Normalization

Transitive: if $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

1st normal form: a table should not contain any multi valued attribute.

RollNo	Name	Course
1	tab	c/C++
2	ash	Java
3	noor	DBMS

$\xrightarrow{1NF}$

RollNo	Name	Course
1	tab	c
1	tab	C++
2	ash	java
3	noor	DBMS.

∴ primary key is composite that means {RollNo, Course}

2nd normal form: table must be in 1NF and all non-key attributes should be fully functional dependent, hence no partial dependencies.

CustomerID	StoreID	Location
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Banglore
4	3	Mumbai

$\xrightarrow{2NF}$

CustomerID	StoreID
1	1
1	3
2	1
3	2
4	3

StoreID	Loca
1	Delhi
2	Bangl
3	Mumba

* these are two different table as now each attribute is dependent on its candidate key for each table, because before it was dependent on part of key.

3rd normal form: table must be in 2NF and there should be no transitive dependency.

RollNo	state	city
1	punjab	mohali
2	haryana	ambala
3	punjab	mohali
4	haryana	ambala
5	bihar	patna.

* primary key is {RollNo}
* FD1 : RollNo $\rightarrow$ state
  FD2 : state $\rightarrow$ city (determined by non-key)

$\downarrow$ 3NF

RollNo	State
1	punjab
2	haryana
3	punjab
4	haryana
5	bihar

State	City
punjab	mohali
haryana	ambala
bihar	patna

BCNF: table should be in 3NF and every right hand side attribute of the functional dependency should depend on the super key of that table.

4NF: it should be in BCNF and should not have any multi-valued dependency.

5NF: it should be in 4NF and cannot be further nonloss decomposed.

* if an attribute is fully as well as partially dependent then also in 2NF a new table is created for partial dependency.

MAPPING CARDINALITIES

1. an entity in A is associated with atmost one entity in B. is called one to one.

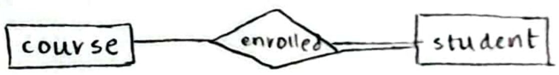
2. an entity A is associated with (0 or more) entities of B, and an entity in B is associated with atmost one entity in A. The PK of 1 relation becomes foreign key of many so merge the many side in 1 to M relationship.

3. an entity in A is associated with atmost one entity in B, and an entity in B can be associated with (0 or more) entities in A. merge the many side.

4. an entity in A is associated with any number of entities in B and vice versa. a new table of a relationship is created that has PK of both the tables.

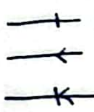
PARTICIPATION CONSTRAINTS

- in total participation every entity in E participates in atleast one relationship in R.
- in partial participation some entity in E participate in relationship R.



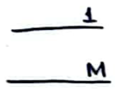
- course is partial because a course may or may not have students enrolled. but if a student exists then each student is expected to be enrolled in atleast one course.

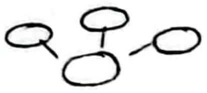
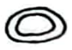
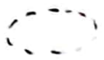
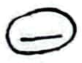
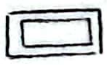
- crows foot



one
many
one or many

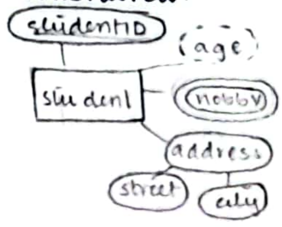
chen model



- composite attribute 
- multi-valued 
- derived 
- key 
- weak entity 

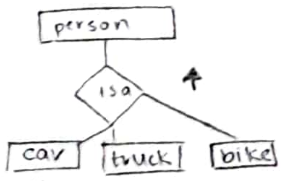
ER-DIAGRAM TO RELATION

- entity type becomes a table.
- all single valued attributes become a column.
- key attribute represented by a primary key.
- multi-valued attribute represented by a separate table. it has the primary key of original table and the name of attribute.
- composite attribute represented by components.
- derived attributes are not considered.



ENHANCED ER-DIAGRAM

- superclass can be further divided into subclasses.
- generalization is a bottom up design where two or more lower level entities are combined.
- many subclass combine to form a superclass.



- specialization is a top down approach where set of sub classes are defined.
- if an entity belongs to one subclass only then it is disjoint
- if an entity can belong to more than one subclass it is overlapping.
- total specialization is when every entity that belongs to a superclass must belong to atleast one subclass.

EER DIAGRAM TO RELATION

option 1, any kind of specialization
a superclass is specialized into subclasses. create relation and place attributes for both classes and add primary key of superclass in each subclass.

person(PID, name, age)
student(PID, GPA)
employee(PID, salary)

option 2, total specialization
we do not create superclass relation, but for each subclass we place its corresponding attributes and attributes of superclass as well.

student(PID, GPA, name, age)
employee(PID, salary, name, age)

option 3: disjoint
create only one relation and place all attribute of superclas and subclass in that relation also add a new attribute that differentiate between different identity.

person(PID, salary, name, age, GPA, persontype)

option 4: overlap & disjoint
create only one relation and place all attribute of superclass and subclass in that relation and add new attributes that is a boolean attribute

person(PID, salary, name, age, GPA, isStudent, isEmployee)

most suitable;
- if sub entities have more attribute and EER is totally specialized then option 2
- if sub entities have more attribute and EER is partially specialized then option 1
- if sub entities have fewer attribute and EER is disjoint then option 3
- if sub entities have fewer attribute and EER is overlap then option 4

AND [illegible]

RELATIONAL ALGEBRA

1. project (π)
   - selects columns from table
     $\pi_{attribute}$ (tablename)
   - displays distinct values.

2. selection (σ)
   - choose a subset of tuples from a relation
     $\sigma_{condition}$ (tablename)

3. cross product.
   columns = M+N (add columns)
   rows = M*N (mul tuples)
   relation1 x relation 2

4. set difference
   - result of A-B which includes all tuples that are in A, but not in B
   - attribute name should match.

5. union
   - domain of every column should be same.
   - contains all tuples that are either in R or S or both.

4. division
   - contains keyword ALL, AT ALL, EVERY

SQL SYNTAX

1. create table Person
   (ID int NOT NULL,
   name varchar(255))

2. insert into
   (ID, name)
   values (22, 'tabidah'),
   (21, 'maryam');

3. adding column
   alter table person
   add email varchar(255)

4. dropping a column.
   alter table persons
   drop column email;

5. rename a column
   alter table person
   rename column name to fname

6. alter datatype.
   alter table person
   modify column ID varchar(255)

7. adding a PK constraint
   alter table person
   add constraint PK (email)

8. adding check constraint.
   alter table person
   add constraint check_pos CHECK
   (ID > 0)

9. alter table person
   drop constraint constraint_name.

10. rename table name
    alter table person ~~new~~
    rename person_new.

11. add column after specific
    alter table person
    add age int
    after name;

12. update person SET age = 10
    WHERE ID = 1

POPULATION

$(\frac{\sigma}{}) < \mu < \bar{x} + Z_{\frac{\alpha}{2}} \left(\frac{\sigma}{\sqrt{n}}\right)$

EM VS DBMS

MS has fast access as it requires writing SQL query.

2. DBMS doesn't require any attributes for accessing data

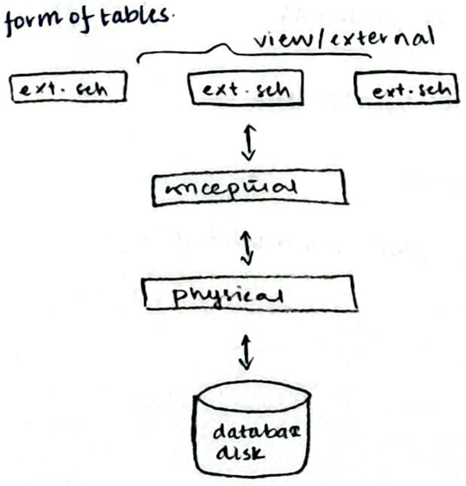
3. multiple users can access data at same time so concurrent access.

4. in DBMS we have role-based security and depends upon who is accessing data.

5. stores unique data meaning same data can't be stored in more than one place

DATA ABSTRACTION

- between user and database, there are three levels.
  1. view level / external
     → describes part of database that a usergroup is interested in e.g for a university management system, a student, faculty, dean will have different views and access.
  2. conceptual / logical level
     → tells what data is stored and hides physical storage structures. } hidden from users
     → entities, datatype, relationships, constraints.
  3. physical / internal level
     → how data is stored and shows physical storage structure. } hidden from users
     → access path for database
- in database (disk) information is stored in files, but the users see it in the form of tables.



INTEGRITY CONSTRAINTS

1. domain
   → should be in its domain of attribute. example, in age column negative values or letters cannot be entered so we can put a check.
2. entity
   → primary key should be unique so no null values or duplicate values are allowed.
3. referential
   → cannot delete record from parent table if records exist in child table.
   → cannot change primary key in parent table if child record exists.
4. key
   → has atleast one unique attribute, can have multiple as well. unique refers to candidate key.

KEYS

- key is an attribute and used to uniquely identify two tuples in the table.

1. candidate key.
   → set of attributes that uniquely identifies example, rollno, emailid, phonenumber, cnic.
   → out of all this only one is primary key and remaining are called alternative keys
   → is a subset of superkey.
2. primary key.
   → should be unique and not null.
   → is a superkey and candidate key.
3. foreign key.
   → an attribute or set of attribute that references to PK of same table or other table.
   → a table can have more than one foreign key.
   → cascade: parent record is deleted, so is child record.
4. super key
   → combination of all possible attributes which can be uniquely identified.
   → its values can be null.
   → not all superkeys are candidate keys
   → maximum super keys are $2^n - 1$.
5. surrogate key
   → a column that serves as an artificial PK

ACID PROPERTIES.

1. atomicity
   → the entire transaction takes place at once or doesnt happen at all.
2. consistency
   → database consistent before and after every transaction.
3. isolation
   → multiple transactions can happen but without interference.
4. durability
   → the changes are permanent in database even if system failure occurs.