Lecture 7:-

Relational Model

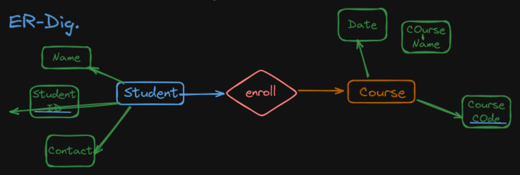
Relational diagram are the representation of data in tabular form

-> In this, table (relation) are the entity sets
-> Having attributes as columns
-> And each row (tuple) represents the entity

#degree of table -> is the no. of attributes in that table

Example of Relational diagram

ER-Dig.



Relational-Dig.

Relation -> Student

Name	Student ID	Contact
Raj	01	99xyz
Kunal	02	988yz
Ravi	03	212xyz
Sonu	04	9320

Relation -> Course

Course	Course ID	Date
Raj	BT-01	---
Kunal	BT-02	---
Ravi	BT-03	---
Sonu	BT-04	---

*Properties of relational model ->

i) Each entity set is a table with unique name
ii) attributes are atomic i.e. cannot broken into further
iii) Each attribute is unique name
iv) Each tuple is unique to no redundancy
v) tuple and attribute can be has no significance
vi) Table must follow constraints -> to maintain data consistency

*Keys in Relational model ->

1) Super Key (SK) -> the set of attributes which unique identify the entity are define as super key.

Ex. -> Student -> {name, studentID}, {studentID, phone}, {name, studentID,phone} etc.

2) Candidate key -> Are derived from super key without redundant attribute are define as Candidate key.

Ex. -> Student -> {studentID}, {studentID, phone} etc. (name is redundant)

3) Primary key (PK) -> Are derived from Candidate key which has least attributes is define as Primary Key.

Ex. -> Student -> {studentID}.

4) Alternate key (AK) -> The candidate key after removal of Primary key is define as Alternate Key.

Ex. -> Student -> {studentID, phone}.

5) Foreign key (FK) -> The primary key of a table (relation) which is used as attribute in another table (relation) is define as Foreign Key.

Ex. -> Student -> {studentID} used in course.

-> Generally, foreign key is used to define relationship between tables
-> The primary key used as attribute in a table is known as child table
-> and the pk of that table used is known as parent table

Example -

Name	Student ID	Contact
Raj	01	99xyz
Kunal	02	988yz
Ravi	03	212xyz
Sonu	04	9320

Foreign KEy

Course	Course ID	Date	Student ID
Raj	BT-01	---	01
Kunal	BT-02	---	02
Ravi	BT-03	---	03
Sonu	BT-04	---	04

5) Composite key -> The primary key of a table having atleast two attributes is define as Composite Key.

Ex. -> Student -> {studentID, Phone }.

5) Compound key -> The primary key of a table formed by two Foreign key.

6) Surrogate Key -> Is synthetic primary key created by DB itself.

ex. -> WE need to merge to table having same primary key say 'ID' so it may create inconsistency so to resolve it Surrogate key is beneficial.

Example - School A

Name	Student ID	Contact
Raj	01	99xyz
Kunal	02	988yz
Ravi	03	212xyz
Sonu	04	9320

School B

Name	Student ID	Contact
Ramu	01	2323
Rishi	02	4353
Akshay	03	46463
Sanvi	04	9676

A and B Both

Surrogate Key	Name	Student ID	Contact
1	Raj	01	99xyz
2	Kunal	02	988yz
3	Ravi	03	212xyz
4	Sonu	04	9320
5	Ramu	01	2323
6	Rishi	02	4353
7	Akshay	03	46463
8	Sanvi	04	9676

** INTEGRITY CONSTRAINTS ->

I) CRUD constraint -> Create Read Undo Delete must done with some integrity policy provided by DB

II) Domain constraint -> Defines Data type of each attribute and some condition of the attribute (ex.-> age >= 18)

III) Entity constraint -> Every entity must has Primary key

*** IV) Refrential constraint ->

1) Insert constraint -> we cannot insert in child table if corresponding value is not present in parent table.

2) Delete constraint -> we cannot delete from parent table if corresponding value is present in child table

=> On delete cascade

We can delete from parent table along with deleting the corresponding entity from child table too

**** # Can foreign key be NULL?

YES.....

=> On delete NULL -> Set FK to NULL of corresponding entity we delete from parent table

IV) KEY constraint ->

i) Not Null -> By default an attribute can be NULL so to avoid it we use Not NULL

```
Example -> create table customer ( Id int Not NULL,
           Name varchar (50) Not NULL,
           Age int);
```

ii) Unique -> Ensure all value of an attribute are unique,
-> Unique can be more than one attribute but PK must be one

```
Example -> create table customer ( Id int Not NULL,
           Name varchar (50) Not NULL,
           Age int,
           UNIQUE (ID)
           );
```

iii) Default constraints -> set default value of attribute

```
Example -> create table customer ( Id int Not NULL,
Name varchar (50) Not NULL, Prime_status int DEFAULT 0
           );
```

iv) Check constraint -> Used to limit value range

```
Example -> create table customer (
           Age int
           );
```

v) Primary Key Constriant -> PK != Null
1 Relation only 1 PK

```
Example -> create table customer (
           ID int Not Null
           PRIMARY KEY (ID)
           );
```

vi) Foreign Key Constraint -> Defines relation b/w 2 table

```
Example -> create table order (
           order ID Not NULL
FOREIGN KEY (CUSTOMER_ID) refering (CUSTOMER_ID)
           );
```

Lecture 8:-

ER Model - Relational Model

1) Strong entity -> Becomes the relation (table) of Relational model

Example -

Loan

PK

Loan ID	Loan amount
01	--
02	--
03	--
04	--

2) Weak entity -> Becomes the relation (table) of Relational model

-> Having PK of the corresponding entity as FK (to establish relationship)

Example - Payment

PK

+

FK

Loan ID	Payment ID	Payment amount
01	11	--
02	22	--
03	33	--
04	44	--

3) Single value attribute -> Added simply as attribute.

4) Composite attribute -> Added each as simple attribute of the relation (table)

Example - Customer

PK

CS_ID	First Name	Last Name	DOB
01	--	--	--
02	--	--	--
03	--	--	--
04	--	--	--

5) Multivalue attribute -> Creates a table of that attribute having PK of the corresponding table (table which has composite attribute) as FK and the attribute.

Example - Customer -> Contact No.

Contact No.

FK

PK

CS_ID	Contact
01	9933
01	8899
02	5566
03	9134

6) Derived attribute -> Not added cause we can derive them using API when needed

7) Generalization -> can be implement in two ways -

Way 1 -> Create table of each entity (parent + child)

Eg -> Account, Saving account, Current Account

Account

PK

Acc_No	Balance
A11	--
A12	--
C11	--
C12	--

Saving Acc

PK

Acc_No	Other att.
A11	--
A12	--

Current Acc

PK

Acc_No	Other att.
C11	--
C12	--

Way 2 -> Create table of only child tables and share the common attribute in both

Eg -> Saving account, Current Account

:- Issue in Way 2-> They share same attribute so if the value of that common attribute may vary in both table which can create redundancy

Example->

Saving Acc

PK

Acc_No	Other att.	Balance
A11	--	--
A12	--	--
A13	--	--
A14	--	--

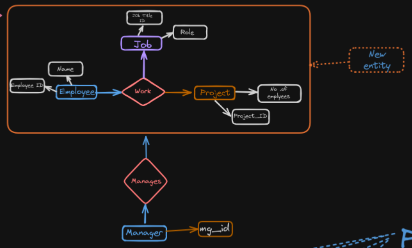
Current Acc

PK

Acc_No	Other att.	Balance
C11	--	--
C12	--	--
C13	--	--
C14	--	--

8) Aggregation -> By creating table of that relation and the table will have PK of all that attributes

Ex. ->



New entity

Then,

PK

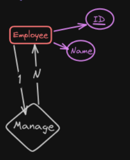
Manages

Employee ID	Job_title_id	Project_ID	mg_ID
A11	--	--	1
A12	--	--	1
A13	--	--	1
A14	--	--	1

9) Unary relation -> Can be of three types->

1) 1 : N -> Having PK as FK in same table

Example ->



PK

FK

Employee_ID	Name	Mg_ID
A11	--	A12
A12	--	A12
A13	--	A12
A14	--	A12

2) 1 : 1 -> Having PK as FK in same table
(and each tuple having corresponding value in FK)

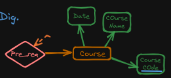
PK

FK

Person_ID	Name	FeoncieID
A11	--	A12
A12	--	A11
A13	--	A14
A14	--	A13

3) M : N -> Implement by creating new table of that relationship
consisting the PKs as FKs.

Example-> ER-Dig.



PK

Course

Course	Course ID	Date
maths	BT-01	---
physics	BT-02	---
commerce	BT-03	---
quantum	BT-04	---

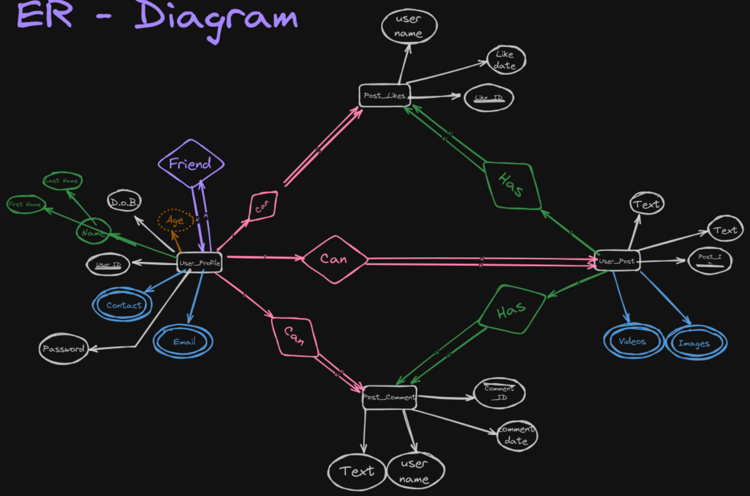
FK1 FK2

Pre-rea

Course_ID	PreRea_ID
BT-01	Null
BT-03	BT-01
BT-04	BT-01
BT-04	BT-02

ER -> to -> Relation of FACEBOOK

ER - Diagram



Relation->

1) User_Profile -> User_id, first_name, last_name, DOB, pass

2) contact -> User_id (fk), contact_no

3) Email -> User_id (fk), email

4) Friend (M:N)-> send_userID (fk), recieve_userID (fk)

5) Post_like -> User_ID (FK), post_id (FK), date_like, like_id (PK)

6) User_post -> User_ID (FK), postID (PK), date_post, text

7) Image -> postID (FK), Images
8) Video -> Post_ID (FK), Videos

9) Post_comment -> PostID (FK), userID (FK), date_comment,
text, comment_ID (PK)