Kepler's laws

What is Data? Data vs Information vs Knowledge

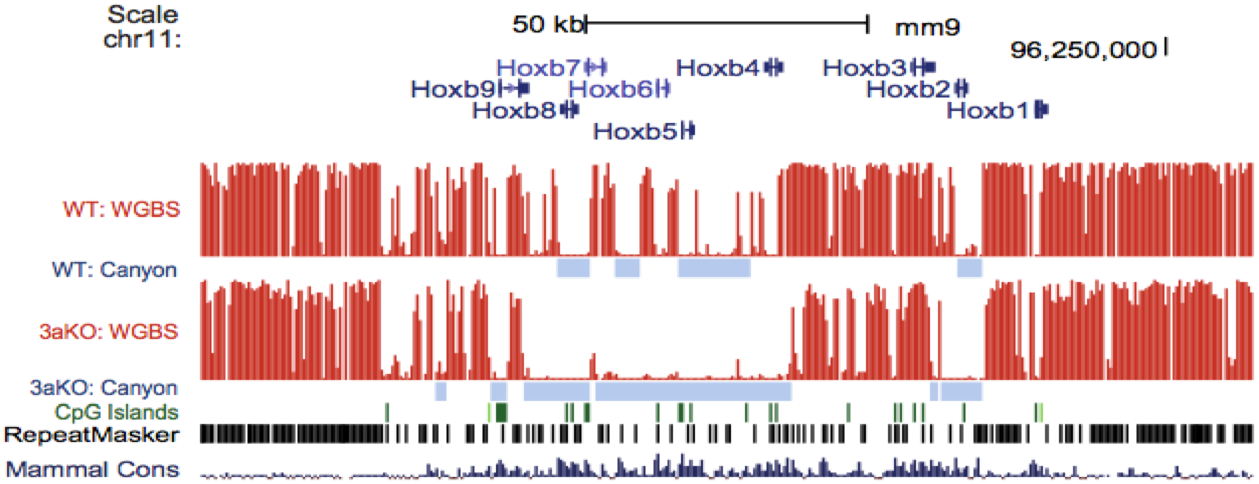
DATA



```
@SN608:4:1101:268.60:93.50#0/1 :GCACTA
NACGCTGAATCAATGTTTCTCCAAAACCATTGATAACTAAATATCATAATA
+SN608:4:1101:268.60:93.50#0/1 :GCACTA
@PP[`Q^`aa``ababaaaba^aa]```baaaaaaa]aa`aS^`aaaaaa]
```

Read name
sequence
Read name
Sequence quality

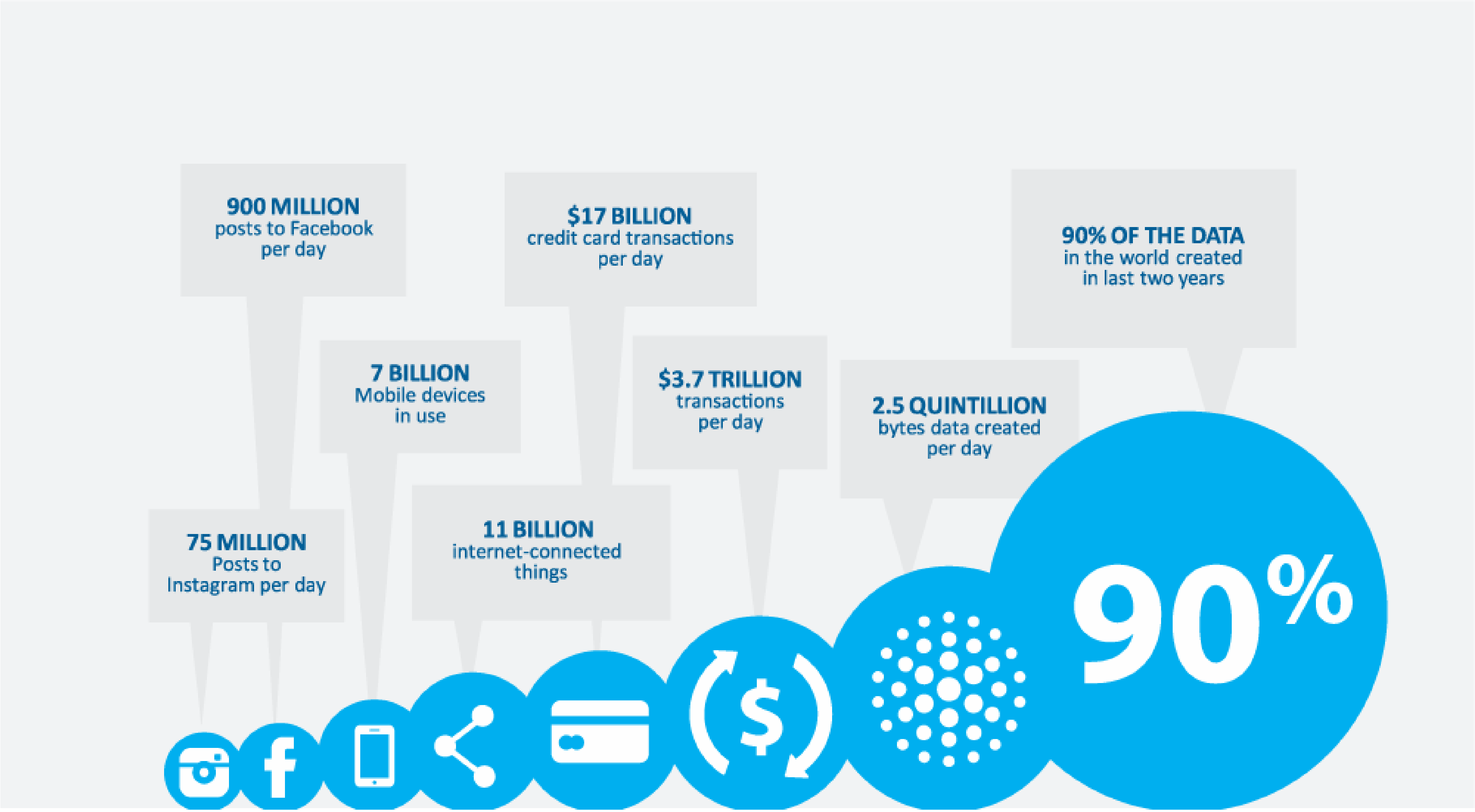
Information



Knowledge DNA methylation is decreased over *some genes* in *some cells*

Mira Jeong, et al., 2013, Nature Genetics

Data Management: Information explosion



900 MILLION posts to Facebook per day

7 BILLION Mobile devices in use

$17 BILLION credit card transactions per day

$3.7 TRILLION transactions per day

90% OF THE DATA in the world created in last two years

2.5 QUINTILLION bytes data created per day

75 MILLION Posts to Instagram per day

11 BILLION internet-connected things

How is Data Managed? primitive vs. advanced

keep records by tying knots

The basis of Data Management

- Let’s say you need to develop a management system for the library, so the facts you need to figure out in the beginning are :
  - Who are the objects? What are the operations?

    Objects: books, students, administrator

    Operations: book borrowing, reservation, book returning etc…

  - So your jobs are:

    How to define objects?
    How to store such data?
    How to implement those operations?
    A visible system that is user-friendly …

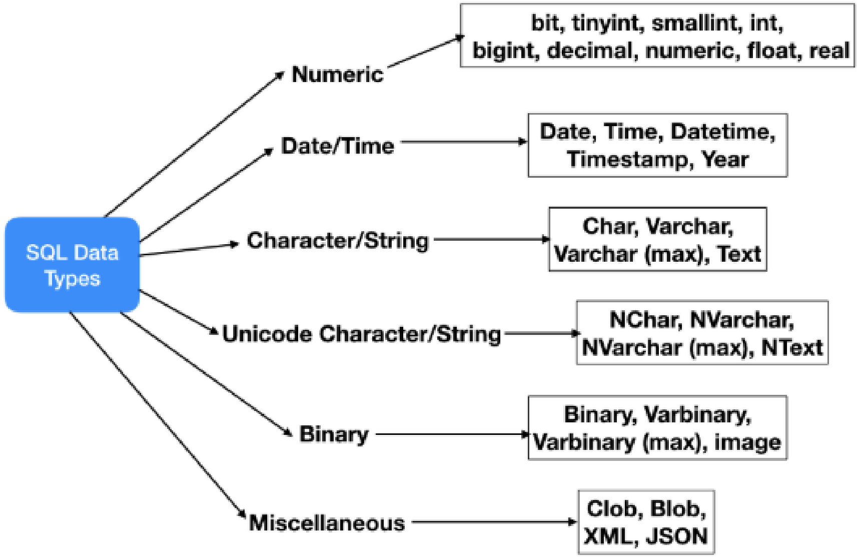
We will answer these questions, through this course

Data Structure/Data Type/Data Management

Data Structure:

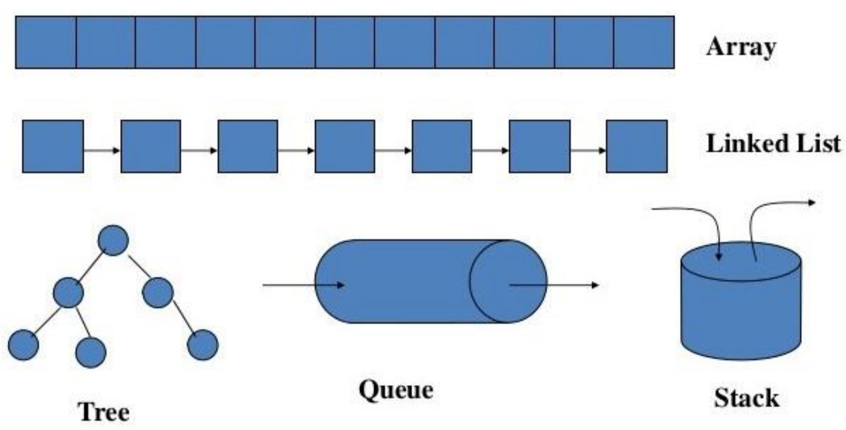
The **logical and mathematical** model of a particular organization of data and the relationship among its members is called a data structure. It is an organized collection of data which perform a **particular operation**.

Data Type:



SQL Data Types

- Numeric → bit, tinyint, smallint, int, bigint, decimal, numeric, float, real
- Date/Time → Date, Time, Datetime, Timestamp, Year
- Character/String → Char, Varchar, Varchar (max), Text
- Unicode Character/String → NChar, NVarchar, NVarchar (max), NText
- Binary → Binary, Varbinary, Varbinary (max), image
- Miscellaneous → Clob, Blob, XML, JSON



Data Management:

- Data definition: logical + physical
- Data operation:
  - Query: type/characteristics/relationship
  - Update: insert, delete, edit
- Data constraints: consistency of data type, restriction on the data value etc.

How is Data Managed? History

Manual Management (before 1950s)

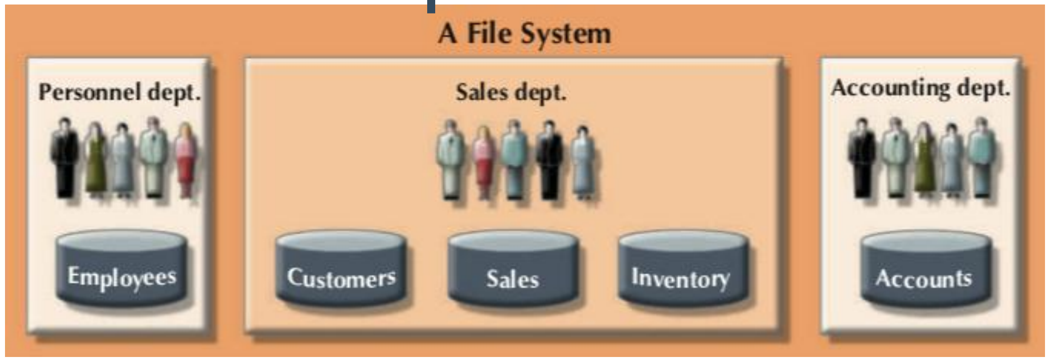
"Farming Age"

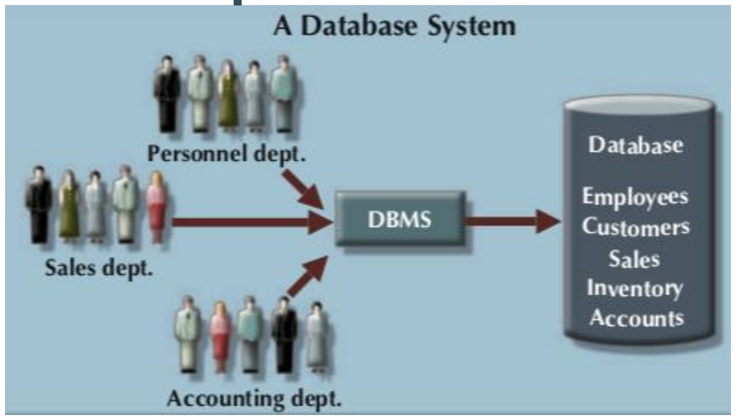
File system (1950-1960)

"Industrial Age"

Database system (After 1960-)

"Information Age"





Manual Data management

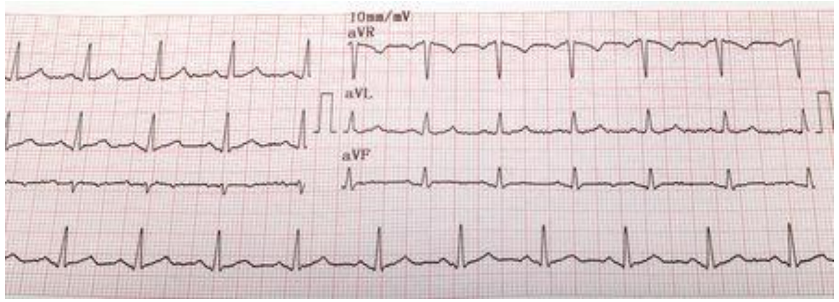
Computers are not used or not widely used

- Small amount of data and simple structure, such as high-order equations, curve fitting, etc.

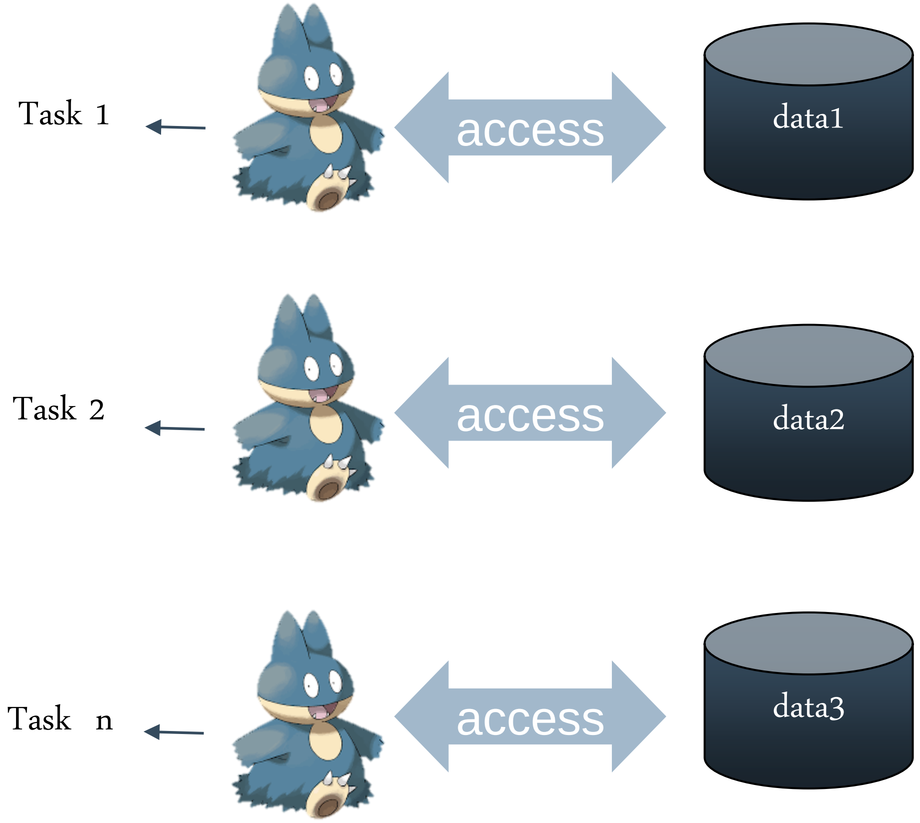
Data stored in sequential access "devices"

- Books, papers, tapes, cards; no disk

No operating system, no data management software



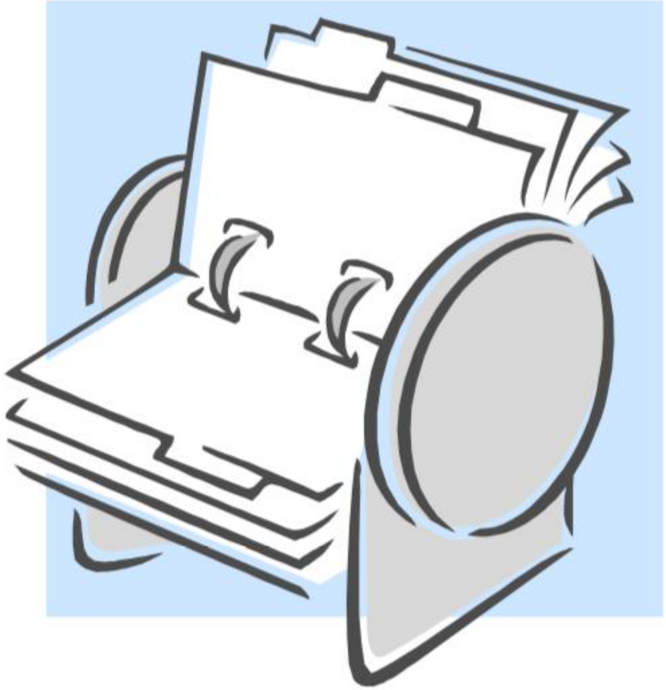
No computer – no 'light'



Manual Data management

Without databases

- Each application manages its own data
- Data is stored multiple times (**redundancy**)
- Problems:
  - Waste of storage space
  - “Forgetting” of changes
  - No centralized, “standardized” data management



Problems of Data Redundancy

Data Redundancy may cause:

- Other software systems cannot process large amounts of data efficiently
- Many users or applications cannot access the same data in parallel without interfering with each other
- Application developers / users cannot develop / use applications without knowing
  - internal representation of data
  - storage media or computers

  (no data independence)
- No data security; potential loss of data

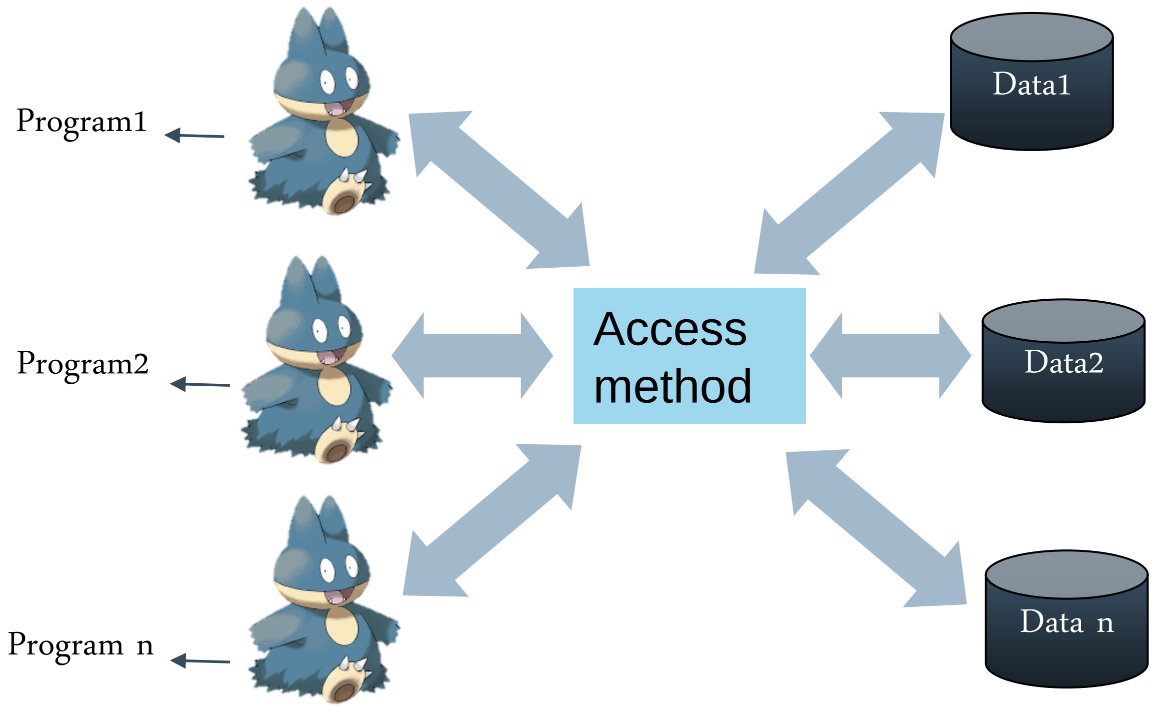
File System

- After 1960s, Computer is used for computing and management
- Data is stored in disk operating system

Characteristics:
Storage space management, Directory management, File read and write management, file protection, user operation interface

Problem with file system:

- Long development time
- Difficulty of getting quick answers
- Complex system administration
- Lack of security and limited data sharing
- Extensive programming
- Data redundancy



Islands of information

Database motivation

- Database system are center piece of modern IT systems

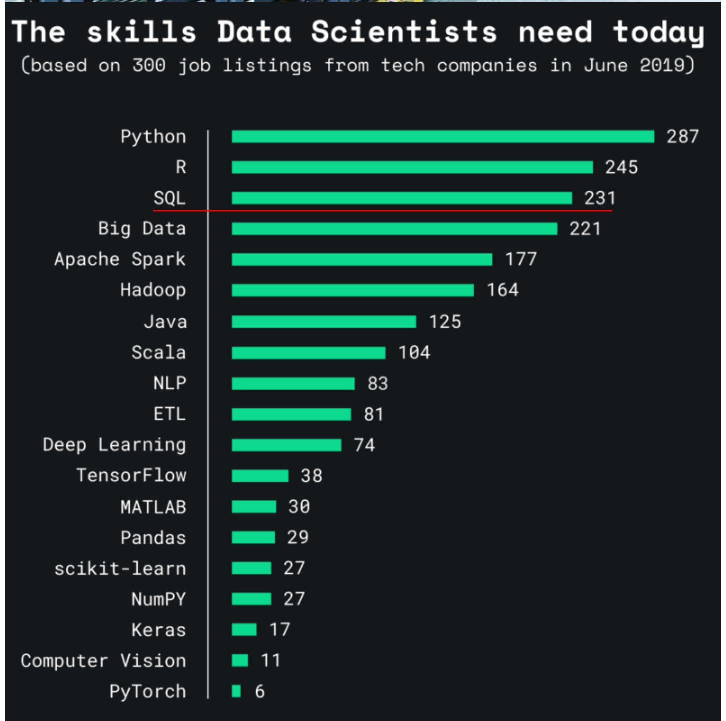
- … ubiquitous

- Database scientists are in high demands in all area

Harvard Business Review РОССИЯ

ANALYTICS

Data Scientist: The Sexiest Job of the 21st Century



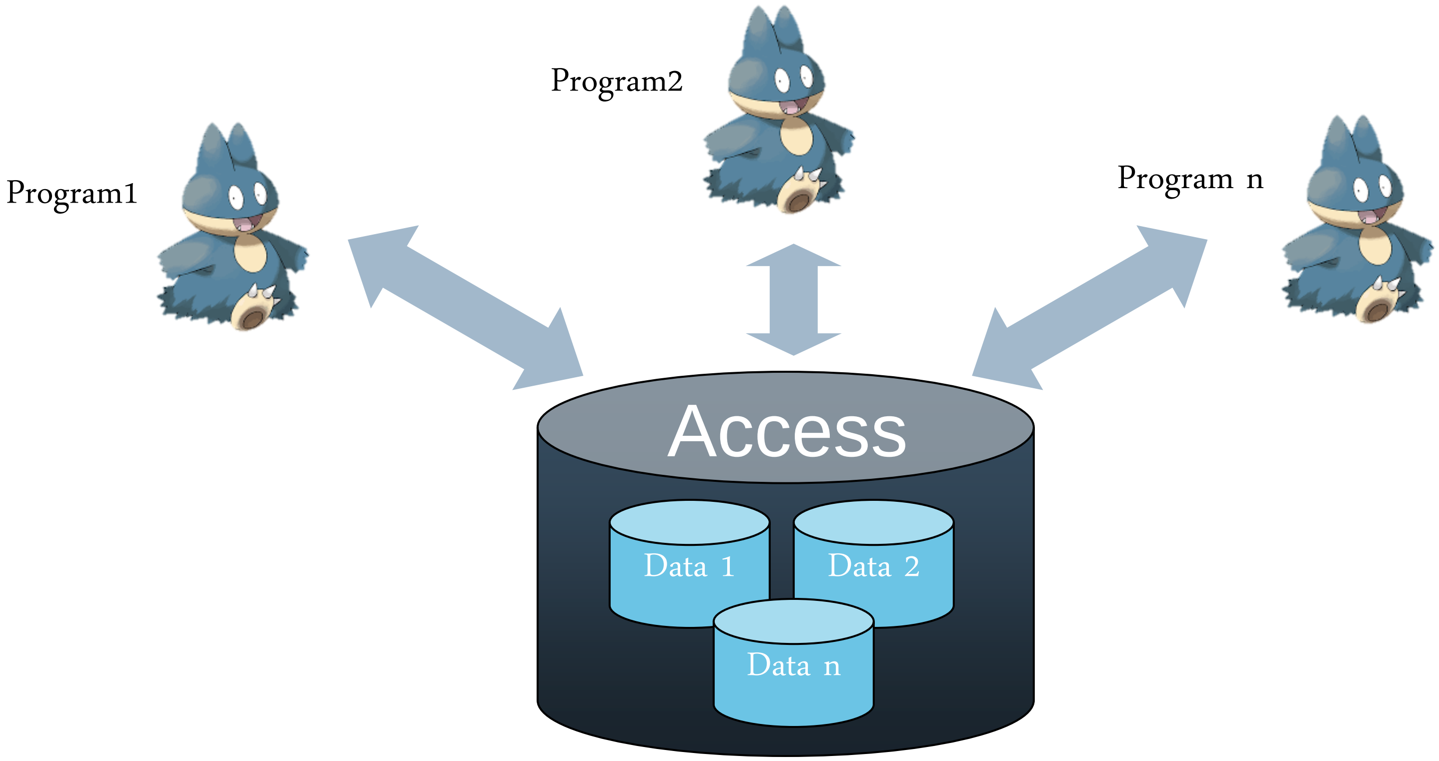
The skills Data Scientists need today

(based on 300 job listings from tech companies in June 2019)

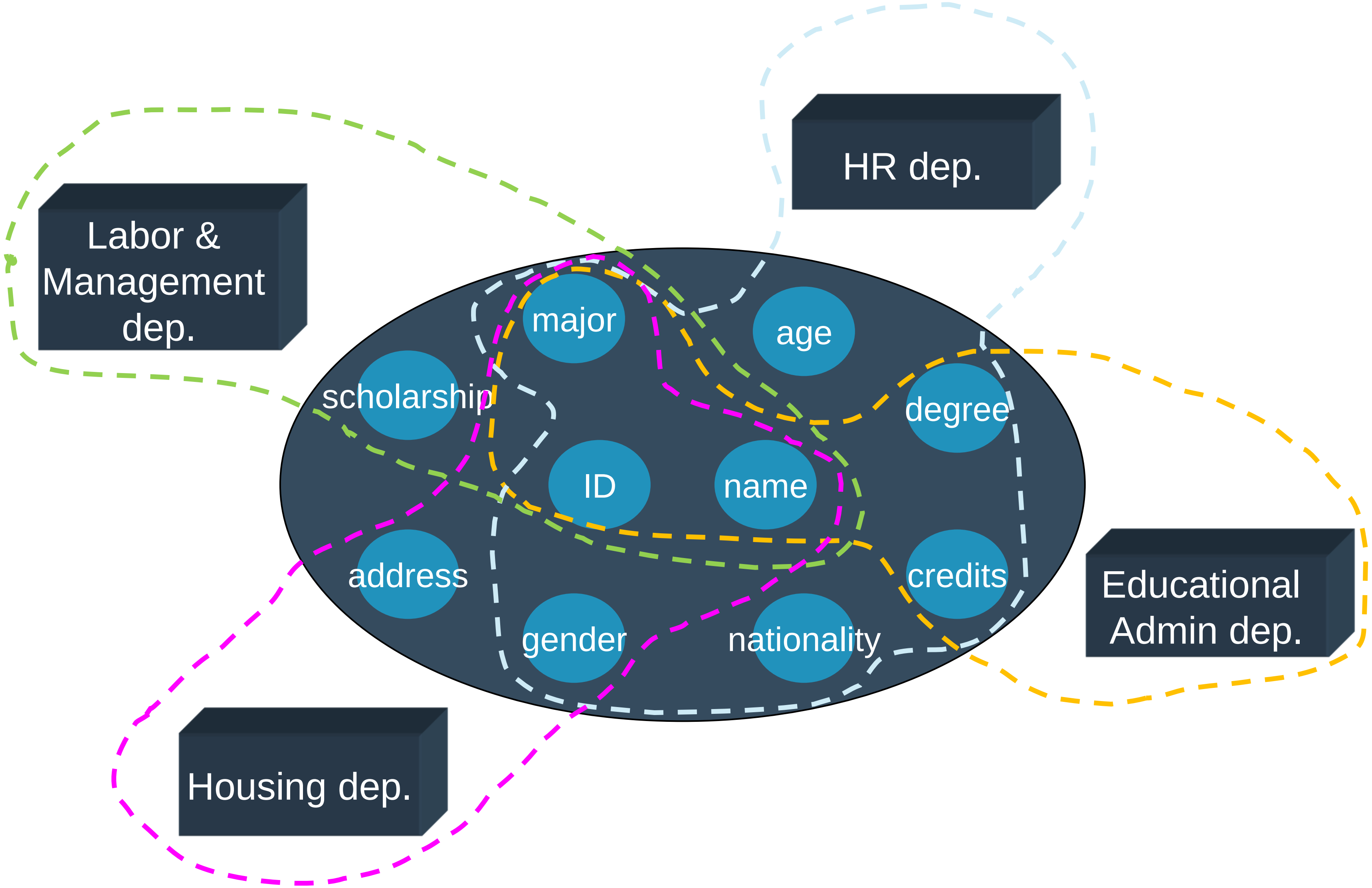
Database System background and characteristics

- Computers become super powerful in managing huge data with complicated relationship
- Data need to be shareable (between different Apps, languages etc)
- The development of storage techniques
- Softwares become more expensive, but hardwares become cheaper

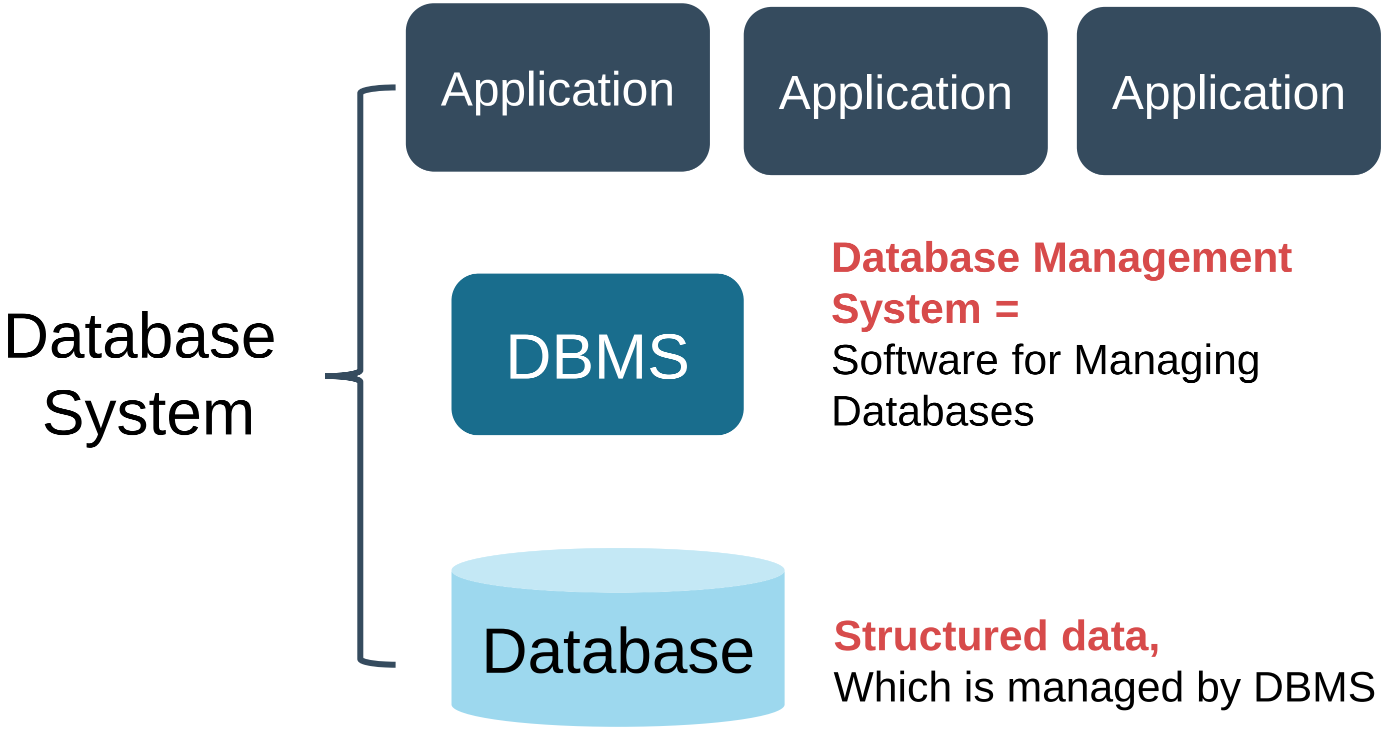
→ Database management systems (DBMS)



School Database Management System



DBMS is the core of a database system



Database Management System =
Software for Managing Databases

Structured data,
Which is managed by DBMS

Advantages of DBMS:

- Accessible for everyone/Apps/programming language in the group
- Data structure indicate the relationship between data
- Low data redundancy
- High independence from Apps
- Easy to expand
- Unified data control
  - Security
  - Integrity
  - Concurrency
  - Easier to recover

How to manage data in the future?

Questions we need to consider:

- How to organize (model and use) data?
- How to store data safely and persistently?
- How to process huge amounts of data (>= terabytes) efficiently?
- How can many users (10,000 or even more) access data concurrently?

Summary for Part 1

- Recognize the importance of database
- Understand the relationship of data vs information vs knowledge
- Evolution of modern database (manual – file system – database)
- illustrate the functions of a database management system (DBMS)

Break time

Week 1 Learning Objectives – Part 2

- **Structure of Relational Databases**
- **Fundamental Relational-Algebra-Operations**
- **Additional Relational-Algebra-Operations**
- **Extended Relational-Algebra-Operations**
- **Null Values**
- **Modification of the Database**

Group exercise/task rules

- Throughout this course, we will have some group exercises/tasks for you to complete
- Students sitting on same table naturally forming a group. You can discuss and solve the problems.
- Then show your answer as volunteer (otherwise I will pick one group randomly)

Before we start, give a fancy name for your team (1 mins).

▯ **Hello SQLer!**

Relational database

- Some early models (immature)
- In 1970, Codd proposed a relational model of database. He won 1981 ACM Turing Award for this.
- A software system used to maintain relational databases is a relational database management system (RDBMS).

- System (early days):
  - System R : developed by IBM
  - INGRES: developed by UCBerkeley
- Current commercial RDBMS:
  - Oracle, SQL Server, DB2, MySQL, PostgreSQL
  - Access (Microsoft)

What is a Relation? example

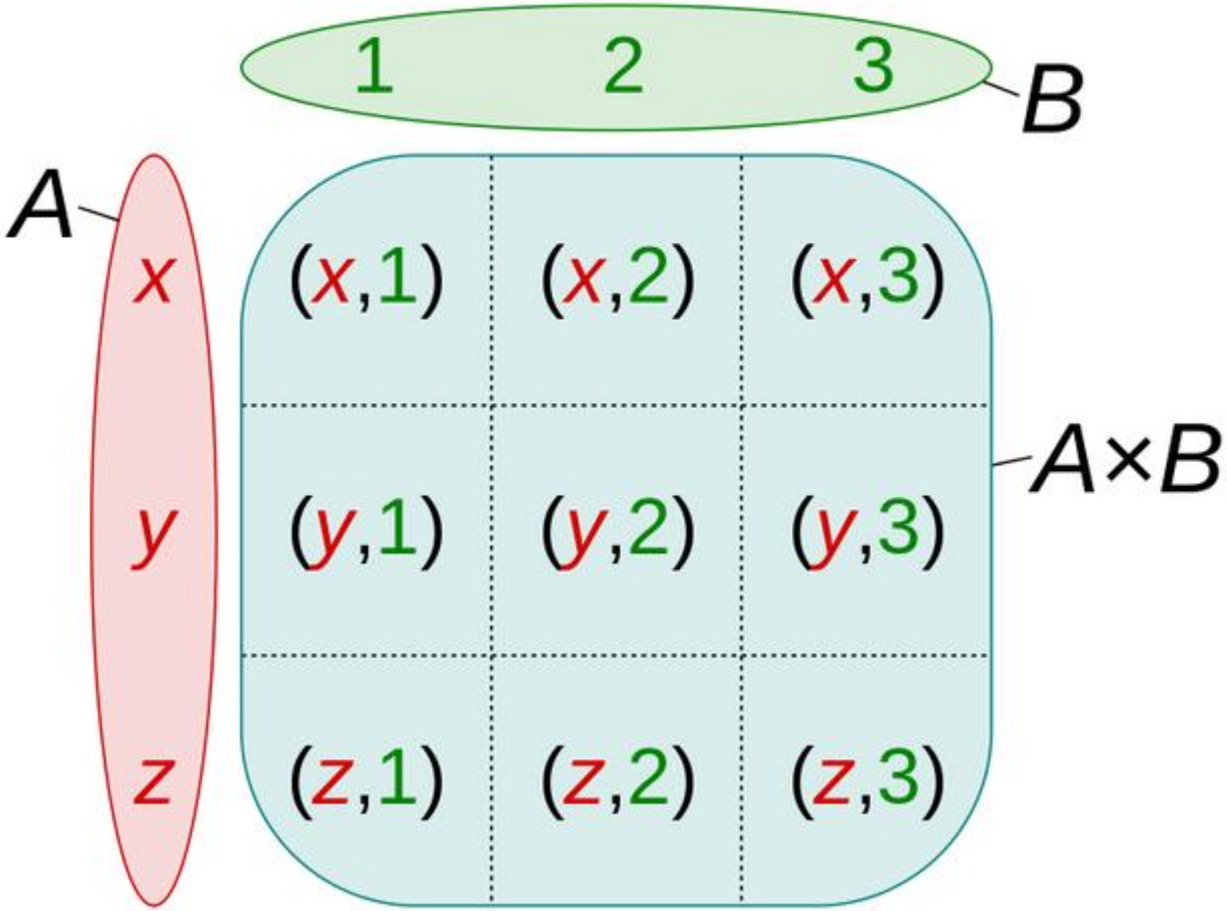
Relation: **attribute 1** **attribute 2**

Name	Gender
Smith	Male
Lisa	Female
Mohamed	Male
Ana	Female
Noah	Male

(Lisa, Female) **tuple**

Domain D1 **{Smith, Lisa, …, Qian, …}** **X** **Domain D2** **{Male, Female}** **=** **All possible tuples** **{ (Smith,Male), …}**

Cartesian product



(x,1)	(x,2)	(x,3)
(y,1)	(y,2)	(y,3)
(z,1)	(z,2)	(z,3)

Basic structure

- Formally, given sets D_1, D_2, …. D_n, a **relation** *r* is a subset of Cartesian product D_1 x D_2 x … x D_n

$$r \in D_1 \text{ x } D_2 \text{ x } \ldots \text{ x } D_n$$

- Thus, a **relation** is a **set of tuples** (a_1, a_2, …, a_n) where each $a_i \in D_i$

Name	Gender
Smith	Male
Lisa	Female
Mohamed	Male
Ana	Female
Noah	Male
Domain D1	**Domain D2**

(Lisa, Female) **tuple**

Basic structure

- Example: If
  - *customer_name* = {Jones, Smith, Curry, Lindsay, …} /* Set of all customer names */
  - *customer_street* = {Main, North, Park, …} /* set of all street names*/
  - *customer_city* = {Harrison, Rye, Pittsfield, …} /* set of all city names */

Then *r* = { (Jones, Main, Harrison),
(Smith, North, Rye),
(Curry, North, Rye),
(Lindsay, Park, Pittsfield) }

is a relation over

customer_name x customer_street x customer_city

Attribute and Domain

- Each attribute has its values from a **domain**
- Attribute values are (normally) required to be **atomic**
  - i.e. indivisible (not a set of values)
- The special value *null* is a member of every domain
  - null may cause complications in the definition of many operations
  - We shall ignore the effect of null values in our main presentation and consider their effect later

attribute domain

customer_name <= {Jones, Smith, Curry, Lindsay, …}

Relational schema: an abstract form

- A_1, A_2, …, A_n are *attributes*

- R = (A_1, A_2, …, A_n) is a *relational schema*

  Example:

  Customer_schema = (*customer_name, customer_street, customer_city*)

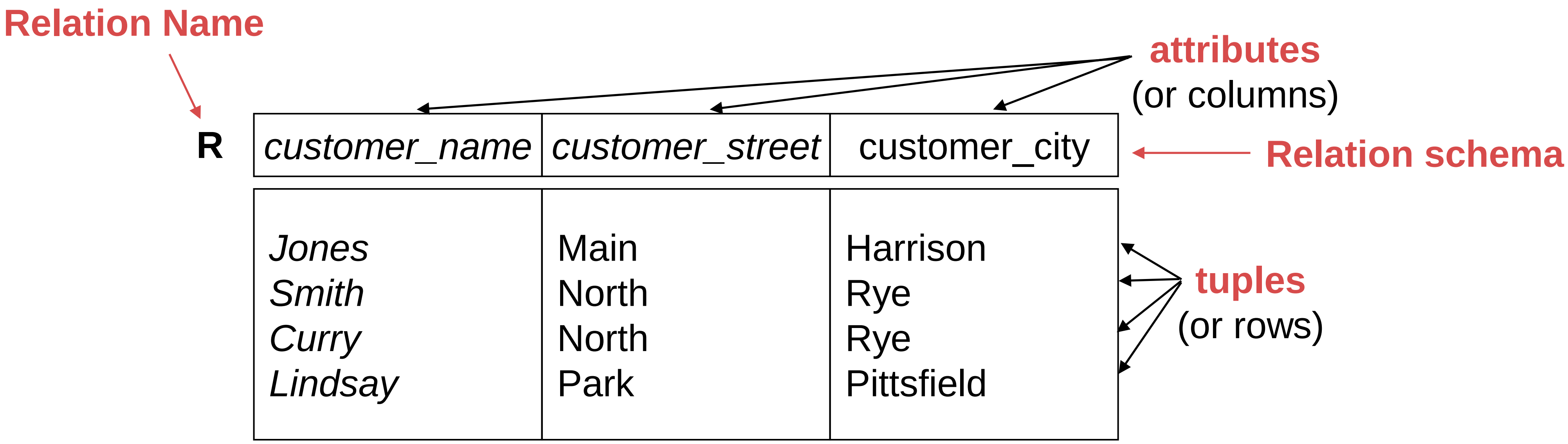
- *r*(*R*) denotes a *relation r* on the *relation schema R*

  Example:

  customer (Customer_schema) *-- a real table*

Relation: an instance of relational schema

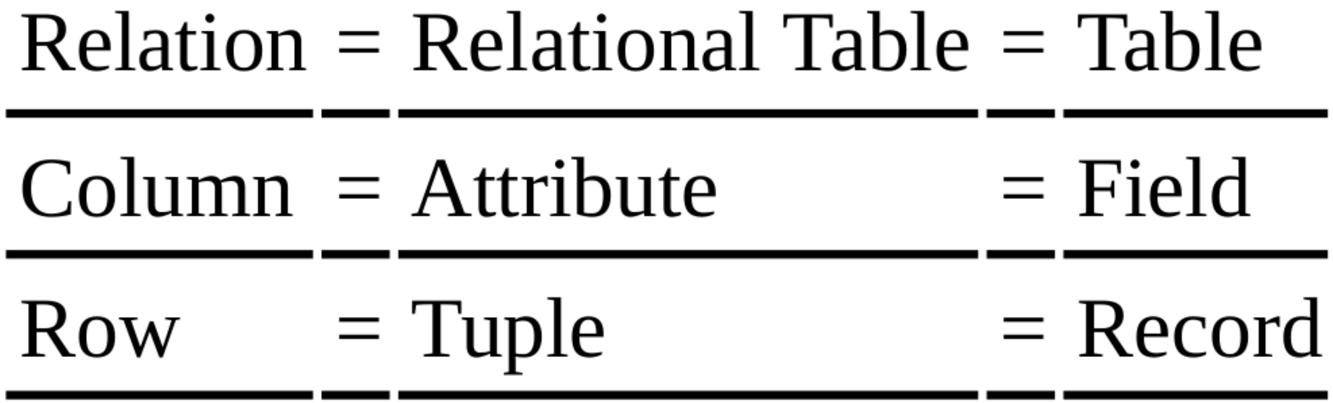
- The current values of a relational schema are specified by a table



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

customer

Some synonyms



Relation	=	Relational Table	=	Table
Column	=	Attribute	=	Field
Row	=	Tuple	=	Record

Characteristics of Relation

- Each attribute is **homogenous**
  - i.e. values are from **same domain**, and each column have the **same data type**
  - attribute values are **atomic** (a single value, but not a set)
  - *Null* can be in any data type
- Tuples are a set
  - so must be **unique**, but can be in any **order**

Attributes/column homogenous

T#	S#	C#
t1	s1	c1
t1	t2	c2

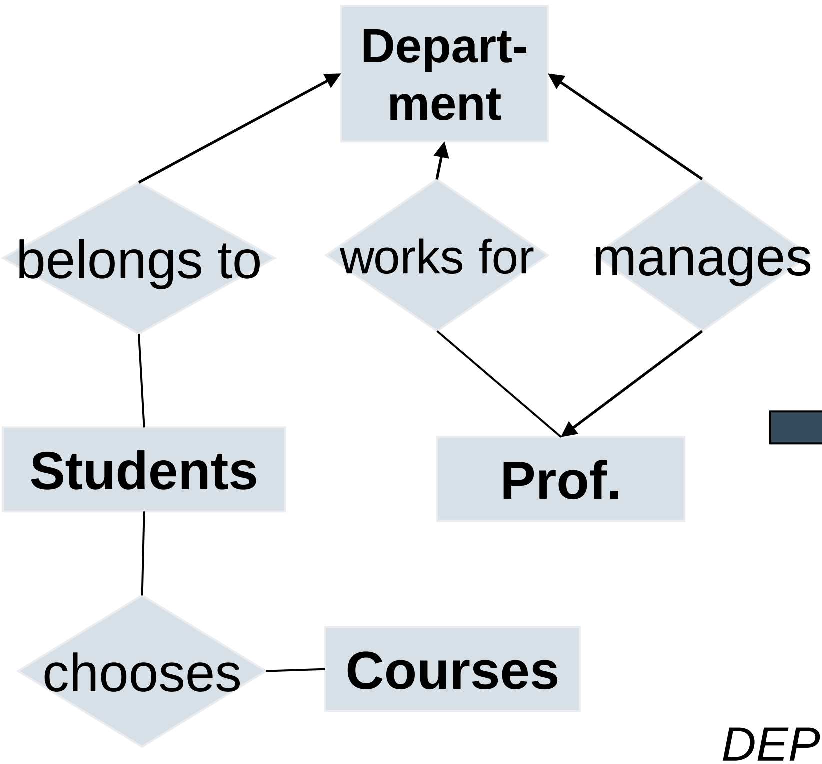
Task 1: Yes or No?

Different attribute from same domain

E#	ENAME	MGR
e1	TOM	e2
e2	jerry	null

Task 2: Yes or No?

Relational Model – from relation to database



DEPT(dno , dname , **dean**)

S(sno , sname, sex, age, **dno**)

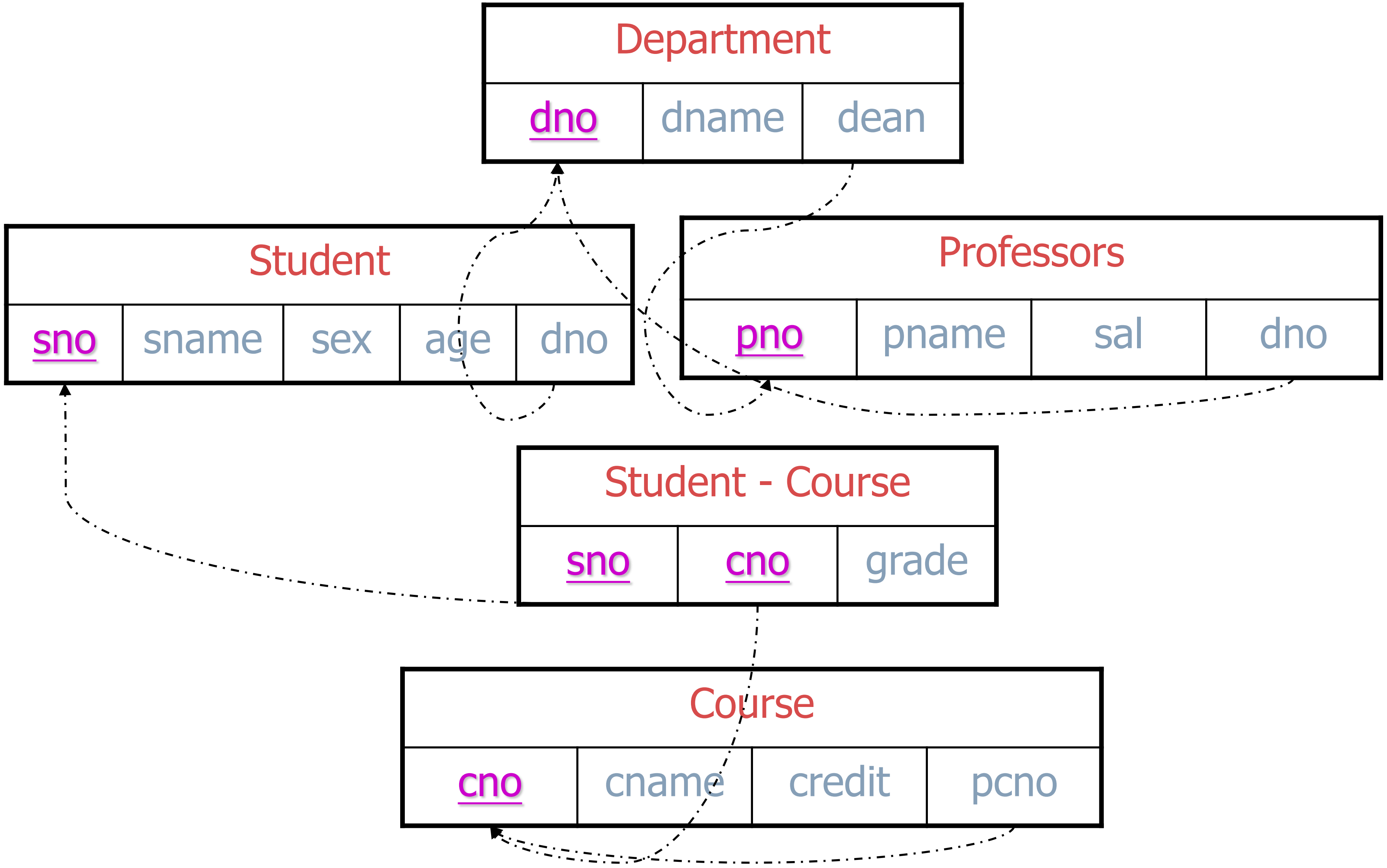
C(cno , cname , **pcno** , credit)

SC(sno, cno , grade)

PROF(pno , pname, sal, **dno**)

DEPT: Department
S: student
C: Course
SC ： Student-Course
PROF ： Professor

Relational Model – from relation to database



Department

dno dname dean

Student

sno sname sex age dno

Professors

pno pname sal dno

Student - Course

sno cno grade

Course

cno cname credit pcno

Integrity Constraints - Keys

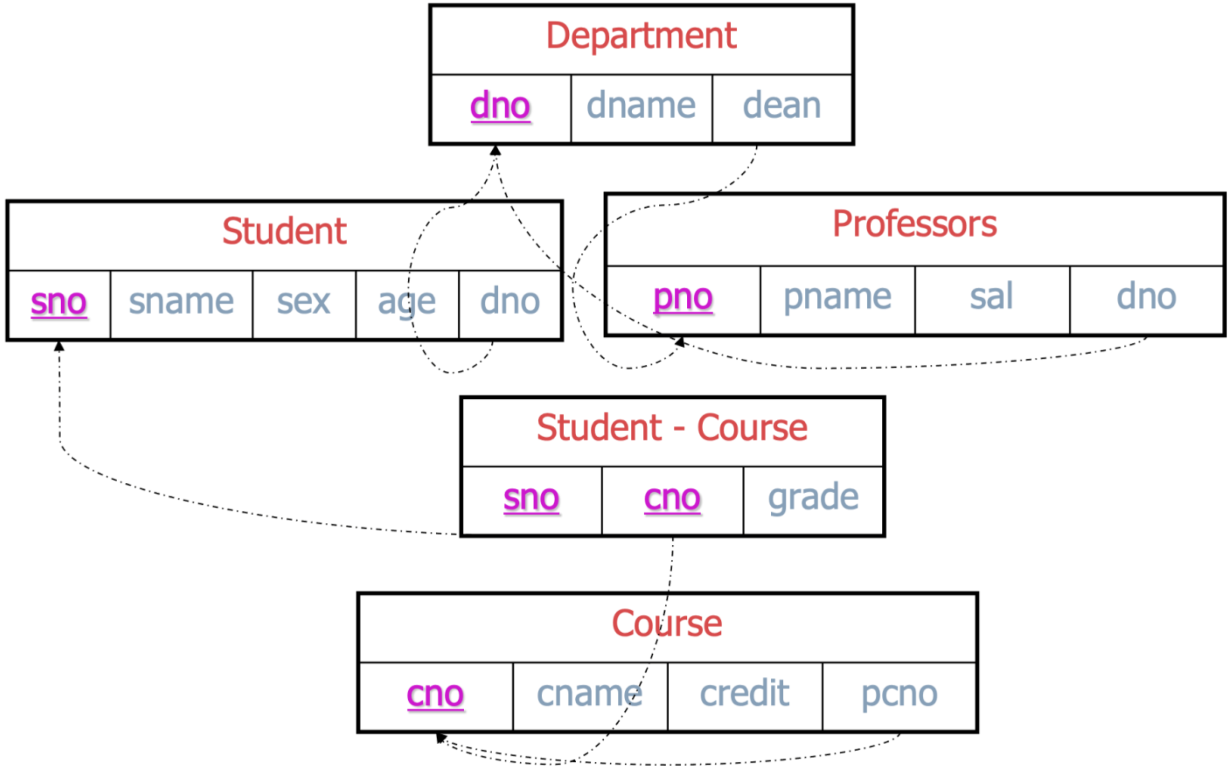
Candidate Key:

The **minimal set of attributes** which can **uniquely** identify a tuple is known as candidate key.

dno, dname in DEPT relations.

Task 3 What could be the candidate keys S relations?

Task 4 What could be the candidate keys C relations?



Integrity Constraints - Keys

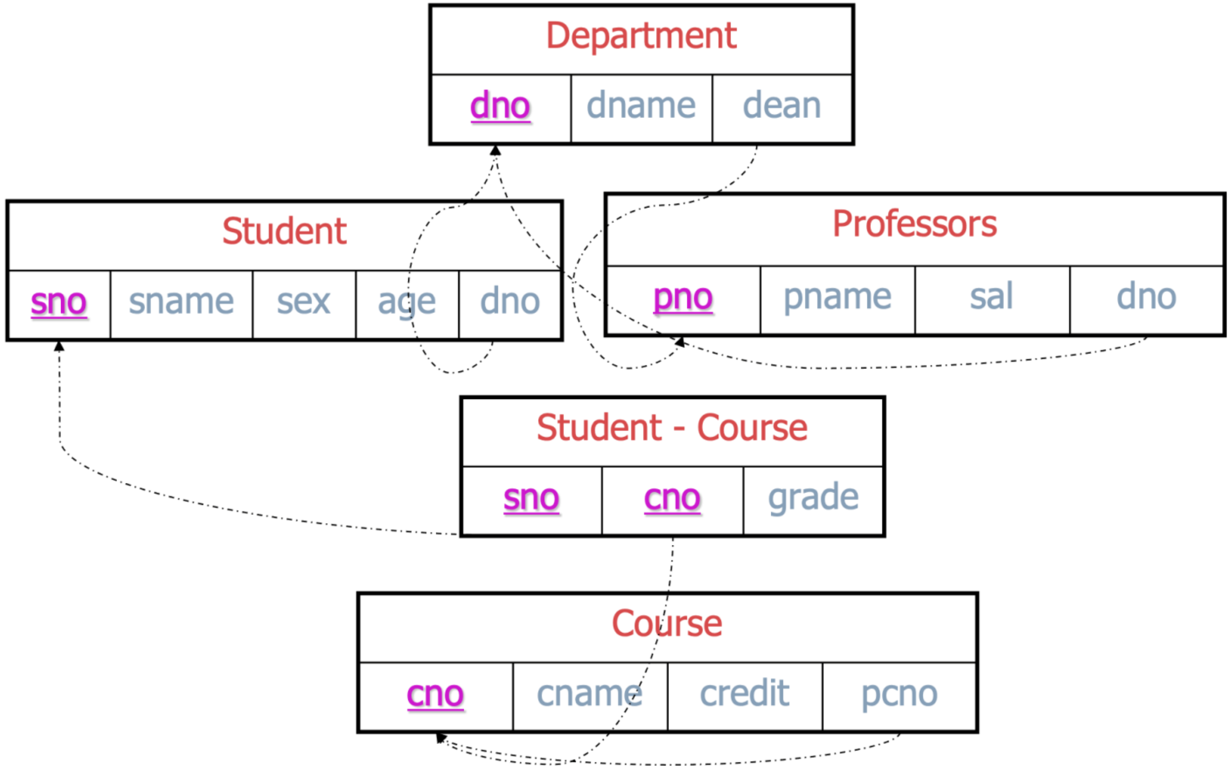
Primary Key

a candidate key chosen as the principal means of **identifying tuples within a relation** (unique & stable). Should choose an attribute that does not change with time.

e.g. *dno or dname in DEPT relations.*

Task 5: What could be the primary key S relations?

Task 6: What could be the primary key C relations?



Department
dno | dname | dean

Student
sno | sname | sex | age | dno

Professors
pno | pname | sal | dno

Student - Course
sno | cno | grade

Course
cno | cname | credit | pcno

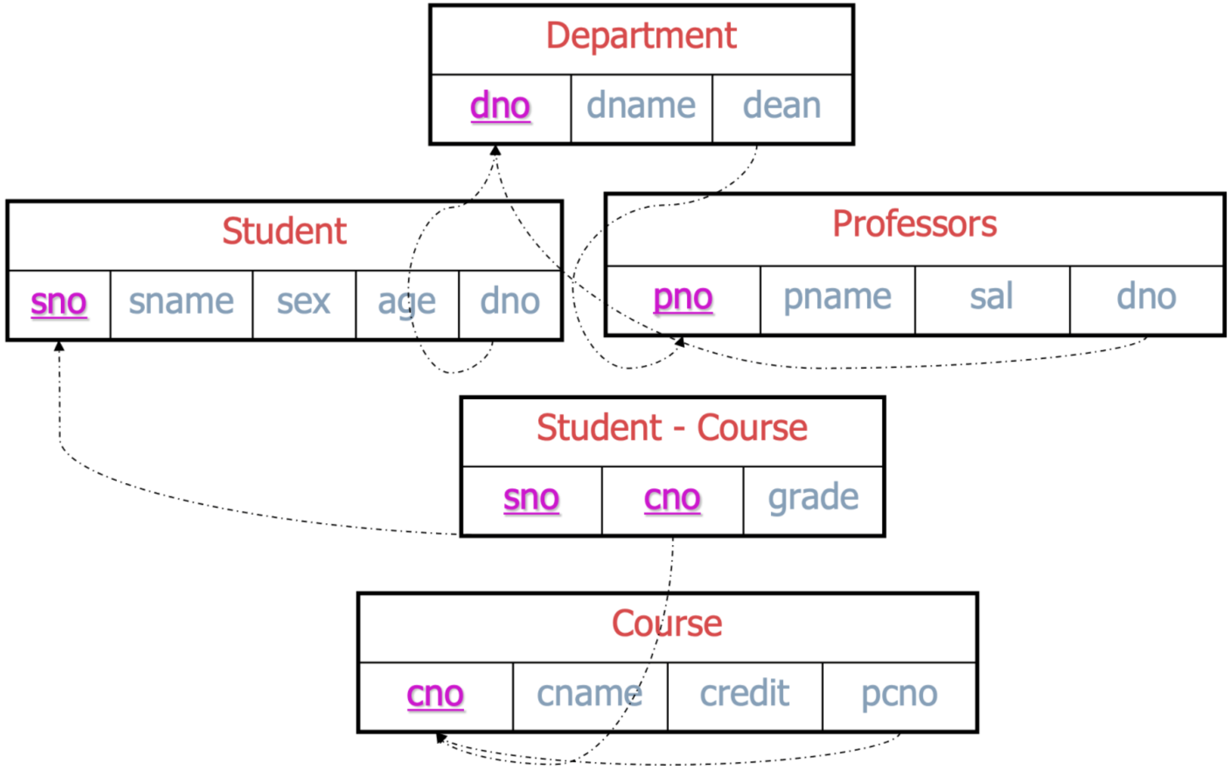
Integrity Constraints - Keys

Foreign Key

A relation schema may have an attribute that **corresponds** to the primary key of **another** relation. The attribute is called a **foreign key**.

e.g. *dno in S relations.*

Task 7: What are the other foreign keys in this database?



Department

<u>dno</u>	dname	dean

Student

<u>sno</u>	sname	sex	age	dno

Professors

<u>pno</u>	pname	sal	dno

Student - Course

<u>sno</u>	<u>cno</u>	grade

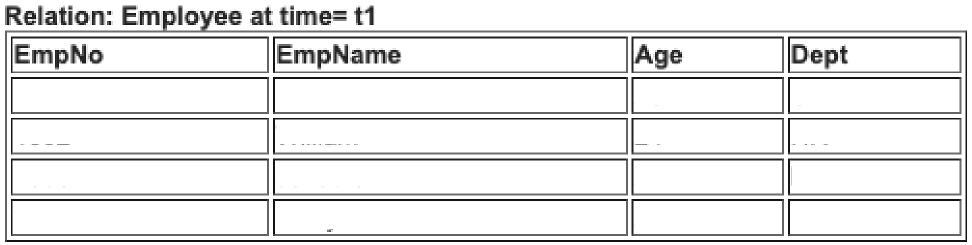
Course

<u>cno</u>	cname	credit	pcno

Relational database

Intension

The intension of a given relation **is independent of time**. It is the permanent part of the relation. It corresponds to what is specified in the relational schema.



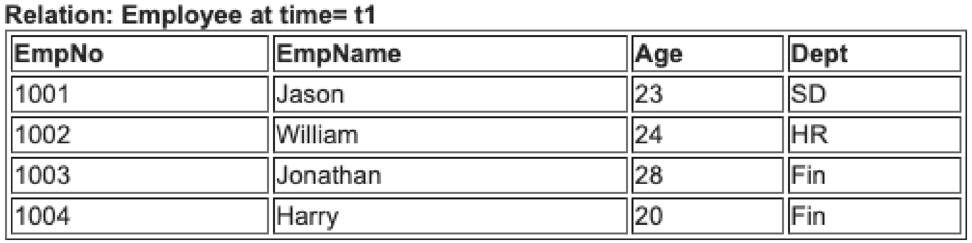
Relation: Employee at time= t1

EmpNo	EmpName	Age	Dept

The teaching building and each room in it is always there, but students inside each room at specific time may varies!

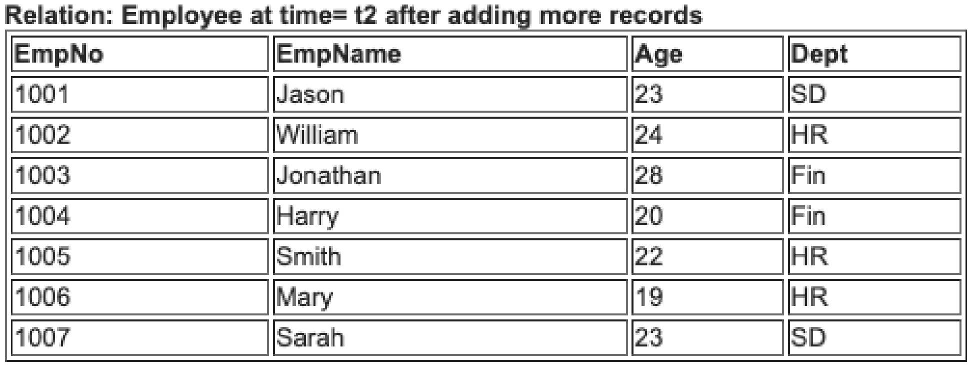
Extension

The extension of a given relation is the set of tuples appearing in that relation at any given instance. The extension thus **varies with time**. It changes as tuples are created,



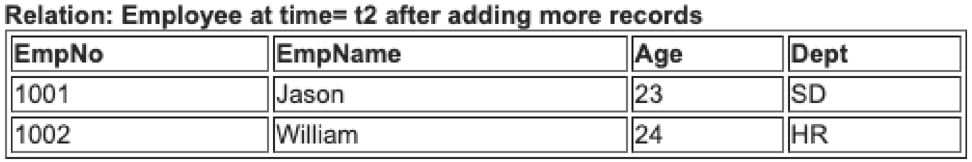
Relation: Employee at time= t1

EmpNo	EmpName	Age	Dept
1001	Jason	23	SD
1002	William	24	HR
1003	Jonathan	28	Fin
1004	Harry	20	Fin



Relation: Employee at time= t2 after adding more records

EmpNo	EmpName	Age	Dept
1001	Jason	23	SD
1002	William	24	HR
1003	Jonathan	28	Fin
1004	Harry	20	Fin
1005	Smith	22	HR
1006	Mary	19	HR
1007	Sarah	23	SD



Relation: Employee at time= t2 after adding more records

EmpNo	EmpName	Age	Dept
1001	Jason	23	SD
1002	William	24	HR

Query Language

- Query languages:
  - Language in which user requests information from the database
  - - Procedural: specify how to get the data wanted
  - - Non-procedural (declarative): only declares which data are wanted
- “Pure” languages:
  - Relational algebra, Tuple relational calculus, Domain relational calculus
  - Pure languages form underlying basis of query languages

- **Query languages:**
  - SQL: Developed by IBM for system R (Relational algebra)
  - QUEL: based on ALPHA by Codd, realized on INGRES
    (Tuple relational calculus)
  - QBE: developed by IBM (Domain relational calculus)

Relational Algebra

- Six basic operators
  - unary operation
    - select: σ
    - project: ∏
    - rename: ρ
  - Multivariate operation
    - union: ∪
    - set difference: –
    - Cartesian product: x
- Additional operations
  - Set intersection、Natural join、Division、Assignment
- Extended Relational-Algebra-Operations
  - Generalized projection、Aggregate functions、outer join
- The operators take one or two relations as inputs and produce a new relation.

LET’S DO SOME HIGH SCHOOL MATH! ☺

Select Operation - Example

- Notation:

  $\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$

  i.e. "Choosing rows"

- *p* is called the **selection predicate**, i.e. a formula consisting of: ∧ (**and**), ∨ (**or**), ¬ (**not**)

- *p* examples:

  brand = "huawei" ∧ price < 5000

  brand = "iphone" ∧ price < 6000

- Relation r

A	*B*	*C*	*D*
α	α	*1*	*7*
α	β	*5*	*7*
β	β	*12*	*3*
β	β	*23*	*10*

Task 8: $\sigma_{A=B \wedge D>5}(r)$

Project Operation - Example

- Notation:

$$\Pi_{A1,A2,\ldots Ak}\ (r)$$

where are attributes of relation *r*

i.e. "*Choosing columns*" $A_1, A_2, \ldots A_k$

- duplicated rows removed (keep unique)

- Example:

$$\Pi_{account\_number,\ balance}\ (account)$$

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

🎯 Task: $\Pi_{A,C}\ (r)$

Rename Operation - Example

- Allows us to **rename the attributes** in the results of relational-algebra expressions
- Example:

$$\rho_X(E)$$

returns the expression *E* under the name *X*

- If a relational-algebra expression *E* has arity *n*, then

$$\rho_{x(A_1, A_{2,\dots}, A_{n,})}(E)$$

returns the result of expression *E* under the name *X*, and with the attributes renamed to A_1, A_2, …., A_n .

Set-Intersection Operation - Example

- Notation: $r \cap s$
- Defined as:

$$r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$$

- Assume:
  - r, s have the *same* ***arity*** (number of attributes)
  - attributes of r and s are **compatible**

- Relation *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

Task : $r \cap s$

Set-Union Operation - Example

- Notation: $r \cup s$
- Defined as:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

- For $r \cup s$ to be valid.

1. *r, s* must have the *same* **arity**
2. The attribute domains must be **compatible**

- e.g. to find all customers with either an account or a loan

$$\Pi_{customer}\,(depositor) \;\cup\; \Pi_{customer}\,(borrower)$$

- Relations *r, s:*

A	*B*
α	*1*
α	*2*
β	*1*

r

A	*B*
α	*2*
β	*3*

s

Task: $r \cup s$

Set-Difference Operation - Example

- Notation *r* – *s*
- Defined as:

$$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$$

- Set differences must be taken between **compatible** relations.
  - *r* and *s* must have the same arity
  - attribute domains of *r* and *s* must be compatible

- Relations *r*, *s*:

A	*B*
α	1
α	2
β	1

r

A	*B*
α	2
β	3

s

Task: *r* – *s*

Cartesian-Product Operation - Example

- Notation *r* x *s*
- Defined as:

$$r \text{ x } s = \{t\, q \mid t \in r \textbf{ and } q \in s\}$$

- Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S = \emptyset$).
- If attributes of *r(R)* and *s(S)* are not disjoint, then need **rename** at first.

- Relations *r, s*:

A	B
α	*1*
β	*2*

r

C	D	E
α	*10*	*a*
β	*10*	*a*
β	*20*	*b*
γ	*10*	*b*

s

🎯 Task: *r* x *s*

Composition-of Operation - Example

- Can build expressions using multiple operations

- Example: $\sigma_{A=C}(r \times s)$
- already know *r x s:*

A	*B*	*C*	*D*	*E*
α	*1*	α	*10*	*a*
α	*1*	β	*10*	*a*
α	*1*	β	*20*	*b*
α	*1*	γ	*10*	*b*
β	*2*	α	*10*	*a*
β	*2*	β	*10*	*a*
β	*2*	β	*20*	*b*
β	*2*	γ	*10*	*b*

🎯 Task: $\sigma_{A=C}(r \times s)$

Natural-join Operation - Example

- Notation: r ⋈ s
- r ⋈ s is a relation on schema $R \cup S$ obtained as follows:
  - Consider each pair of tuples t_r from r and t_s from s on **shared attributes**
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple t to the result, where
    - t has the same value as t_r on r
    - t has the same value as t_s on s

- Relations r, s:

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	$\in$

s

Task: r ⋈ s

Division Operation - Example

- Notation: $r \div s$
- Suited to queries that include the phrase “for all”.
- Let *r* and *s* be relations on schemas *R* and *S* respectively where
  - $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - $S = (B_1, \ldots, B_n)$

  The result of $r \div s$ is a relation on schema

  $R - S = (A_1, \ldots, A_m)$

$$r \div s = \{\, t \mid t \in \prod_{R\text{-}S}(r) \wedge \forall\, u \in s\ (\, tu \in r\,)\,\}$$

Where *tu* means the concatenation of tuples *t* and *u* to produce a single tuple

- Relations *r, s*:

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
$\in$	6
$\in$	1
β	2

r

B
1
2

s

Task: $r \div s$

Generalized Projection

- Extends the projection operation by **allowing arithmetic functions** to be used in the projection list.

$$\Pi_{F_1, F_2, \ldots, F_n}(E)$$

- *E* is any relational-algebra expression
- Each of F_1, F_2, …, F_n are are arithmetic expressions involving constants and attributes in the schema of *E*.
- e.g.

Given relation *credit_info(customer_name, limit, credit_balance),* find how much more each person can spend:

$$\Pi_{customer\_name,\ limit\ -\ credit\_balance}\ (credit\_info)$$

Aggregate Functions and Operations

- **Aggregation function** takes a collection of values and returns a single value as a result.

  avg: average value
  min: minimum value
  max: maximum value
  sum: sum of values
  count: number of values

- **Aggregate operation** in relational algebra

$$_{G_1, G_2, \ldots, G_n}\vartheta_{F_1(A_1), F_2(A_2), \ldots, F_n(A_n)}(E)$$

E is any relational-algebra expression

- G_1, G_2 …, G_n is a list of attributes on which to group (can be empty)
- Each F_i is an aggregate function
- Each A_i is an attribute name

- Relation *r*:

A	*B*	*C*
α	α	7
α	β	7
β	β	3
β	β	10

Task: $g_{\text{sum(c)}}$ (r)

Aggregate Functions and Operations

- Relation *account* **grouped by** *branch-name*:

branch_name	*account_number*	*balance*
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

Task: ${}_{branch\_name}\,\mathcal{G}\,{}_{\mathbf{sum}(balance)}\ (account)$

Outer Join

- An extension of the join operation that avoids loss of information.
- Computes the join and then **adds tuples form one relation that does not match tuples in the other relation** to the result of the join.
- Uses *null* values:
  - *null* signifies that the value is unknown or does not exist
  - All comparisons involving *null* are (roughly speaking) **false** by definition.
    - We shall study precise meaning of comparisons with nulls later

Outer Join

- Relation *loan*

loan_number	*branch_name*	*amount*
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

- Relation *borrower*

customer_name	*loan_number*
Jones	L-170
Smith	L-230
Hayes	L-155

- Join

loan ⋈ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

- Left Outer Join

loan ⟕ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*

Null Values

- It is possible for tuples to have a null value, denoted by *null*, for some of their attributes
- *null* signifies an unknown value or that a value does not exist.
- *null* should best be avoided theoretically, though in practice they are used. *Null* could cause potential problems:
  - DBMSs (e.g. MySQL) may treat *null* in some way, e.g. arithmetic expression involving *null* as *null*, aggregate functions ignoring null values.
  - However, no such guarantee for all DBMSs

Modification of the Database

- The content of the database may be modified using the following operations:
  - Deletion
  - Insertion
  - Updating
- All these operations are expressed using the assignment operator.

Deletion

- A delete request is expressed similarly to a query, except instead of displaying tuples to the user, the selected tuples are removed from the database.
- Can delete only whole tuples; cannot delete values on only particular attributes
- A deletion is expressed in relational algebra by:

$$r \leftarrow r - E$$

where r is a relation and E is a relational algebra query.

Insertion

- To insert data into a relation, we either:
  - specify a tuple to be inserted
  - write a query whose result is a set of tuples to be inserted
- in relational algebra, an insertion is expressed by:

  $$r \leftarrow r \cup E$$

  where *r* is a relation and *E* is a relational algebra expression.
- The insertion of a single tuple is expressed by letting *E* be a constant relation containing one tuple.

Updating

- A mechanism to **change a value in a tuple** without charging *all* values in the tuple
- Use the generalized projection operator to do this task

$$r \leftarrow \prod_{F_1,F_2,\ldots,F_l,}(r)$$

- Each F_i is either
  - the l^{th} attribute of *r*, if the l^{th} attribute is not updated, or,
  - if the attribute is to be updated F_i is an expression, involving only constants and the attributes of *r*, which gives the new value for the attribute

Summary for Part 2

- **Structure of Relational Databases**
  - Relations, domain, attributes, Schema, tuples, Keys, intension/extension
- **Fundamental Relational-Algebra-Operations**
  - Select, project, union, set difference, cartesian product, rename
- **Additional Relational-Algebra-Operations**
  - Set intersection, natural join, division, assignment
- **Extended Relational-Algebra-Operations**
  - Generalized projection, aggregate functions, outer join
- **Null Values**
- **Modification of the Database**
  - Deletion, insertion, updating

FINALLY!
THE END OF THE PRESENTATION