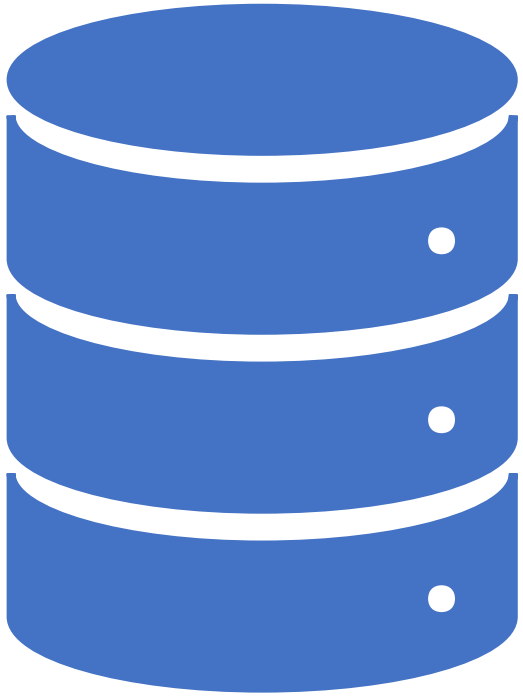


Introduction to Databases

Data

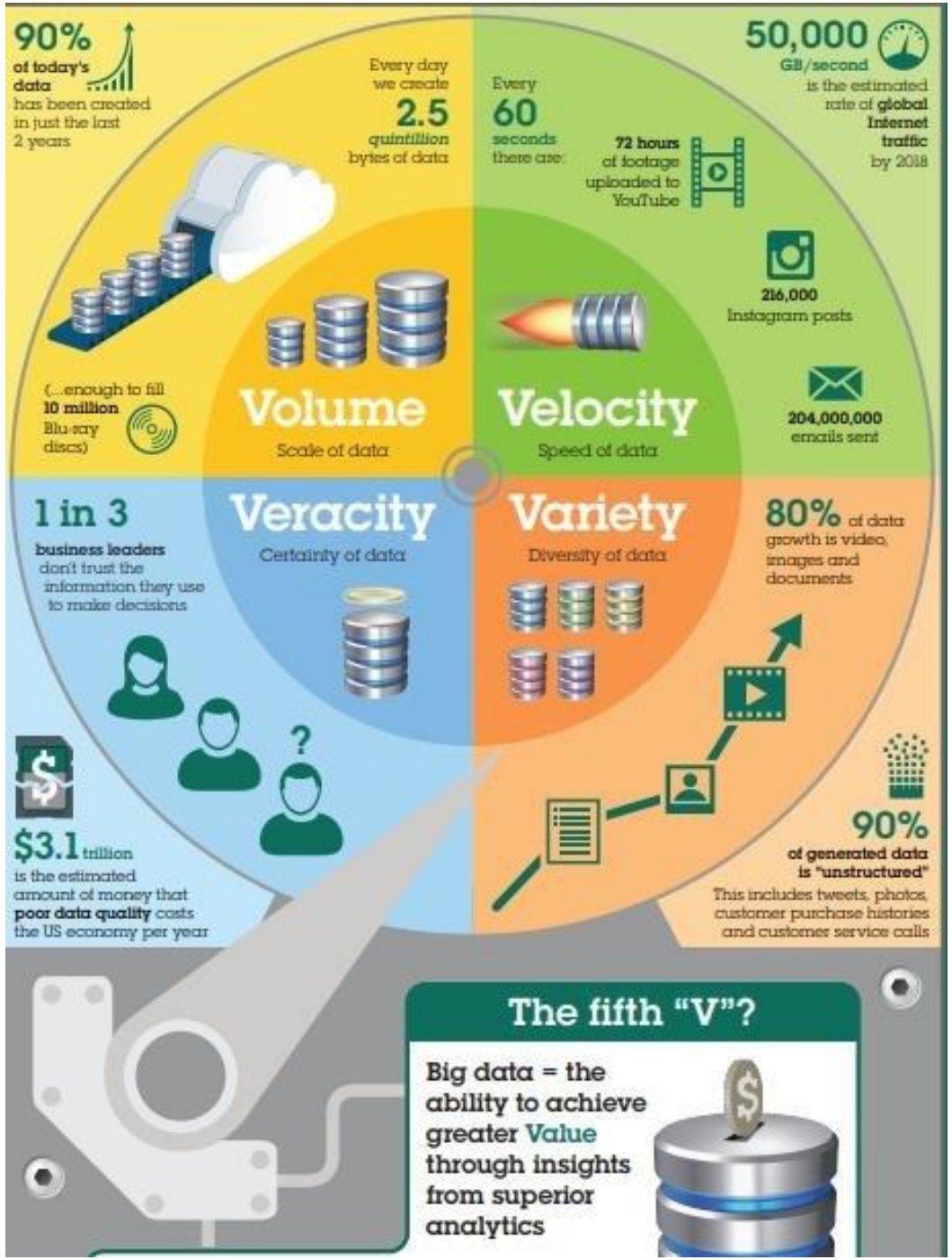
- What is data?
- Collection of information.
- Data is everywhere.
- Huge amount of data is generated daily.
- Personal, WWW, organizational, scientific.
- Can we do something from this data?
  - Structured/ unstructured data.
  - Information and knowledge.

Data Explosion

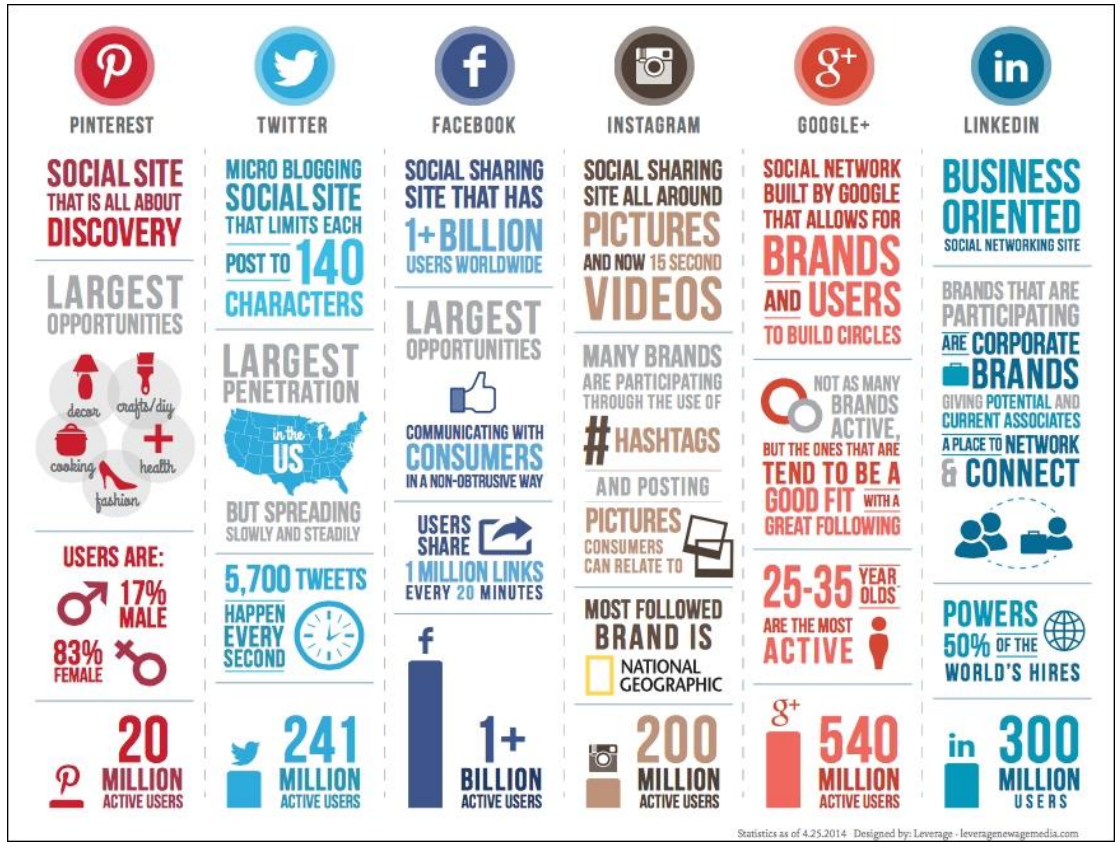
- **Much more is produced every day**
- Walmart: 583 terabytes of sales and inventory data
  - Adds a billion rows every day
  - “we know how many 2.4 ounces of tubes of toothpastes sold yesterday and what was sold with them”
- A single astrophysics simulation of galaxy formation can generate several PB of data, most of it thrown away
- Machine generated data
  - Sensor devices/networks, microphones/camera, Web server logs, Network Monitoring, etc.
- Online services (old data)
  - Twicer: 177M tweets sent on 3/1/2011 (nothing special about the date)
  - Dropbox: 1M files saved every 15 mins
  - Facebook: 135+ Billion Messages a Month
  - Reddit: 270 Million Page Views a Month in May 2010

Data Explosion

- Much of this data is not stored in traditional RDBMS
- **A major challenge is to manage this data, answer queries over it, gather interesting and useful insights from it**
- **“Big Data”**
  - Everyone is either doing big data, or wants to...
  - No one seems to agree on what it really means !
  - Not just about scale/volume of data
- **“Data Scientist”**
  - Goal: Extract meaning from data and creating data products
  - Need a broad range of skills: programming, statistics, math, ...


90%
of today's data
has been created in just the last 2 years
Every day we create
2.5
quintillion
bytes of data
(...enough to fill 10 million Blu-ray discs)
Volume
Scale of data
Every
60
seconds there are:
72 hours
of footage uploaded to YouTube
50,000
GB/second
is the estimated rate of global Internet traffic by 2018
216,000
Instagram posts
204,000,000
emails sent
Velocity
Speed of data
1 in 3
business leaders
don't trust the information they use to make decisions
Veracity
Certainty of data
Variety
Diversity of data
80% of data growth is video, images and documents
$3.1 trillion
is the estimated amount of money that poor data quality costs the US economy per year
90%
of generated data is "unstructured"
This includes tweets, photos, customer purchase histories and customer service calls
The fifth "V"?
Big data = the ability to achieve greater Value through insights from superior analytics

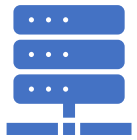
Usage of data



Data Management

- **A large fraction of the data is still in traditional DBMS systems**
  - Still open and active research areas about improving performance, energy efficiency, new functionalities, and so on...
- **Much of the data not stored in traditional database systems**
  - For a variety of fairly valid reasons
  - Stream processing systems (focusing on *streaming* data)
  - Batch analysis frameworks (like Hadoop, Pregel)
    - Typically data stored in distributed file systems
  - Key-value stores (like Redis, MongoDB, ...)
  - Semi-structured data stores (for XML query processing)
  - Graph databases (somewhat new)
- However, many lessons to be learned from database research

Databases

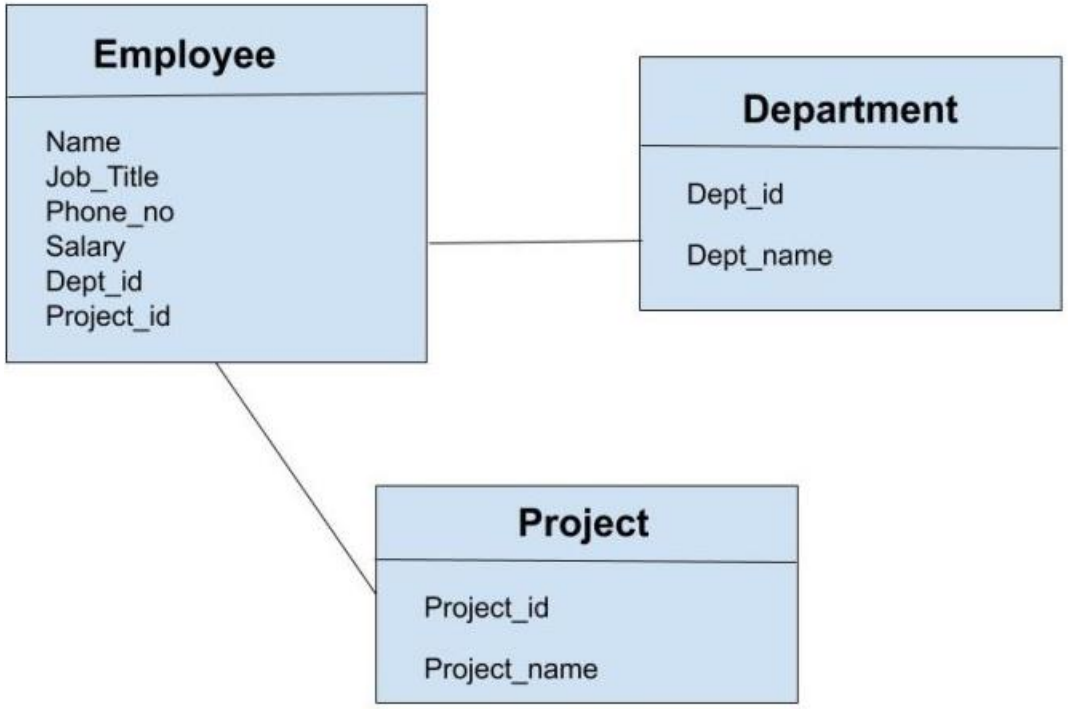


Organized collection (large and related) of data.

Why database?

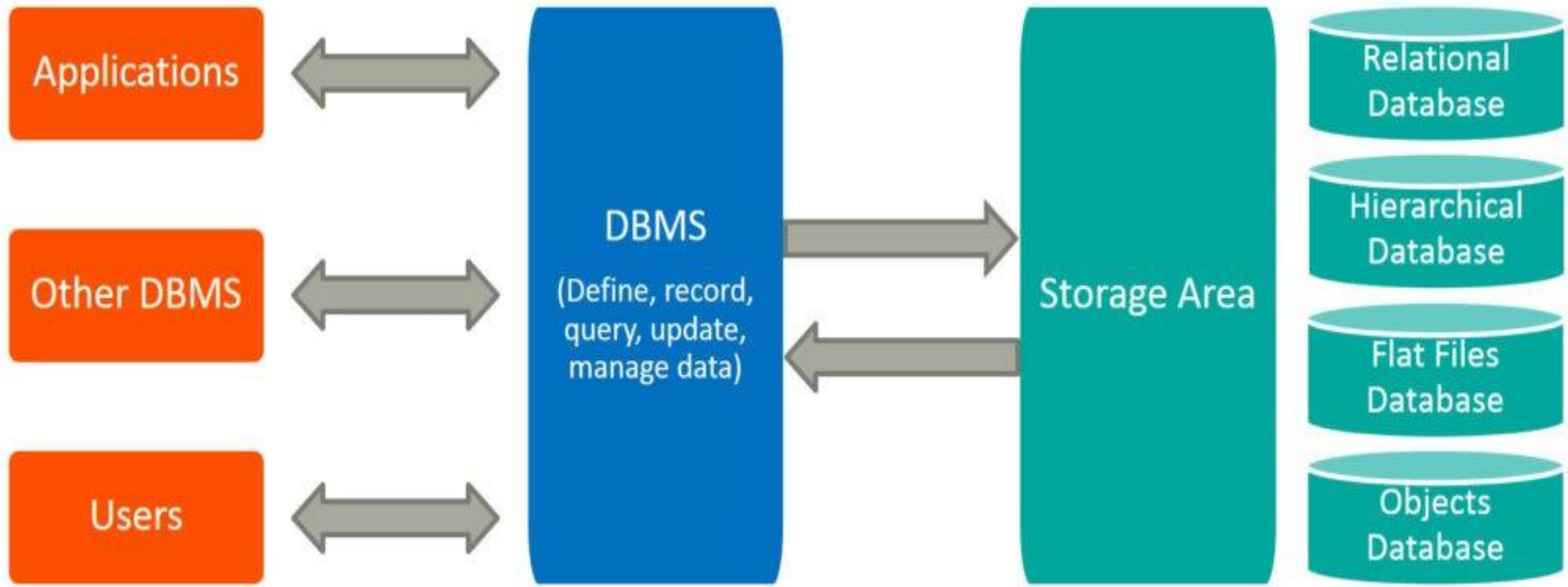
Analyze, manage and use data efficiently.

Data is accessed, manipulated and updated.



Schema

Database Management System



Provide environment to access, manage the database.

Database Management System

Software system that stores and manages database

Contains information about a particular organization

- Collection of interrelated data
- Set of programs to access the data
- An environment that is both convenient and efficient to use

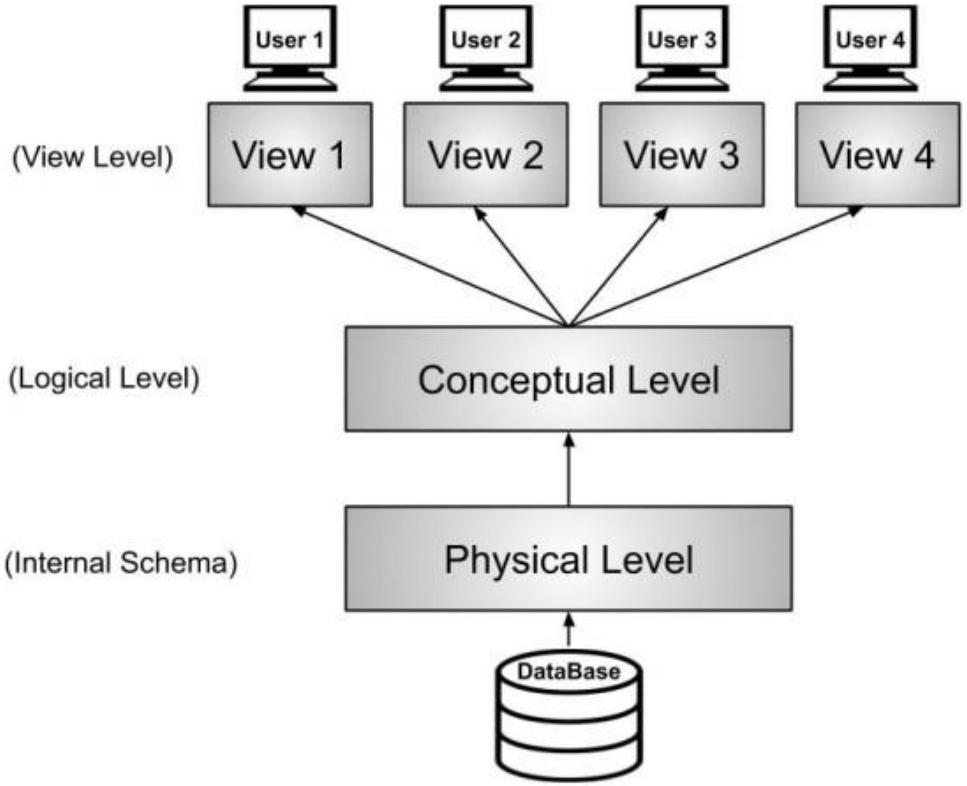
Applications

- Sales: customers, products, purchases
- Universities: registration, grades
- Airlines: reservations, schedules
- Banking: transactions
- Manufacturing: production, inventory, orders, supply chain

Databases can be very large.

Levels of Abstraction

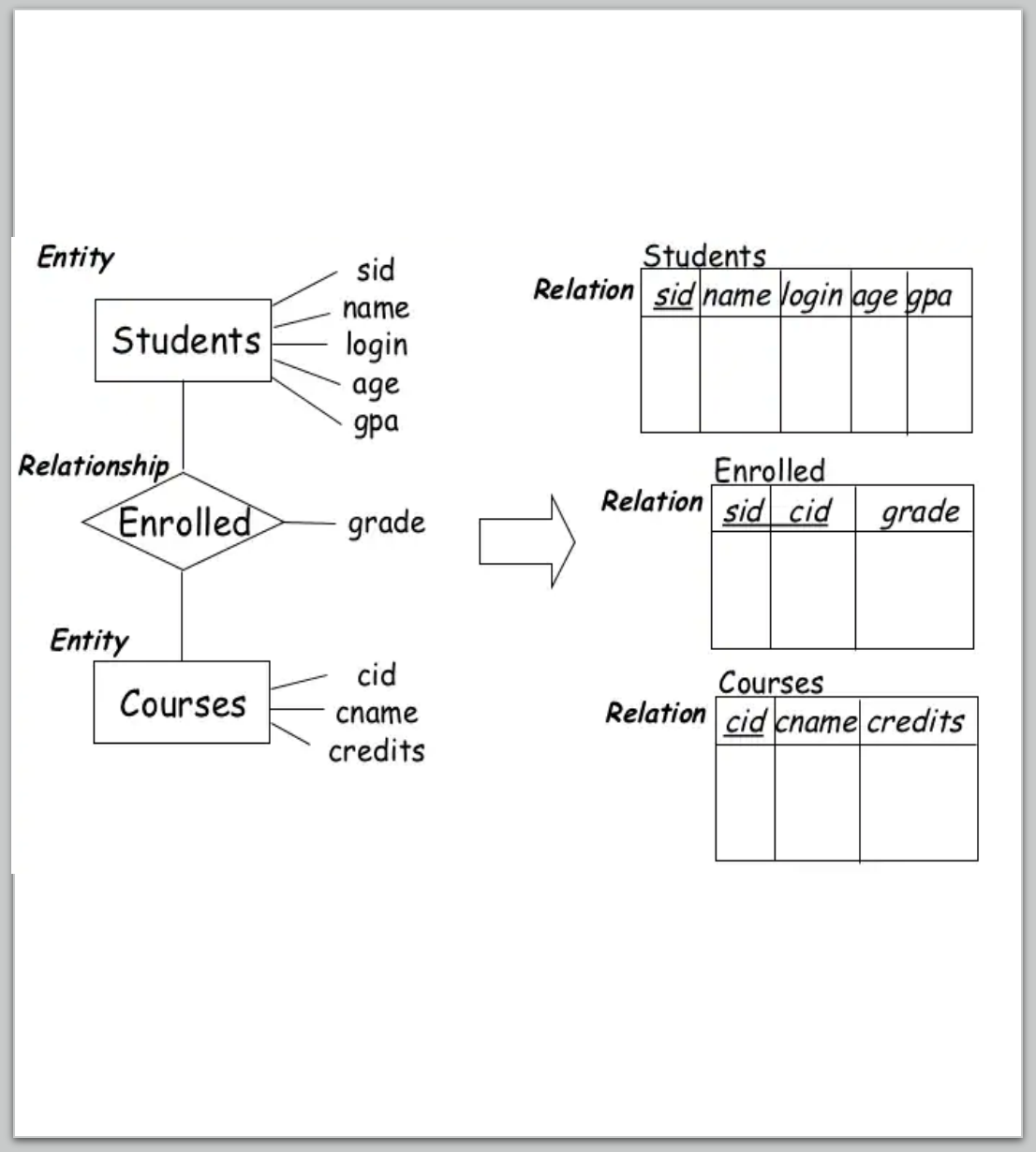
- Many views, single conceptual (logical) schema and physical schema
- **Physical level:** describes how a record is stored
- **Logical level**: describes data stored in database, and the relationships among the data
- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.



Levels of Data Abstraction

Example: University database

- **Conceptual schema:**
  - Students(sid: string, name: string, login: string, age: integer, gpa:real)
  - Courses(cid: string, cname:string, credits:integer)
  - Enrolled(sid:string, cid:string, grade:string)
- **Physical schema:**
  - Relations stored as unordered files.
  - Index on first column of Students.
- **External Schema (View):**
  - Course_info(cid:string,enrollment:integer)

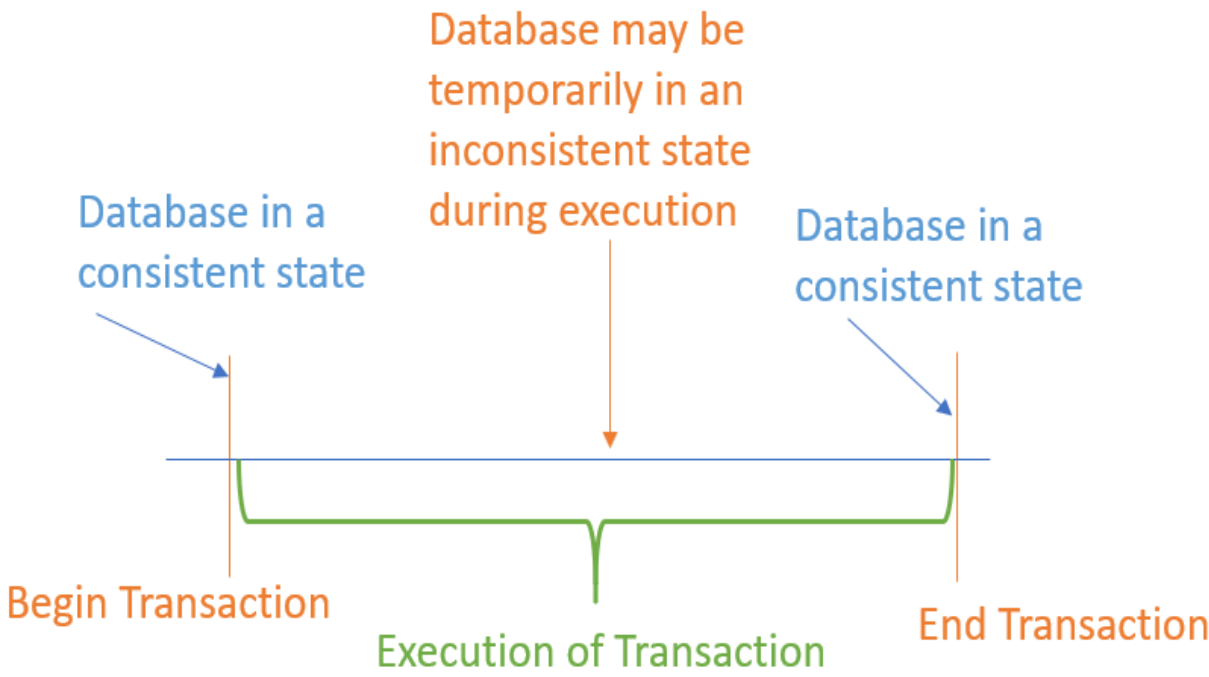


Data Independence

- A major advantages of DBMS is they provide data abstraction.
- Able to modify the schema at one level of the database system without altering the schema at the next higher level.
- Applications insulated from how data is structured and stored.
- Query doesn't need to know how the data is stored.
- **Logical data independence:** Protection from changes in logical structure of data.
  - If a user application operates on a subset of attributes of a relation, it should not affect when new attributes are added. Can make changes without affecting existing schemas.
- **Physical data independence:** Protection from changes in physical structure of data.
  - Change in internal data using different file organization or devices should be possible without changing logical schema structure.

Database Properties

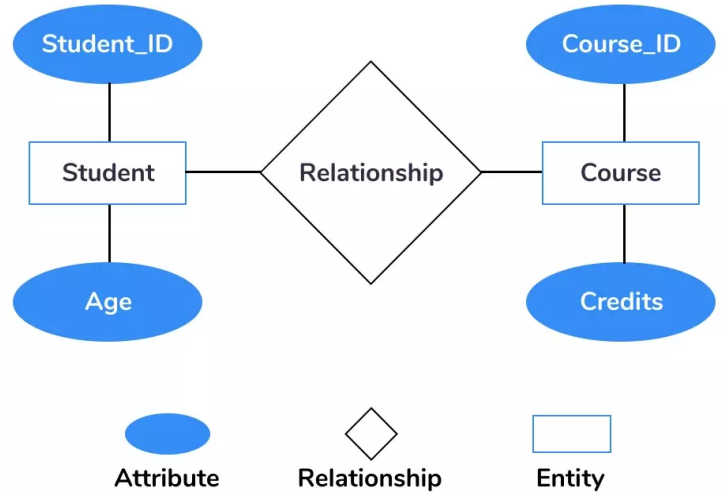
- To maintain consistency in a database, before and after the transaction, ACID properties are followed.
    1. **Atomicity:** If any operation is performed on the data, either it should be performed or executed completely or should not be executed at all.
    2. **Consistency:** Integrity of the data should be maintained, which means if a change in the database is made, it should remain preserved always.
    3. **Isolation:** Transactions occur independently without interference. Changes should be visible only after they have been made to the main memory.
    4. **Durability:** Data after the successful execution of the operation becomes permanent in the database.



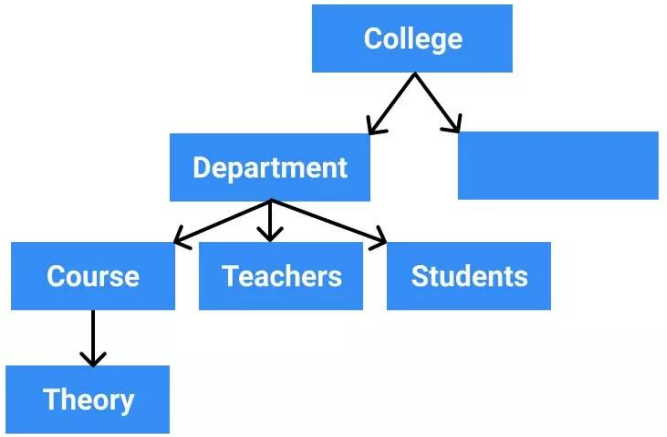
Data Models

- A data model is formal framework for describing data.
- Collection of concepts for describing data, relationships and constraints.
- How to store & organize the data?
  - How many attributes are really needed about a student/course/faculty?
  - What is an efficient way to organize the data?
- How to query the data and generate reports for the end users ?

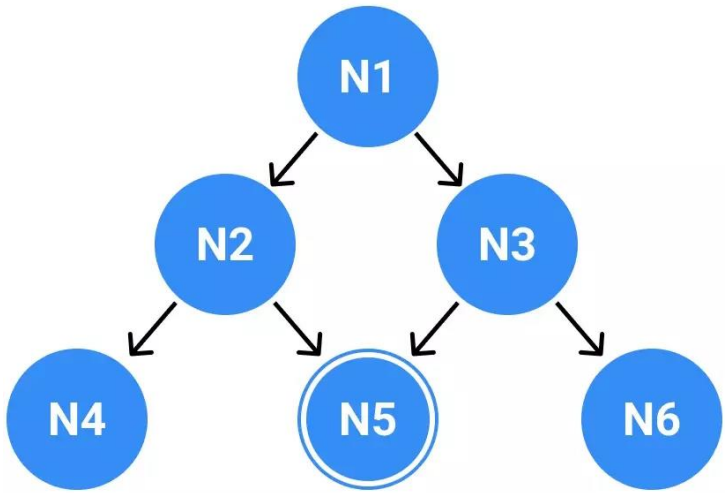
Entity Relationship Model



Hierarchical Model



Network Model



Relational Model

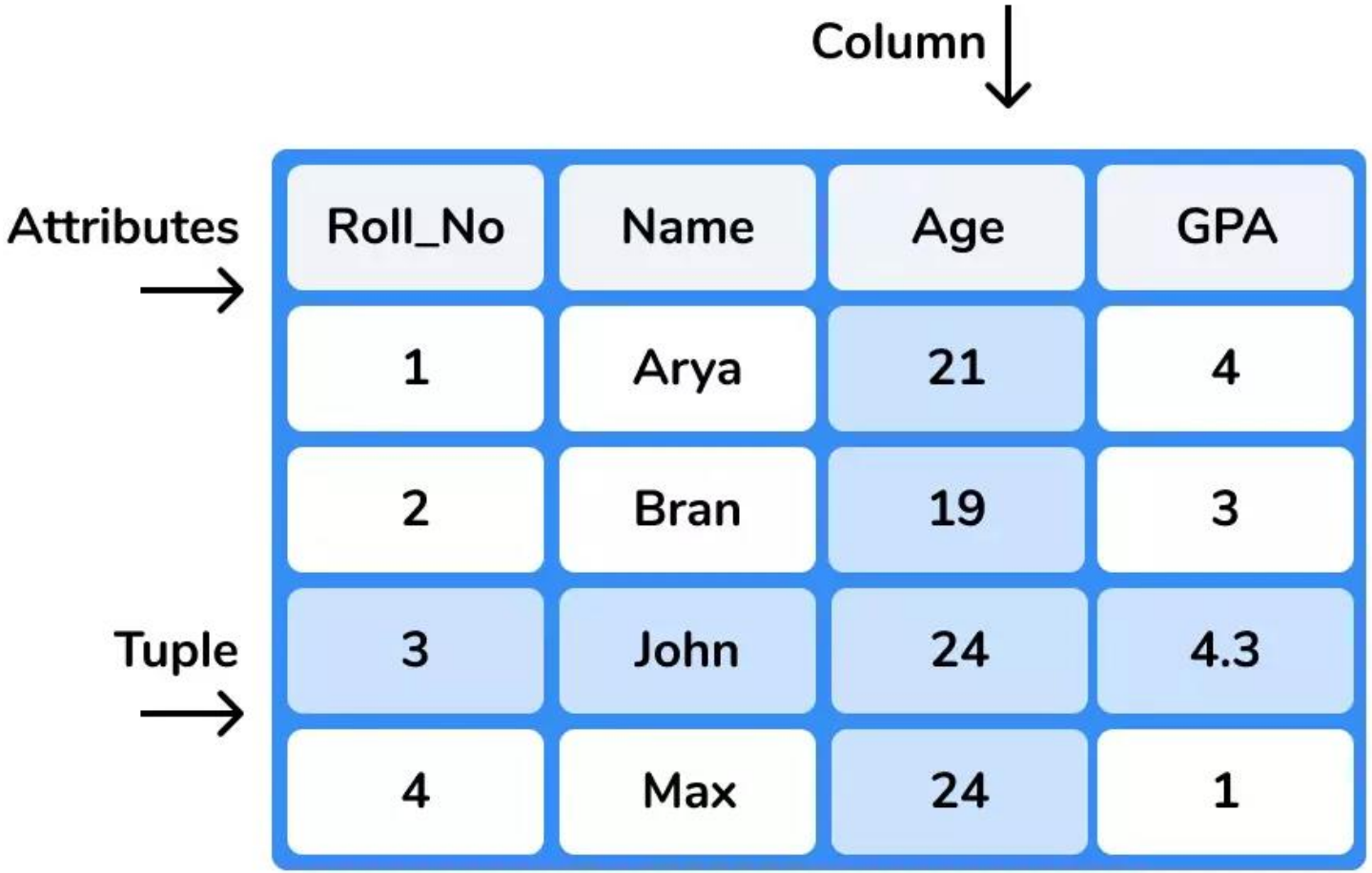
Roll_No	Name
1	Arya
2	Bran
3	John

Course_ID	Sub_Name
IT_101	Java
IT_102	C++
IT_103	DBMS

Roll_No	Course_ID	Marks
1	IT_101	100
1	IT_102	99
3	IT_103	93

Relational Model

- All the data is stored in various tables.
- A theory of normalization guides you in designing relations.



Data Definition Language (DDL)

Specific notation for defining the database schema.

Used to create and modify the structure of database objects in the database.

DDL compiler generates a set of table templates stored in a data dictionary.

Data dictionary contains metadata (i.e., data about data)

- Database schema
- Integrity constraints
  - Primary key (ID uniquely identifies instructors)
- Authorization
  - Who can access what

Data Manipulation Language (DML)

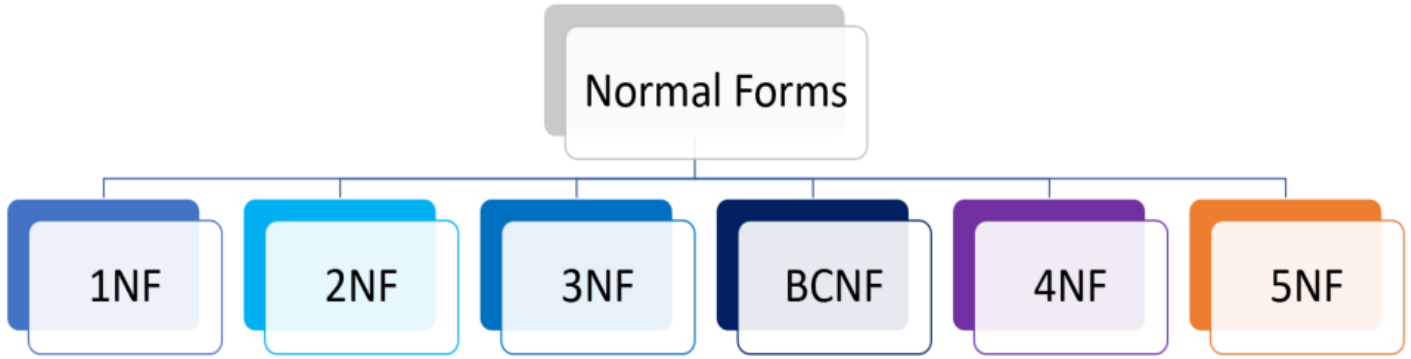
- Language for accessing and manipulating the data organized by the appropriate data model.
  - DML is also known as query language
- Two classes of languages
  - Pure – used for proving properties about computational power and for optimization
    - Relational Algebra
    - Tuple relational calculus
  - Commercial – used in commercial systems
    - SQL is the most widely used commercial language

Other Languages

- DCL
  - Grant and take back authority from any database user.
- DQL
  - Fetch the data from the database.
  - Perform queries on the data to get some schema relation based on the query passed to it.
- TCL
  - Deals with the transaction within the database.
  - can only be used with DML commands

Data Normalization

- Process of organizing the data in the database.
- Reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization rules divides larger tables into smaller tables and links them using relationships.
- Normal form is used to reduce redundancy from the database table.



Redundancy in databases

- Redundancy in a database denotes the repetition of stored data.
- Redundancy might cause various anomalies and problems pertaining to storage requirements:
- **Insertion anomalies:** It may be impossible to store certain information without storing some other, unrelated information.
- **Deletion anomalies:** It may be impossible to delete certain information without losing some other, unrelated information.
- **Update anomalies:** If one copy of such repeated data is updated, all copies need to be updated to prevent inconsistency.
- Increasing storage requirements: The storage requirements may increase over time.
  These issues can be addressed by decomposing the database – normalization forces this!!!

Insertion Anomaly

- Consider the following table (the attributes are not null) detailing some of the cars available in the market.

Company	Country	Make	Distributor
Maruti	India	WagonR	Carwala
Maruti	India	WagonR	Bhalla
Toyota	Japan	RAV4	CarTrade
BMW	Germany	X1	CarTrade

- Suppose Tesla, a company from US, is now collaborating with Toyota to bring the make RAV4 in the market with no distributor announced yet.
- This insertion is not possible in the above table as the Distributor cannot be null.

Deletion Anomaly

- Consider the following table (the attributes are not null) detailing some of the cars available in the market.

Company	Country	Make	Distributor
Maruti	India	WagonR	Carwala
Maruti	India	WagonR	Bhalla
Toyota	Japan	RAV4	CarTrade
BMW	Germany	X1	CarTrade

- Suppose CarTrade is no more a distributor for the make X1 of BMW, a company from Germany.
- This deletion from the above table would result in the car record being deleted.

Updation Anomaly

- Consider the following table (the attributes are not null) detailing some of the cars available in the market.

Company	Country	Make	Distributor
Maruti	India	WagonR	Carwala
Maruti	India	WagonR	Bhalla
Toyota	Japan	RAV4	CarTrade
BMW	Germany	X1	CarTrade

- Suppose Maruti is no more an Indian company due to its 100% procurement by Suzuki Motor Corporation, a company from Japan.
- This update is to be made in multiple records in the above table resulting into atomicity challenges.

Keys and Functional Dependency

- **Candidate Key:** The minimal set of attributes that can uniquely identify a tuple
- **Primary Key:** There can be more than one candidate key in relation out of which one can be chosen as the primary key.
- **Super Key:** The set of attributes that can uniquely identify a tuple is known as Super Key.
- **Foreign Key:** acts as a primary key in one table and it acts as secondary key in another table. It combines two or more relations (table) at a time.
- **Functional Dependency (X->Y)**: Y is dependent on X. We can find out Y using X.

1NF

- Each table cell should contain a single value.
- Each record needs to be unique.

Student#	Advisor	Adv-Room	Class1	Class2	Class3
1022	Jones	412	101-07	143-01	159-02
4123	Smith	216	201-01	211-02	214-01

Student#	Advisor	Adv-Room	Class#
1022	Jones	412	101-07
1022	Jones	412	143-01
1022	Jones	412	159-02
4123	Smith	216	201-01
4123	Smith	216	211-02
4123	Smith	216	214-01

2NF

- Be in 1NF
- All non-key attributes are fully functionally dependent on the primary key

Primary Key:

Single column value used to identify a database record uniquely.

Students:

Student#	Advisor	Adv-Room
1022	Jones	412
4123	Smith	216

Registration:

Student#	Class#
1022	101-07
1022	143-01
1022	159-02
4123	201-01
4123	211-02
4123	214-01

3NF

- Be in 2NF
- Has no transitive functional dependencies.

Transitive Functional Dependencies:

Changing a non-key column, might cause any of the other non-key columns to change

Students:

Student#	Advisor
1022	Jones
4123	Smith

Faculty:

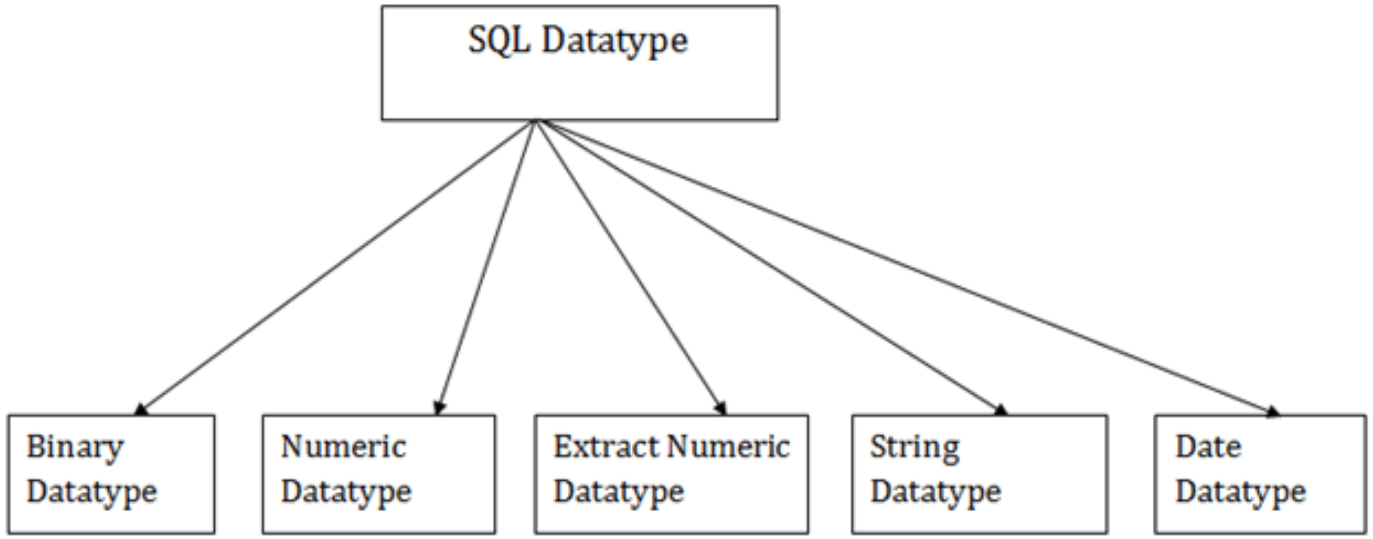
Name	Room	Dept
Jones	412	42
Smith	216	42

SQL

- SQL stands for Structured Query Language. It is used for storing, manipulating and retrieving data in databases.
- All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL Server use SQL as their standard database language.
- SQL allows users to query the database in number of ways, using English-like statements.
- It is not case sensitive. Generally, keywords of SQL are written in uppercase.

SQL Datatype

- SQL Datatype is used to define the values that a column can contain.
- Every column is required to have a name and data type in the database table.



Create Table Construct

- An SQL relation is defined using the create table command:

create table r (A_1 D_1 , A_2 D_2 , ..., A_n D_n ,
(integrity-constraint 1),...,
(integrity-constraint k))

- Example:

create table instructor (
ID **char**(5),
Name **varchar**(20) **not null**,
dept_name **varchar**(20),
Salary **numeric**(8,2),
primary key (ID),
foreign key (dept_name) **references** department)

insert into instructor **values** ('10211', 'Smith', 'Biology', 66000);
insert into instructor **values** ('10211', null, 'Biology', 66000);

- primary key declaration on an attribute automatically ensures not null

Drop and Alter Table Constructs

- drop table student
  - Deletes the table and its contents
- delete from student
  - Deletes all contents of table, but retains table
- alter table
  - alter table r add A D
    - where A is the name of the attribute to be added to relation r and D is the domain of A.
    - All tuples in the relation are assigned null as the value for the new attribute.
  - alter table r drop A
    - Where A is the name of an attribute of relation r

Basic Query Structure

- The SQL data-manipulation language (DML) provides the ability to query information, and insert, delete and update tuples.
- A typical SQL query has the form:

select A_1 , A_2 , ..., A_n

from r_1 , r_2 , ..., r_m

where P

- The select clause list the attributes desired in the result of a query
- Example: find the names of all instructors:

select name

from instructor

The select Clause

- SQL allows duplicates in relations as well as in query results.
- To force the elimination of duplicates, insert the keyword **distinct** after select.
- Find the names of all departments from instructor, and remove duplicates.

select distinct dept_name

from instructor

- The keyword **all** specifies that duplicates should not be removed.

select all dept_name

from instructor

- An asterisk in the select clause denotes "all attributes".

The where Clause

- The where clause specifies conditions that the result must satisfy.
- To find all instructors in Comp. Sci. dept with salary > 80000

select name

from instructor

where dept_name = 'Comp. Sci.' **and** salary > 80000

- Comparison results can be combined using the logical connectives **and**, **or**, and **not**.
- Comparisons can be applied to results of arithmetic expressions.

The from Clause

- The from clause lists the relations involved in the query
  - Corresponds to the Cartesian product operation of the relational algebra.
- Find the Cartesian product instructor X teaches

select ∗

from instructor, teaches

  - generates every possible instructor – teaches pair, with all attributes from both relations
- Cartesian product is not very useful directly, but useful when combined with where-clause condition (selection operation in relational algebra).

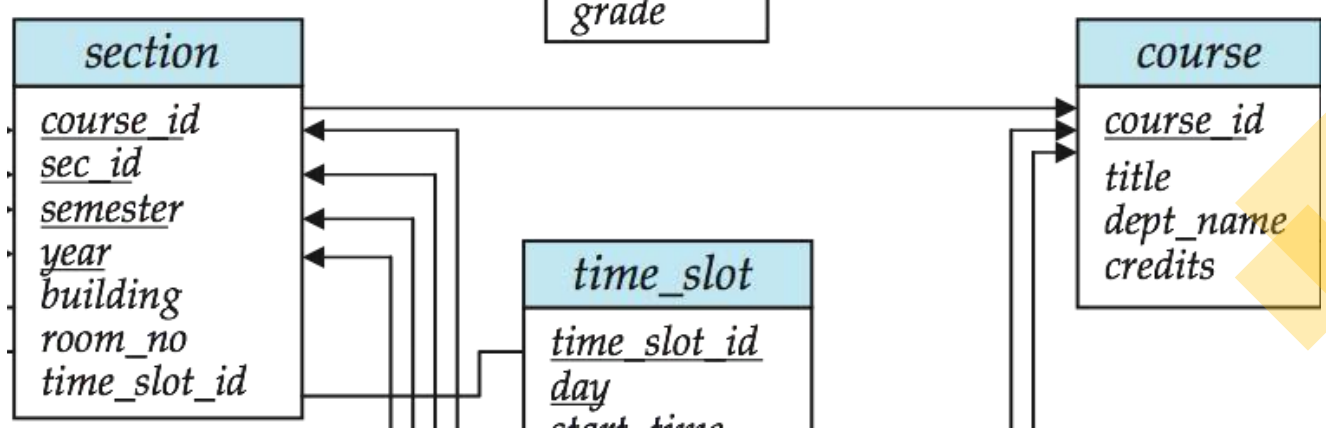
Cartesian Product: instructor X teaches

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

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009

inst.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2009
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2009
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2010
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2009
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2010
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2010
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2009
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

Joins



- For all instructors who have taught some course, find their names and the course ID of the courses they taught.

select name, course_id

from instructor, teaches

where instructor.ID = teaches.ID

- Find the course ID, semester, year and title of each course offered by the Comp. Sci. department

select section.course_id, semester, year, title

from section, course

where section.course_id = course.course_id **and** dept_name = 'Comp. Sci.'

Natural Join

- Natural join matches tuples with the same values for all common attributes, and retains only one copy of each common column

select

from instructor **natural join** teaches;

ID	*name*	*dept_name*	*salary*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010

Example:

- List the names of instructors along with the course ID of the courses that they taught.

select name, course_id

from instructor, teaches

where instructor.ID = teaches.ID;

select name, course_id

from instructor **natural join** teaches;

The Rename Operation

- The SQL allows renaming relations and attributes using the as clause:

old-name **as** new-name

- Find the names of all instructors who have a higher salary than some instructor in 'Comp. Sci'.

select distinct T. name

from instructor **as** T, instructor **as** S

where T.salary > S.salary **and** S.dept_name = 'Comp. Sci.'

- Keyword as is optional and may be omitted

instructor **as** T ≡ instructor T

String Operations

- The operator "like" uses patterns that are described using two special characters:
  - percent ( ). The character matches any substring.
  - underscore (_). The _ character matches any character.
- Find the names of all instructors whose name has the substring "dar".

select name

from instructor

where name **like** '%dar%'

- Patterns are case sensitive.
- Pattern matching examples:
  - ‘Intro%’ matches any string beginning with “Intro”.
  - ‘%Comp%’ matches any string containing “Comp” as a substring.
  - ‘_ _ _’ matches any string of exactly three characters.
  - ‘_ _ _ %’ matches any string of at least three characters.

Ordering the Display of Tuples

- List in alphabetic order the names of all instructors

select distinct name

from instructor

order by name

- We may specify **desc** for descending order or **asc** for ascending order, for each attribute; ascending order is the default.
- Example: **order by** name **desc**
- Can sort on multiple attributes
- Example: **order by** dept_name, name

Set Operations

- Set operations **union**, **intersect**, and **except**
  - Each of the above operations automatically eliminates duplicates
- To retain all duplicates, use the corresponding multiset versions **union all**, **intersect all** and **except all**.
- Suppose a tuple occurs m times in r and n times in s, then, it occurs:
  - m + n times in r **union all** s
  - min(m,n) times in r **intersect all** s
  - max(0, m – n) times in r **except all** s
- Find courses that ran in Fall 2009 or in Spring 2010

(**select** course_id **from** section **where** sem = ‘Fall’ **and** year = 2009)

union

(**select** course_id **from** section **where** sem = ‘Spring’ **and** year = 2010)

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.
- The result of any arithmetic expression involving null is null
  - Example: 5 + null returns null
- The predicate **is null** can be used to check for null values.
  - Example: Find all instructors whose salary is null.

```
select name
from instructor
where salary is null
```

Aggregate Functions

- These functions operate on the multiset of values of a column of a relation, and return a value
  - Avg, max, min, count, sum
- Find the average salary of instructors in the Computer Science department

```
select avg (salary)
from instructor
where dept_name= 'Comp. Sci.';
```

- Find the total number of instructors who teach a course in the Spring 2010 semester

```
select count (distinct ID)
from teaches
where semester = 'Spring' and year = 2010
```

Aggregate Functions – Group By

- Find the average salary of instructors in each department

select dept_name, **avg** (salary)

from instructor

group by dept_name;

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

dept_name	*avg_salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

Aggregate Functions – Having Clause

- Find the names and average salaries of all departments whose average salary is greater than 42000

```
select dept_name, avg (salary)
from instructor
group by dept_name
having avg (salary) > 42000 ;
```

Note: predicates in the having clause are applied after the formation of groups whereas predicates in the where clause are applied before forming groups

Nested Subqueries

- SQL provides a mechanism for the nesting of subqueries.
- A subquery is a select-from-where expression that is nested within another query.
- A common use of subqueries is to perform tests for set membership, set comparisons, and set cardinality.

Example Query

- Find courses offered in Fall 2009 and in Spring 2010

```
select distinct course_id
from section
where semester = 'Fall' and year= 2009 and
course_id in (select course_id
from section
where semester = 'Spring' and year= 2010);
```

Stored Procedures

- SQL statements which can be stored.
- A procedure can be thought of as a function or a method.
- if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.
  - CREATE PROCEDURE *procedure_name*
    AS
    sql_statement
    GO;
  - EXEC *procedure_name*;

Cursors

- Temporary memory or work area is allocated for operations to be used later.
  - DECLARE cursor_name CURSOR FOR select_statement;
  - OPEN cursor_name;
  - FETCH NEXT FROM cursor_name;
  - CLOSE cursor_name;
  - DEALLOCATE cursor_name;