little bit
COM

Structural Query language

DBMS - database management system

→ Database is a collection of related data.
↳ facts and figures that can be processed to produce information

→ Database management system stores data in such a way that it becomes easier to retrive, manipulate and produce information.

→ characteristics of database:
↳ Real world entity
↳ Relation based table
↳ Isolation of data
↳ less redundancy
↳ consistency

→ ACID properties :-
↳ Atomic transaction :- series of transaction such that either all occur or nothing occurs
↳ Consistency :- database is consistent after transaction
↳ Isolation
↳ durability :- ~~Commit~~ committed transaction survive permanently

~~→ Tier architecture~~

→ Tier architecture
↳ Tier 1 → work done directly on dbms.
↳ Tier 2 → work done through application / language
↳ Tier 3 → work done through application

→ Relational model → data is stored in form of table and data is related.
→ each ~~row~~ row contain unique value
→ each column contain values from same domain

database schema → Structure of database

Structural Query language

relational algebra + tuple relational calculus

→ Data definition language (DDL), Data manipulation language (DML)
Data control language (DCL), Transaction control (TC)
↳ authentication, permission

→ Key → set of attribute which help you identify row
↳ Super key → identify row in table
↳ candidate key → unique, not null
↳ primary key → one key from candidate key
↳ alternate key → all other candidate key except primary key
↳ foreign key → column that create relationship between two table.

Relationship → association between entity.

Relational data model → In relational database model data and relations between them are stored into table
↳ allow definition of data structure, storage and ~~data~~ retrieval operations.
→ query languages :-
↳ Relational algebra
↳ Relational calculus

→ why normalization → to reduce redundancy

→ Anomalies → update, deletion and insert

Normal forms →

↳ first → ~~every~~ values of every attribute should be indivisible
↳ second → should be in 1st and every non key element should depend on primary key.
→ third → should be in 2nd, no non key attribute should ~~be~~ be transitively dependent of primary key

↳ Boyce-codel → if $x \to A$ then x must be super key

→ equivlant schedules ↳ result equivalence
↳ view "
↳ conflict "

→ State of Transaction ↳ Active
↳ partially committed
↳ Failed
↳ Aborted
↳ committed

→ Deadlock → two ~~task~~ task are waiting for each other to finish but none is willing to ~~give~~ give resources to other
↳ wait die scheme
↳ wound wait scheme

→ deadlock avoidance
↳ wait-for graph

→ Lock protocol
↳ simplestic
↳ pre claiming
↳ Two phase locking
↳ Strict two phase locking

Structural Query language

relational algebra + tupled relational calculus

Create database name → create database
Drop database name → delete database
use database name → use database

Table

```
Create table name (
 id Int not null auto-increment,
 first name varchar (255),
 last name varchar (255),
 email varchar (255),
 primary key (id)
);
```

Insert into name (first name, last name, email)
Values ("Thon", "smith", "abc@email.com");

Update name set name = "abc" where id = 2

delete from name where id = 3

Alter table name add name varchar (255)
modify name & int (11)
drop name

Show table

select * from name; (* → all)
select first name from name;
select * from name where id = 2
select * from name order by id asc/desc;

select distinct from name ; (different entries)
select * from name where first name like '%n'
↳ ends with n
'%n%'
↳ n in middle
'n%'
↳ start with n
Select * from customers where state in ('x', 'y', 'z')
↳ name of states

Relationship

Table product id, customer id foreign keys → product id
customer id
Foreign key (customer id) refrences customer (id) ;

Aggrigate

Select Avg (age) from customers ;
" max(age) " " ;
" sum(age) " " ;)
" count(age) " " ;