Unit-7

Relational Database Design

Ankit Pangeni

DATE

* Relational Database Design Using ER-to-relational mapping

ER diagrams can be mapped into relational schema, ie. it is possible to create relational schema using ER diagram. We cannot import all the ER constraints into relational model, but an approximate schema can be generated. After designing the ER diagram of the system, we need to convert it into relational models which can directly be implemented by any RDBMs like oracle, MySQL, etc.

To reduce the ER diagram into tables, normally we divide ER diagram into following section

1. Mapping strong entity sets to ER
2. Mapping weak entity sets to ER
3. Mapping relation sets to ER
   - i) Mapping of Binary 1:1 relation types to ER
   - ii) Mapping of Binary 1:N relation types to ER
   - iii) Mapping of Binary M:N relation types to ER
4. Mapping of multivalued attributes to ER
5. Mapping composite attributes to ER.
6. Mapping of N-ary relationship types to ER
7. Mapping specialization / generalization to ER
8. Mapping aggregation to ER.

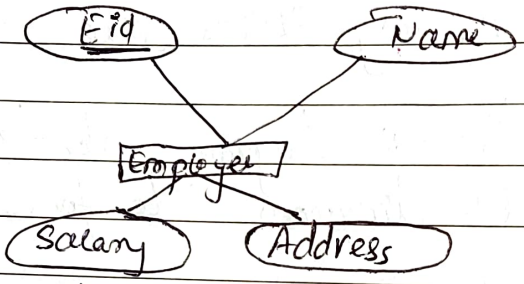
PAGE

DATE

1. Mapping Strong entity sets to ER:-

→ Create table for each of the strong entity
→ Entity's attributes should become fields of table with their respective datatypes
→ Declare key attribute of strong entity set as primary key of the table.

Ex: Let's take a strong entity set Employee with four attributes:



We create a table "Employee" with fields Eid, Name, Salary, Address and make Eid the primary key

Employee

Eid	Name	Salary	Address

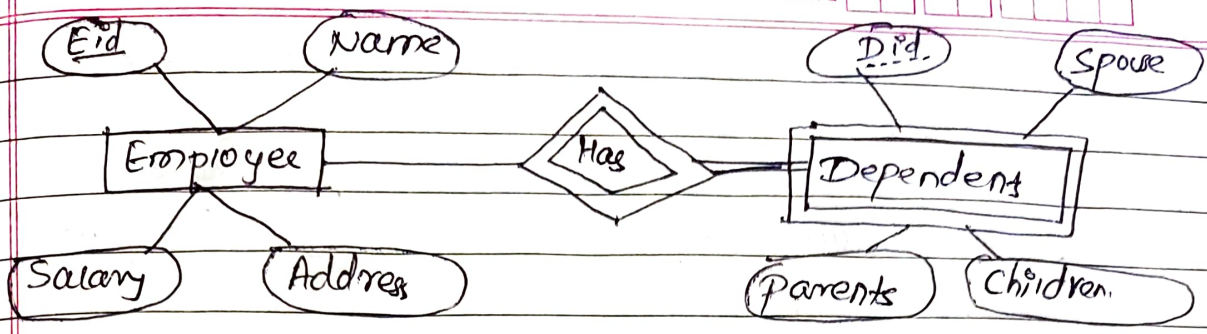
2. Mapping Weak entity sets to ER

A weak entity set doesn't have its own primary key & always participates in one-to-many relationship with owner entity set & has total participation.

→ Create table for weak entity set.
→ Add all its attributes to table as field
→ Add primary key of identifying entity set
→ Declare all foreign key constraints

classmate

PAGE



To draw table of weak entity set 'Dependents' we simply set all their attributes & also set primary key of Employee table to the Dependents table.

Employee

Eid	Name	Salary	Address

Dependents

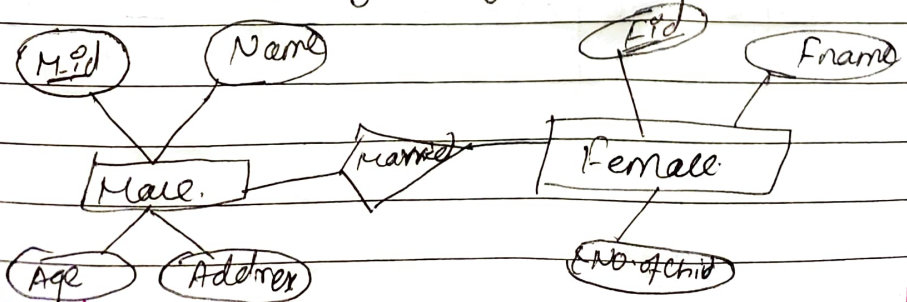
Eid	Did	Parents	Spouse	Children

3. Mapping Relation sets to ER

To construct tables from given relationships we may have following types of relationships.

1) Mapping of Binary 1:1 relation types to ER

For constructing table from a binary one to one relationship, we set primary key of any one of the entity set as foreign key.



DATE

Here, we can set m.id as foreign key to Female table or f.id to Male table as their foreign key

Male

Mid	Name	Age	Address

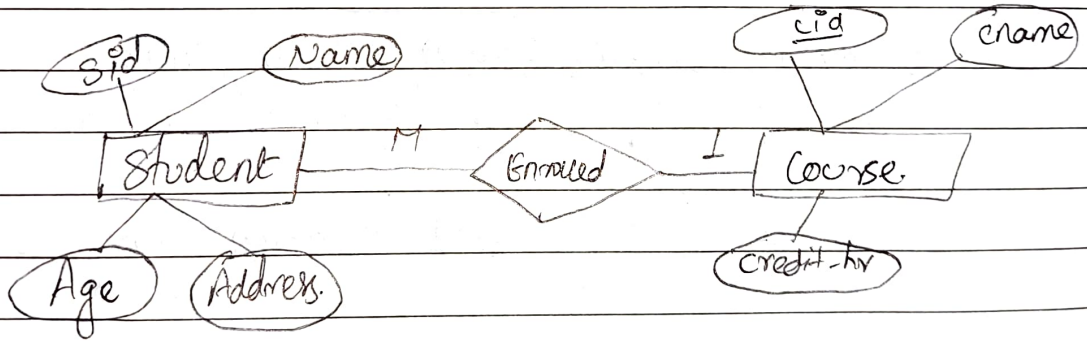
Female

Fid	Mid	Fname	Mother Name

pp) Mapping of Binary 1:N relationship types to ER

For binary one-to-many relationship identify the relation that represent the participating entity type at the N-side of the relationship type & then include primary key of one side entity set into many side entity set as foreign key. Separate relation is created for the relationship set only when the relationship set has its own attributes.

Eg:



Student

Sid	Name	Age	Address	Cid

Course

Cid	Cname	Credit_hr

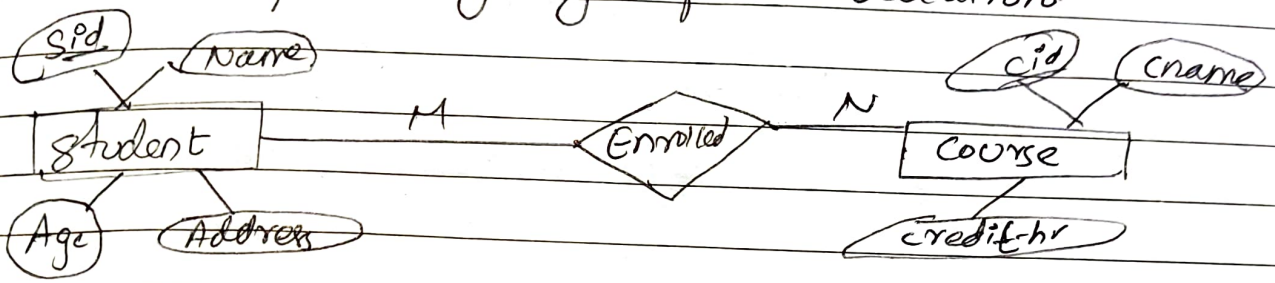
PAGE

classmate

DATE

iii) Mapping of Binary M:N relationship types to ER.

For a binary Many-to-Many relationship type, separate relation is created for the relationship type. primary key for each participating entity set is included as foreign key in the relation & their combination will form the primary key of the relation.



Here, we create new table Enrolled that contains the primary keys of entities Student & Course.

Student

Sid	Name	Age	Address

Course

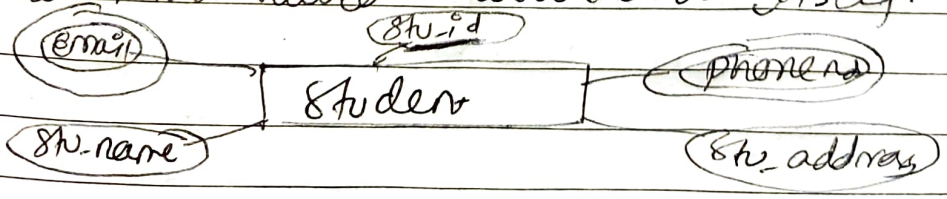
Cid	Cname	credit-hr

Enrolled

Sid	Cid

4. Mapping of multivalued attributes to ER.

If an entity has multivalued attribute, separate relation is created with primary key of entity set and multivalued attribute itself.



Student

Stu_id	Stu_name	Stu_address

Email

Stu_id	Email

Phone

Stu_id	Phones

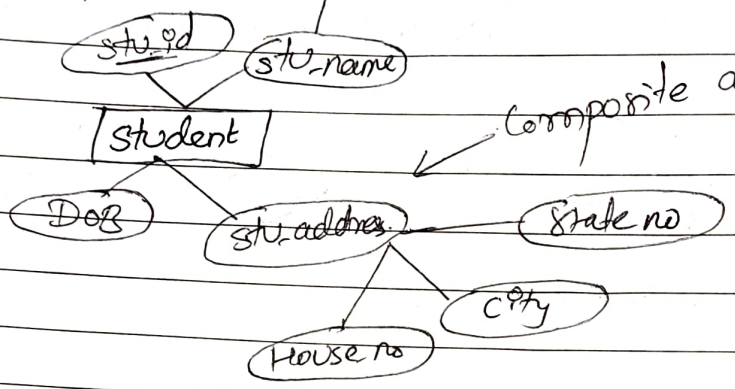
classmate

PAGE

DATE

5. Mapping Composite attributes to ER

If an entity has composite attributes, no separate (column) is created for composite attribute itself. Rather attributes (columns) are created for component attributes of the composite attribute

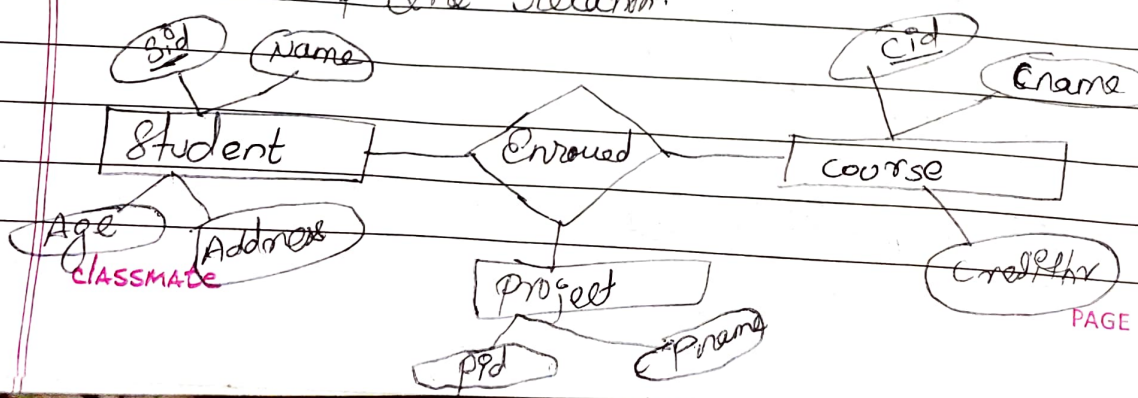


Student

stu_id	stu_name	DOB	House no	City	State_no

6. Mapping of N-ary relationship types to ER

For each n-ary relationship set for $n > 2$, a new relation is created. Primary keys of all participating entity sets are included in the relation as foreign key attributes. Besides this, all simple attributes of the n-ary relationship set (or simple components of composite attributes) are included as attributes of the relation.



CLASSMATE

PAGE

DATE

Student

Sid	Name	Age	Address

Course

Cid	Cname	creditHr

Project

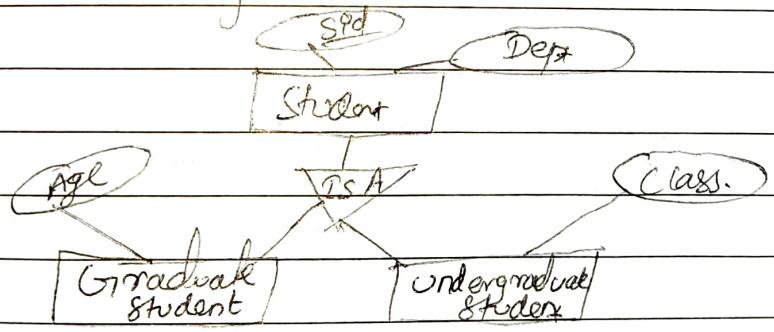
Pid	Pname

Enrolled

Sid	Cid	Pid

7. Mapping specialization/generalization to ER.

To construct relational tables from given ER diagram with specialization/generalization, we set primary key of the supper class to their subclass as their foreign key. If subclasses are disjoint & complete, then relation for a subclass entity set includes all attributes of superclass entity set & all of its own attributes.



Here, we set Sid to both Grad. & UnGrad Student

Student

Sid	Dept

Graduate student

Sid	Age

undergraduate student

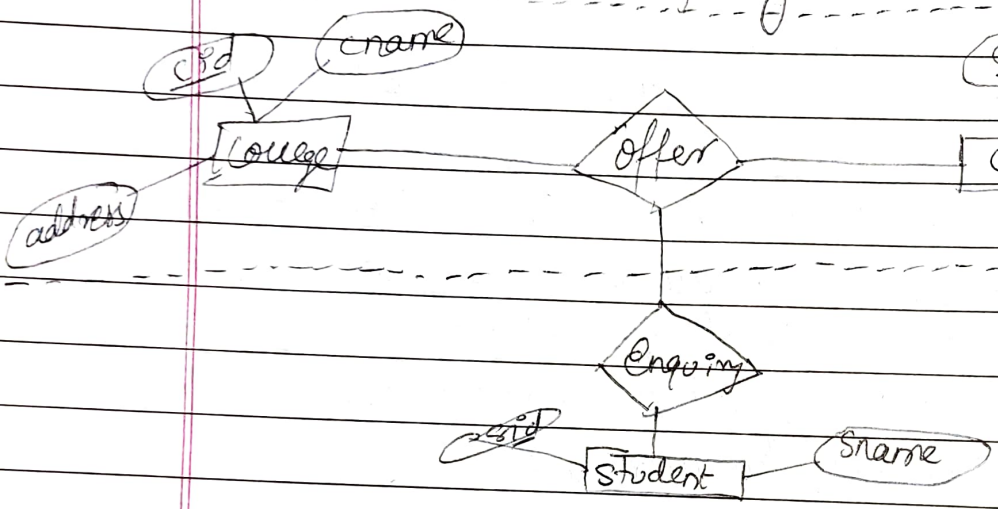
Sid	class

PAGE

DATE

8. Mapping Aggregation to ER:

In this there is no distinction between entity sets and relationship sets in relational model. In relational model separate relation is created for this relationship and the relation contains primary key of associated entity set & the relationship set & its own attributes, if any.



College

Cid	Cname	address

Course

Co_id	Co_name

Enquiry

Sid	Sname	Cid	Coid

DATE

* Informal Design Guidelines for Relational Schemas

Informal guidelines may be used as measures to determine the quality of relation schema design are listed below:-

→ Making sure that the semantics of the attributes is clear in the schema.
→ Reducing the redundant information in tuples.
→ Reducing the null values in tuples
→ Disallowing the possibility of generating invalid (spurious) tuples

1) Imparting clear Semantics to attributes in relation

Whenever we are going to form relational schema, there should be some meaning among the attributes. This meaning is called Semantics. This Semantics relates one attribute to another with some relation.

Guideline:-

→ Informally each tuple in a relation should represent one entity or relationship instance.
→ Attributes of different entities shouldn't be mixed in same relation.
→ Only foreign keys should be used to refer other entities.
→ Entity & relationship should be kept apart as much as possible.

DATE

2) Reducing the redundant information in tuples and update anomalies

If a table contains attributes of multiple entities it may cause redundant information. It wastes storage. Problems with update anomalies.

- Insertion anomalies
- Deletion anomalies
- Modification anomalies

We can reduce redundant information by using Normalization. It is the process of reducing a single table into a multiple simple tables

Ex: Let's take two table.

Student

Sid	Sname	Semester

Department

D_no	D_name	Address

If we integrate these two relations into a single table

Sid	Sname	Semester	D_no	D_name	Address.

→ Here if we insert the tuples, there may be N students in one department, so D_no, D_name, & Address are repeated N times, which causes data redundancy

→ Another problem is update anomalies, ie. if we insert new department that has no students

→ If we delete last student of a department, then whole information about the department will be deleted

→ If we change value of one attributes of a particular table. Then we must update the tuples of all the students belonging to that department else data will become inconsistent.

CLASSMATE

PAGE

Guideline:

DATE

So, design the base relation schemas so that no insertion, deletion or modification anomalies are present in the relations. If any anomalies are present, note them clearly & make sure that the programs that update the database will operate correctly.

3) Reducing null values in tuples

If many of the attributes do not apply to all tuples in the relation, we end up with many NULLs in those tuples. This can waste storage. Another problem with NULLs is when aggregate functions/operations such as COUNT or SUM are applied. SELECT & JOIN operations involve comparisons; if NULL values are present, then results may become unpredictable.

There are many reasons for nulls, such as

→ attribute not applicable or invalid
→ attribute value unknown
→ value known to exist, but unavailable.

So, to reduce null values in tuples, avoid placing attributes in a base relation whose values may frequently be NULL. If NULLs are unavoidable, make sure that they apply in exceptional cases only.

4) Disallowing the possibility of generating spurious tuples

The relations should be designed to satisfy the loseless join condition. The "Loseless" join property is used to guarantee meaningful results for join operations. No spurious or invalid tuples should be generated by doing a natural-join of any relation.

DATE

Guideline: Design relation schemas so that they can be joined with equality conditions on attributes that are appropriately related (primary key, foreign key) pairs in a way that guarantees no spurious tuples are generated.

Avoid relations that contain matching attributes that aren't (foreign key, primary key) combinations because joining such attributes may produce spurious tuples

* Functional Dependencies

Functional dependency is a relationship that exists when one attribute uniquely determines the other attribute.

If R is a relation with attributes X and Y, a functional dependency between the attributes is represented as $X \rightarrow Y$, which specifies Y is functionally dependent on X. Here, X is a determinant set and Y is a dependant attribute. For each value of X there is exactly one value of Y. And one value of Y can have multiple values of X.

Note: If $X \rightarrow Y$ in R, then this does not say whether or not $Y \rightarrow X$ in R.

DATE

Example:

Employee Table:

Employee_Id	Employee_Name	Employee-Department	Salary
1	Ryan	Engineering	50,000
2	Justin	Politics	40,000
3	Andrew	Management	30,000
4	Ankit	Human Resource	40,000

In this example, If we know the value of Employee_id we can obtain Employee Name, ~~department~~ Salary, etc. Hence, we can say, Employee-Name, department and Salary are functionally dependent on Employee_id.

Types of functional dependency

1. Trivial functional dependency.

The dependency of an attribute on a set of attributes is known as trivial if the set of attributes include that attribute. Symbolically, $A \rightarrow B$ is trivial functional dependency if B is a subset of A. $A \rightarrow A$ and $B \rightarrow B$ is also trivial. Ex: In above table {Employee_Id, Employee_Name} → Employee_id is trivial as Employee_Id is a subset of {Employee_id, Emp_Name}. That makes sense because if we know the values of ~~Student_Id~~ Employee_id & Employee_name then the value of Employee_id can be uniquely determined.

DATE

2. Non-trivial functional dependency

If a functional dependency $X \to Y$ holds true when Y is not a subset of X then this dependency is called ex, in above table. Employee_id → Employee_name is non-trivial as Employee_name is not a subset of Employee

3. Fully Functional dependency.

An attribute is fully functional dependent on another attribute, if it is functionally dependent on that attribute and not on any of its proper subset. Mathematically, for a relation schema R, $X \to Y$, then Y is said to be fully functional dependent on X if $Z \to Y$ is false for all $Z \subset X$.
Ex: $AB \to Q$ (true), $B \to Q$ (false) then Q is FFD on ABC.

4. Partial Functional Dependency.

An attribute is partial functional dependent on a second attribute iff it is functionally dependent on the 2nd attribute & also dependency occur on any subset of the 2nd attribute. Mathematically, if $X \to Y$, then Y is said to be partial functional dependent on X if by removal of some attributes from X and the dependency still holds
Ex In above table, {Emp_id, Emp_department} → {Emp_Name} is partial because Emp_id → Emp_Name also holds.

DATE

5. Transitive dependency.

A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies. Ex: $X \to Z$ is a transitive dependency if the following three functional dependencies hold true
$[X \to Y, Y \not\to X, Y \to Z]$ then $X \to Z$

Ex: in above table, Employee_Id → Employee_Name
Employee_Name → Employee_Department
Also, Employee_Id ⟶ Employee_Department.

6. Multivalued dependency.

It occurs when there are more than one independent multivalued attributes in a table.
Ex: Consider a bike manufacture company, which produces two colors Black & Red bikes in each model every year. The attributes are:
Bike_model, Manuf_year, Color.
Here, manuf_year and color are independent of each other & dependent on bike model. In this case, these two columns are said to be multivalued dependent on bike_model. They can be represented as
bike_model ⟶ manuf_year

PAGE

DATE

* Normal forms Based on Primary keys

→ Concept of normalization

Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in a relation may cause insertion, deletion and update anamalies so, it helps to minimize the redundancy in relations Normal forms are used to eliminate or reduce redundancy in database tables It divides larger tables to smaller tables and links them using relationships.

→ Normal forms

Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.

Basically, the normal form refers to the highest normal form condition that the relation meets, and hence indicates the degree to which it has been normalized.

DATE

→ Practical use of normal forms

Although several higher normal forms have been defined, database design as practiced in industry today pays particular attention to normalization only upto 3NF, BCNF, or at most 4NF. The database designers need not normalize to the highest possible normal form. Relations may be left in a lower normalization state, such as 2NF, for performance reasons

→ Advantages of Normalization

- Helps in maintaining data integrity
- Helps in simplifying the structure of table.
- Eliminates insertion, update & deletion anomalies
- Eliminates redundant data & less storage space is required
- Better understanding of data.

→ Denormalization

It is the process of storing the join of higher normal form relations as a base relation, which is in a lower normal form

→ Keys and attributes participating in keys

- Super key: It is an attribute or a set of attributes that is used to uniquely identify all attributes in a relation. All super keys can't be candidate keys but its reverse is true. ex: relation R(A,B,C,D,E,F) then functional dependencies AB→CDEF, CD→ABEF, CB→DF, D→B

classmate

AB → Super key, CD → Super key, CB → NO, D → No.

because we can identify rest of attributes CDEF using AB.

PAGE

DATE

- Candidate key: It is a set of attributes or a attribute which uniquely identify the tuples in a relation or table. As we know that primary key is a minimal super key, so there is one & only one primary key in any relation but there is more than one candidate key can take place candidate key's attributes can contain NULL value which oppose to the primary key. Ex: Student {ID, F_name, L_name, phone_no} Here we can see two candidate keys ID, and {F_name, L_name, phone_no}.

- Prime attribute: Attributes of the database tables which are candidate keys of the database tables are called prime attributes.
ex:

Registration_no	Rollno	Name	Class	Year

Prime attributes — non-prime attributes

- Non prime attribute: Attributes of the database tables which do not exist in any of the possible candidate keys of the database tables are called non-prime attributes. In above example, Name, Class, year are non-prime attributes.

DATE

Types of Normal Form based on Primary Key

The normal forms first, second and third are called primary key normal forms. Here each of the non-prime attributes of the relation depends upon the primary attribute of the relation.

1. First Normal Form

A relation is said to be in 1NF iff:

- There are no duplicate rows in the table.
- Domain of all attributes are atomic: ie. each cell is single valued.
- Data for a particular column are of similar kind
- There are no repeating group or arrays

Ex: Convert the following un-normalized relation to 1NF

Product ID	Color	Price
1	red, green	15.99
2	yellow	23.99
3	green	17.50
4	yellow, blue	9.99
5	red	29.99

It is not in 1NF because of multivalued attribute color

⇒ Now, let's convert it to 1NF

Product ID	Color	Price
1	red	15.99
1	Green	15.99
2	yellow	23.99
3	green	17.50
4	yellow	9.99
4	blue	9.99
5	red	29.99

DATE

Eg.2.

Sid	Sname	Phone	State	Age
1	Ram	9848372626	State1	23
		8901867812		
2	Hari	9851027182	State 2	33
3	Ramesh	9848372098	State 1	22
		9851103824		
4	Anand	9840094302	State 3	29

⇒ Now, convert given relation to 1NF as below

Student

Sid	Sname	State	Age
1	Ram	State 1	23
2	Hari	State 2	33
3	Ramesh	State 1	22
4	Anand	State 3	29

Phone_detail.

Sid	Phone
1	9848372626
1	8901867812
2	9851027182
3	9848372098
3	9851103824
4	9840094302

2. Second Normal form

A relation is said to be in 2NF iff.

- It is in 1NF
- No partial dependencies exist between non key attributes (non-prime) & key attributes

DATE

The main aim of 2NF is to ensure that all information in one relation is only about one thing.

Ex: Convert the following 1NF to 2NF

Teacher

Tid	Tsubject	Tage
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Candidate keys: {Tid, Tsubject}

Non-prime attribute: Tage.

It is in 1NF. It's not in 2NF because non-prime attribute Tage is dependant on Tid alone which is a proper subset of candidate key

:) Let's convert to 2NF

teachers_details

Tid	Tage
111	38
222	38
333	40

teachers_subject

Tid	Tsubject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

Ex: Convert the following 1NF to 2NF

Customer ID	Store ID	Purchase Location
1	1	Pokhara
1	3	Ktm
2	1	Pokhara
3	2	Butwal
4	3	Ktm

classmate

PAGE

It has a composite primary key (customer_ID, Store ID). Non-key attribute (Purchase location) only depends on Store ID. which is only part of the primary key. So, it doesn't satisfy 2NF

=> Let's convert to 2NF

Table_store

Store ID	Purchase location
1	Pokhara
2	Butwal
3	Ktm

Table_purchase

Customer ID	Store ID
1	1
1	3
2	1
3	2
4	3

3. Third Normal form (3NF)

A relation is said to be in 3NF iff

- It is in 2NF
- No transitive dependencies exist between non-key attributes & the key attributes i.e. no non key attribute is functionally dependent on other non-key attributes

Ex:

Eid	Dept	Depthead
1	System	Ram Thapa
2	Finance	Hari Sharma
3	Admin	Reja Shrestha
4	System	Ram Thapa
5	Sales	Ravi Tiwari

DATE

Convert the following 2NF to 3NF

⇒ Converting 2NF to 3NF by removing transitive dependen

R1

Eid	Dept
1	System
2	Finance
3	Admin
4	System
5	Sales

R2

Dept	Deptheau
System	Ram thapa
Finance	Hari sharma
Admin	Rita shrestha
Sales	Ravi tiwari

Ex. Convert the following relation to 3NF

Row no	Game	Fees structure
1	Basketball	500
2	Basketball	500
3	Basketball	500
4	Cricket	600
5	Cricket	600
6	Cricket	600
7	Tennis	400

⇒

Row no	Game
1	Basketball
2	Basketball
3	Basketball
4	Cricket
5	Cricket
6	cricket
7	Tennis

Game	Fees structure
Basketball	500
Cricket	600
Tennis	400

classmate

PAGE

DATE

4. Boyce-codd normal form (BCNF)

BCNF is an extended form of 3NF. So also called 3.5 NF or advanced 3NF
A relation is in BCNF iff
- It is in 3NF
- If every determinant is a candidate key
  ie. for each functional dependency ($X \to Y$), X should be a candidate key.

BCNF decomposition does not always satisfy dependency preserving property.

Ex: Convert the following 3NF to BCNF

Student	Teacher	Subject
Thansi	P.Naresh	Database
Thansi	K.Das	C
Subbu	P.Naresh	Database
Subbu	R.Prasad	C

⇒ Given functional dependencies:
F.D | (Student, Teacher) → Subject ~~(Student,~~
(Student, Subject) → Teacher
Teacher → Subject

Candidate keys are:- (Student, teacher) and (Student, Subject)

classmate

PAGE

DATE

Let's Convert R1 to BCNF

R1

Teacher	Subject
P.Naresh	Database
K.Das	C
R.Prasad	C

R2

Student	Teacher
Thansi	P.Naresh
Thansi	K.Das
Subbu	P.Naresh
Subbu	R.Prasad

Ex:2

Emp_id	Emp nationality	Emp_dept	dept_type	dept_no_of_emp
1001	Nepalese	Production	D002	200
1001	Nepalese	Store	D002	250
1002	Indian	Technical	D134	100
1002	Indian	Purchasing	D134	600

⇒ Given functional dependencies

Emp_id → emp_nationality

emp_dept → {dept_type, dept_no_of_emp}

Candidate key - {Emp_id, Emp_dept}

Let's convert the 3NF to BCNF

⇒ Emp_nationality table

Emp_id	Emp_nationality
1001	Nepalese
1002	Indian

emp_dept table mapping

Emp_id	Emp_dept
1001	Production
1001	Store
1002	Technical
1002	Purchasing

Emp_dept table

Emp_dept	dept_type	dept_no_of_emp
Production	D001	200
Store	D001	250
Technical	D134	100
Purchasing	D134	600

PAGE

DATE

5) Fourth Normal form (4NF)

A relation is in 4NF if
- It is in BCNF
- It contains no multivalued dependencies.

A table with a multivalued dependency violates the normalization standard of 4NF because it creates unnecessary redundancies and can contribute to inconsistent data. To bring this up to 4NF, it is necessary to break information into two tables

Ex: Convert the following BCNF to 4NF

Course	Teacher	Book
DBMS	Ram Thapa	DB concept
DBMS	Hari Sharma	Katson books
DBMS	Hari Sharma	DB Concept
DBMS	Ram Thapa	Katson Book

⇒ Here, Course →→ Teacher & Course →→ Books. So for certain value of Course we are having set of values for teacher & Books such that teacher & books are not related. So, it has multivalued dep

Course	Teacher
DBMS	Ram Thapa
DBMS	Hari Sharma

Course	Book
DBMS	DB concept
DBMS	Katson books

classmate

PAGE

DATE

* Properties of Relational Decomposition

→ Decomposition

Decomposition in DBMS is to break a relation into multiple relations to bring it into an appropriate normal form. It helps to ~~reduce~~ remove redundancy, inconsistencies, and anomalies from a database.

The decomposed relations when reconstructed should give the original relation accurately without loss of any information

→ Properties of Relational decomposition

1. Loseless Decomposition
2. Dependency Preservation
3. Lack of redundancy
4. Attribute preservation

• (Loseless Decomposition)

It means that there is no loss of information during decomposition. A decomposition of a relation R into relations R1, R2, R3 ---, RN is called loseless decomposition if the relation R can be reconstructed back using natural join

DATE

of relations R1, R2, R3, ---RN without any loss of information

It is to be noted that natural join is the way to recover the relation from the decomposed relations

Ex: Decomposition of relation R = {A, B, C}

R

A	B	C
1	P	U
2	Q	V
3	R	W

R_1

A	B
1	P
2	Q
3	R

$\Pi_{A,B}(R)$

R_2

A	C
1	U
2	V
3	W

$\Pi_{B,C}(R)$

The decomposition of R into R_1 & R_2 is lossless because $R = \Pi_{R1}(R) \bowtie \Pi_{R2}(R)$

DATE

- (Dependency Preservation)

It states that every dependency must be satisfied by at least one decomposed table.

If a relation R is decomposed into relation R_1 and R_2, then the dependencies of R either must be a part of R_1 or R_2 or must be derivable from the combination of functional dependencies of R_1 and R_2.

Ex: Suppose there is a relation R(A, B, C, D) with functional dependency set (A→BC). The relation R is decomposed into R_1(ABC) & R_2 (AD) which is dependency preserving because FD, A→BC is a part of relation R_1(ABC)

- (Lack of redundancy)

Redundancy is the problem of repetition of data in the database. The proper decomposition should not suffer from any data redundancy. The lack of data redundancy property may be achieved by normalization process

- (Attribute Preservation)

All the attributes that were in the original relation which is being decomposed, must be preserved in the resulting decomposed relations set. The attributes in R will appear in at least one relation schema R_i in the decomposition i.e no attribute is lost.

CLASSMATE PAGE

Questions asked from this chapter.

Q. Explain Boyce-codd normal form with example. How is it different from 3rd Normal form. (2078 - 5 marks) (2075 - 5 marks) (2072 - 5 marks / 2071 - 10 marks)

Q What is normal form? Explain their types. Explain about lose-less join decomposition (2076 - 10 marks)

Q Define functional dependencies Explain trivial and non trivial dependencies (2078 - 5 marks) (2074 - Short note, 2075 - Short note) (2073 - 5 marks)

Q Discuss lose-less decomposition & dependency preservation property of normalization (2075 - 5 marks)

Q. Discuss first, second & third normal forms with suitable example. (2069 - 5 marks) (2074 - 10 marks) (2070 - 5 marks)

Q. What is functional dependency? Describe full & partial dependency with suitable example. (2069 - 5 marks)

Q With the information given below, calculate any three members of F (2070 - 6 marks)

$R = (A, B, C, G, H, I)$

$F = \{ A \to B,\ A \to C,\ CG \to I,\ B \to H \}$