10/7/18

UNIT - 2

Relation:- Representing the data in a relational model is called a relation.

schema - structure of data

Relation schema:- It describes fields for the table, the schema specifies the relation name, name of each field, domain of each field

eg:- Student (SID : Int, SName: String, Branch : String)

Domain:- The domain of a database is the set of all allowable values (or) attributes of the database

eg:- Gender (Male, Female, unknown)

Degree:- The no.of fields (or) attributes in a relation is called degree of that relation.

Cardinality:- The no.of Tuples in a relation. is called cardinality.

ex:- Student

ID	Name	Address
1.	Chenchu	562
2.	Kummu	563
3.	Mani	593

Degree : 3
cardinality : 3

Entity:- An Entity is an Object which have some existance

* The existence is either physical (or) conceptual

eg:- person is an entity with physical existence

Job, course are entities with conceptual existence

Attribute:- The property of an entity is called attribute

eg:- Student is an entity with attributes SID, Sname, branch

Types of Attributes:-

1. Simple / Atomic attribute :- An attribute which is not divisible is called simple attribute.

ex:- Age of person Entity.

2. Composite Attribute:- An attribute which can be divisible into parts is called composite attribute.

ex:- Name & Address of person entity.

3. Single valued attribute:- An attribute having single value eg:- Age, DOB, ID

4. Multi valued attribute:- Attribute having multiple values eg:- phone Number, Email ID.

5. Stored Attribute:- An attribute which is used for deriving a new attribute.

eg:- DOB

6. Derived Attribute:- An attribute which is desiving from another attribute.

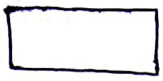
eg:- Age of person entity.

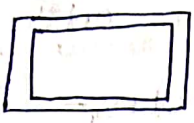
Relationship:- The association among the entity is called Relationship

Relationship Set:- The Collection of Relationships is called Relationship Set.

* A Relationship can be defined with an attribute is called descriptive Attribute.

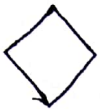
ER Diagrams:- The following symbols are used in database design.

 → Entity

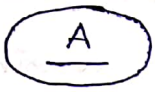
 → weak entity

 → Attribute

 → Multi value attribute

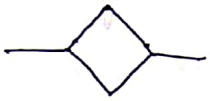
 → Relationship

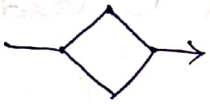
 → weak relationship

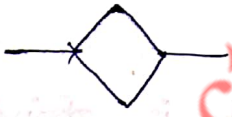
 → primary key of an attribute

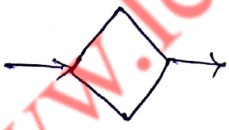
 → derived attribute

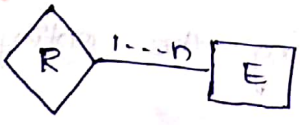
 → attribute of weak entity

 → Many to many

 → Many to one

 → one to many

 → one to one



→ cardinality limits

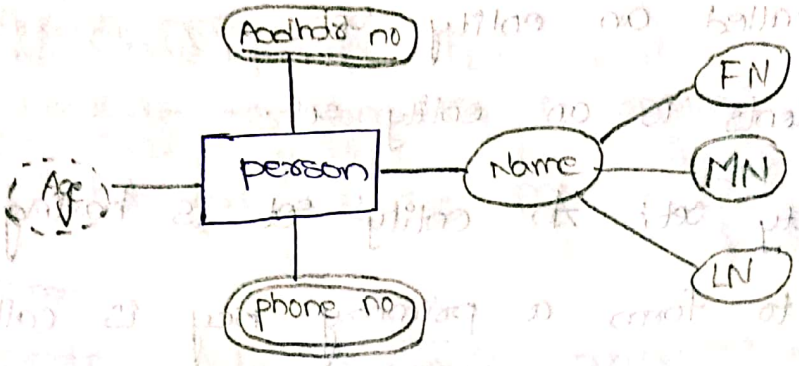
→ generalization

→ Specialization

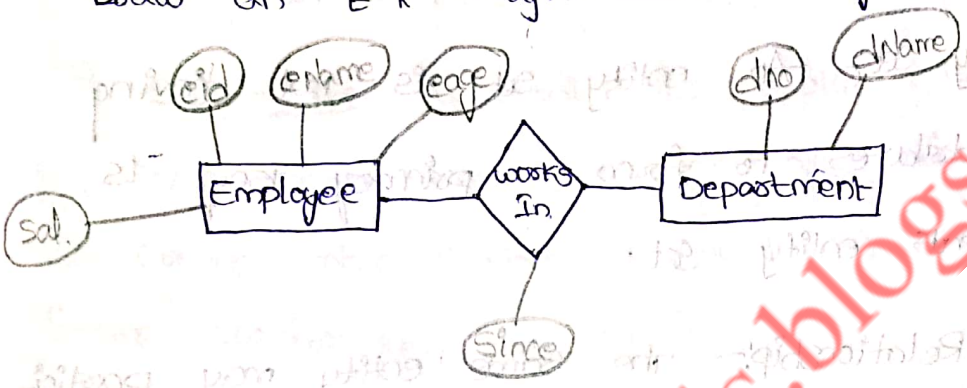
12/7/18

Ex 6 Draw an E-R diagram for person entity?

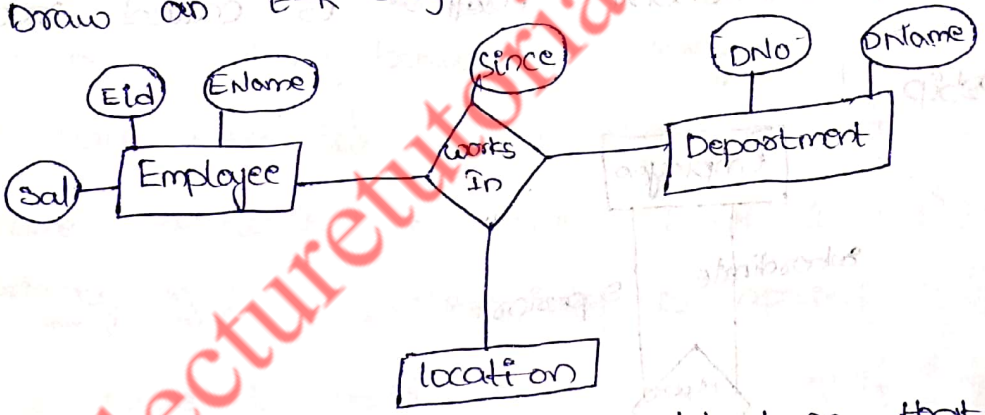
Person (Aadhar NO, Name, phone NO, Age)



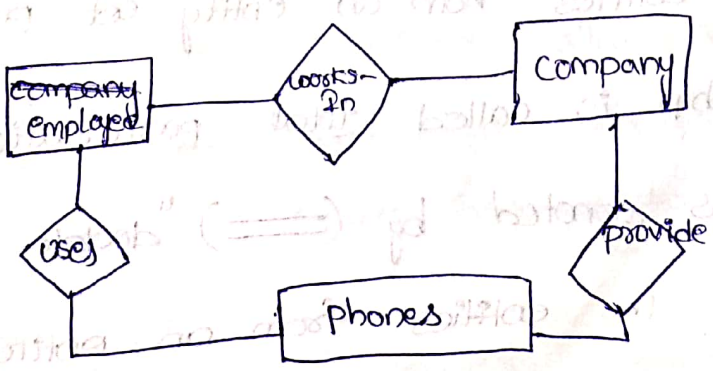
Draw an E-R diagram for Binary Relationship



Draw an E-R diagram for Ternary Relationship



Draw an E-R diagram for the data-base that keeps tracks of company employee phones.



14/7/15

Entity set:- collection of entities of the same type is called an entity set.

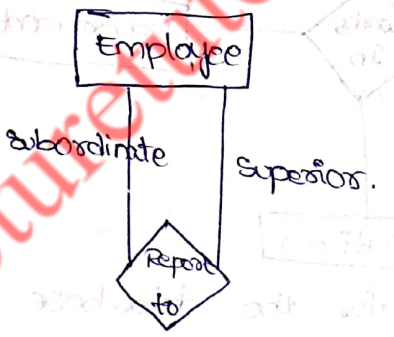
eg:- students is an entity set.

Strong Entity set:- An entity set is having sufficient attributes to form a primary key is called strong entity set

Weak Entity set:- An entity set is not having sufficient attributes to form a primary key is called weak entity set.

Recursive Relationship:- the same entity may participate in a relationship itself is called recursive relationship.

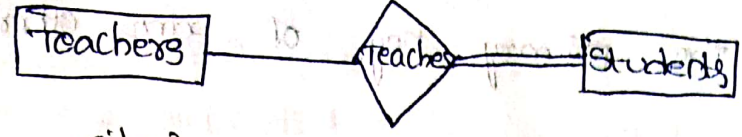
eg:-



Note:-

* If all the entities from an entity set participating in a relationship is called total participation constrain. It is denoted by (═══) "double lines"

* If some of the entities from an entity set participating in a relationship is called partial participation constrain. It is denoted by "single line" (———)

eg:- 

16/7/18 unit-3

Integrity constraints:-

1) NOT NULL:- This constrain is applied to a column, it means that you can not pass a null value to that column

ex:- Create table student(SID number(5) NOT NULL, Sname varchar(20) NOT NULL);

2) Unique:- This constrain is applied to a column, it means that column will not allow duplicates.

ex:- Create table student (SID number(5) NOT NULL UNIQUE, Sname varchar(20) NOT NULL);

3) CHECK:- this constrain is used to restrict the value of a column between a range

ex:- Create table Student (SID number(5) NOT NULL UNIQUE, Sname varchar(20) NOT NULL CHECK (SID > 3));

4) primary KEY:- A primary key is applied to a column, in a table, is used to uniquely identify each row in a table.

* A table can have only one primary key.
* A primary key will not allow duplicates.

ex:- Create table student (SID number(5) PRIMARY KEY, Sname varchar(20), Branch varchar(20));

5) FOREIGN KEY:-

* FOREIGN KEY represents relation ship between the tables
* A FOREIGN KEY of a column, whose values are

derived from the primary key of some other table.

Syntax:-

Create table table_name(col1 datatype (size) PRIMARY KEY, col2 datatype (size), col3 datatype (size) foreign KEY REFERENCES Another_Table_Name);

17/7/18

KEY CONSTRIANTS:-

Keys:- A key allows us to identify a set of attributes that are enough to differentiate entity.

Super key:- A super key is a set of one (or) more attributes that taken collectively, allows us to identify uniquely an entity in the entity set

ex:- SID (Student entity set)

candidate key:- A super key which has no proper subset is called a candidate key.

ex:- SID

primary key:- It is the candidate key is used for identifying and accessing the records.

Composite key:- It is a key which requires more than one attribute

Alternate key:- It is a candidate key not be used for primary key.

Secondary key:- The set of attributes commonly used for accessing records but not necessarily

unique.

Foreign key constraint:-

* A Foreign Key is a key used to links two tables together.
* A Foreign key is a field in one table that refers to the primary key in another table
* The Table containing the foreign key is called child table and the table containing the candidate key is called parent table.

eg:- persons(PID, Names, Age)

Order (OID, Oname, PID)

From the above tables.

* PID is primary key in persons table
* OID is primary key in Orders table.
* PID is foreign key in Orders table

Syntax:-

1) create table persons(PID number(5) PRIM NOT NULL PRIMARY KEY, Name varchar(20), Age number(3));

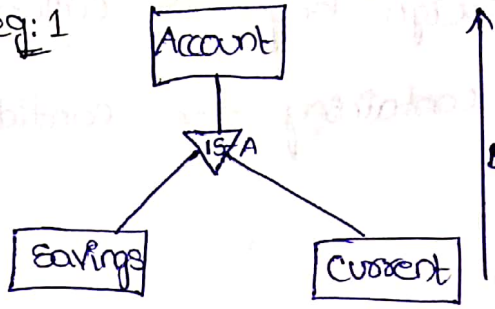
2) create table orders(OID number(5) NOT NULL PRIMARY KEY, Oname varchar(20), PID number(5) Foreign Key References person(PID));

9/7/18

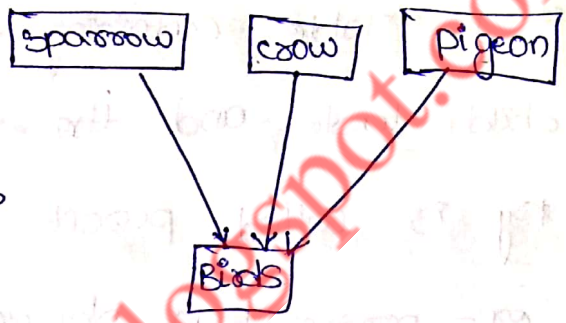
Generalization:- It is Bottom up approach, in which two lower level entities combine to form a higher level entity.

* In generalization a no.of entities are brought together into one generalized entity based on their similar characteristics.

eg:1



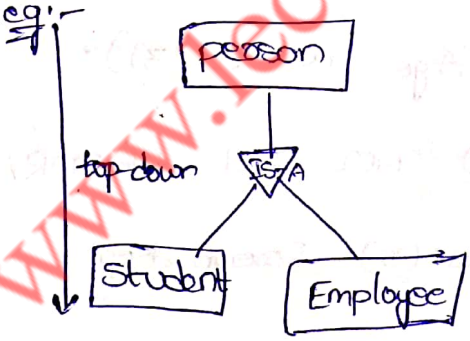
eg:2



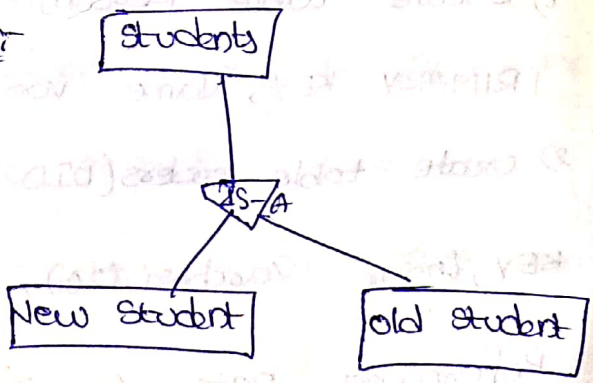
Specialization:- It is a TOP down approach, in which one higher level entity can be broken down into two lower level entities.

* In specialization a group of entities is divided into sub-groups based on their characteristics.

eg:-

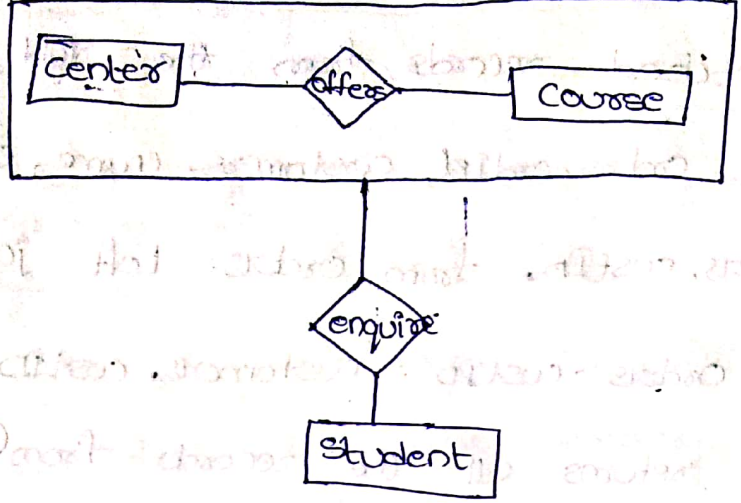


eg:-



Aggregation:-

When relation between two entities is treated as a single entity called as aggregation.



21/7/18 (2)

6M

Joins in SQL :-

* SQL joins are used to combine records from two (or) more tables based on a common field b/w them.

* The most common type of join is simple/Equi/

Inner join

Orders

ordID	CUSID	ordname
111	10	Watch
152	2	Mobile
100	5	Dress
175	7	Laptop
222	18	Books
130	29	Rings

customers.

CUSID	CName	Address
5	cherchu	ongole
7	vyshu	Nellore
9	Bhavani	chennai
10	Sai	Hyderabad
21	Sridhar	Vijayawada
15	Manju	Kandukur

Types of joins:-

1) Inner (simple/Equi) join:- It returns all the rows when there is atleast one match in both tables.

eg:- SQL > Select orders.ordid, customers.Cname, orders.Ordname, customers.custID from orders Inner join customers ON orders.custID = customers.custID;

ii) Left join:- It returns all the records from left table and matched records from the right table.

eg:- SQL> select Orders.OrdID, customers.cname, Orders.OrdName, customers.custID. from orders Left join customers ON Orders.custID = customers.custID;

iii) Right join:- It returns all the records from right table and matched records from left table.

eg:- SQL> select Orders.OrdId, Customers.cname, Orders.OrdName, customers.custID from orders Right join customers ON orders.custID = customers.custID;

iv) full join:- It returns all the records from left and right tables.

eg:- SQL> select orders.ordId, customers.cname, Orders.ordName, customers.custID from orders Full join customers ON Orders.CUSTID = customers.custID;

v) Cross join:- It combines all the records from the first table with every record from the right table.

eg:- SQL> select Emp.eid, Emp.Ename, Dept.Deptno, Dept.Dname from Emp Cross join Dept ON Emp.Deptno = Dept.Deptno;

v) Self joins:- *Joining a table itself is called self join

*this is a type of "Inner join", whose both columns belongs to the same table.

* In this join we need to open two copies of a same table in same memory.

* Since the table name is same for both instances, we use table Aliases to make identical copies of same table, to open in different memory locations.

Eg:- SQL> Select e1.Empid, e1.ename as Fname, e2.ename as Lastname from Employee E1 Inner join Employee E2 ON E1.Empid = E2.Empid;

23/7/18

Like predicate:-

*This allows comparision of one string value with another string value which is not identical.

* this is acheived by using wild card characters (%, _)

* Two types of wild card characters are available.

i) % :- It allows to match any string of any length.

ii) _ :- It allows to match only one character.

Sid	Sname
S1	Sai
S2	Sridhar
D1	chenchu

Sname

In predicate :- In any case a value needs to be compared with a list of values, then "in predicate" is used

eg:- select * from Student where Sname In ('sai', 'Sai', 'chenchu', 'vyshu');

Arithmetic Operators:-

* Oracle allows arithmetic operators, to be used while viewing the records from a table (or) while performing manipulation operations such as insert, update, delete.

* The following are the arithmetic operators.

Addition (+), substraction (-), multiplication (*), division (/)

SNO	SName	Marks
561	Sai	500
562	Lakshmi	550
563	Vyshu	600
564	Bhavani	580
565	Sridhar	520

eg:- List out the student details, after adding marks 60 to each and every student.

→ select * from Student where Marks = Marks + 60;

→ select * from Student where marks = marks - 60;

→ select * from student where marks = marks * 2;

→ select * from Student where marks = marks / 2;

25/7/18

Logical operators:-

The following are the logical operators used in Oracle

1) AND operator:- this operator allows to create an SQL statement based on two (or) more conditions being met

2) OR operator:- This operator allows to create an SQL statement, where records are written when any one of the condition being met.

3) NOT operator:- This operator display the records from a table that do not satisfy the condition

4) BETWEEN operator:- This operator is used to select the data within a range of values

Student

SID	Sname	marks	Age
61	Ram	400	20
62	Raju	450	24
63	Ravi	500	22
64	Rakesh	550	21

Eg:1 list out the student details with marks and age, where the marks between a range of 410 to 510 & age b/w a range of 21 to 24.

Ans:- select marks, age from student where (marks >=410 AND marks <=510) AND (Age >=21 AND Age <=24);

Eg:2 List out the student names where the names are Ram or Rakesh or Chenchu or Sai.

select Sname from Student where Sname = ('Ram' OR 'Rakesh' OR 'chenchu' OR 'Sai');

Eg:- List out the student details where the age not in a range of 20 or 24.

→ select * From Student WHERE NOT (Age=20 or Age = 24)

Eg:- list out the student details with name, age where the age is in the range of 23 to 25.

→ select name, age from Student where Age BETWEEN 23 to 25;

UNIT-3

SET operators(OR) Relational SET operators:-

These operators are used to combine the information of similar data type from one (or) more tables.

Data type of corresponding columns in all the select statements should be same.

25/7/18

UNION:- Combining the results of two select statements into one result set and then eliminates any duplicate rows from the result set

Ex:- select Eno, Ename from Emp, where DeptNo = 10 UNION select ENO, Ename from Emp where DeptNo=20;

UNION ALL:-

Combining the results of two select statements into one result set including duplicates from the result set.

Eg:- select Eno, Ename from Emp where DeptNo = 10 UNION ALL select Eno, Ename from Emp where DeptNo=20;

INTERSECT :- It returns only these rows that are returned by each of the two select statements

eg:- select Eno, Ename from Emp where DeptNo=10
INTERSECT select Eno, Ename from Emp where DeptNo 20

MINUS :- It returns the results set of one select statement and removes those rows that are returned by second select statement.

eg:- select Eno, Ename from Emp where DeptNo = 10 MINUS select Eno, Ename from Emp where DeptNo = 20;

unit-3

NULL VALUES:-

* "NULL" is a special marker used in SQL to indicate that a data value does not exist in the data base.
* In SQL "NULL" is the term is used to represent a missing value
* An SQL a "NULL" value in a table is a value in a field that appears to be blank.
* "NULL" value is not different than a "zero" value or a field that contains spaces
* The following Syntax is used for "NULL" while creating a table.

Syntax:- create table tablename (col1 Datatype (size) primary key NOT NULL, col2 Datatype (size) NOT NULL, col3 Datatype (size));

SUB QUERIES:- A Sub-Query (or) Inner Query (or) Nested Query is a query with in another SQL query and embedded with in "WHERE class".

* A Sub-Query is used to return data that will be used in the main query as a condition.

* Sub Queries can be used with SELECT, INSERT, UPDATE, DELETE statements along with operators. like =, <, <=, >, >= IN, BETWEEN ....

* The following rules must be use for defining a sub query ( )

* Sub queries must be enclosed with in "parenthesis"

* A sub query can have only one column in the select statement.

* An ORDER BY command can not be used in a Sub query

* BETWEEN operator can not be used with a Sub query, but can be used in with in the Sub query.

eg:-

Customer

Id	Name	Age	Address	Salary
1.	chenchu	20	ongole	45,000
2.	vyshu	19	Nellore	40,000
3.	Bhavani	25	Hyd	50,000
4.	Sridhar	22	Chennai	55,000

Sub Queries with SELECT Statements:-

* The Sub Query can be used with Select Statement, the basic syntax is as follows

Syntax:-

Select columnNames from TableName WHERE colName OPERATOR (Select colName from tableName where Condition);

Ex:- Select * from customers where ID IN(select ID from customers where salary > 50,000);

Sub Queries with INSERT statement :-

* A Sub Query can be used with INSERT statement, the INSERT statement used the data returned from the sub Query to insert into another table.

ex:- INSERT INTO customersBackup select * from customers where ID IN (select ID from customers);

Sub Queries with UPDATE statement :-

The Sub Queries can be used in UPDATE statement, either single (or) multiple columns in a table can be updated.

eg:- UPDATE salary by 0.25 times in customer table for all the customers whose AGE is >= 25

→ UPDATE customer set salary = salary * 0.25 where Age IN (select AGE from customer where AGE >=25)

Sub Queries with DELETE Statement :-

* The sub queries can also be used with delete statements

Eg:- Delete the records from customer table for all the customers whose AGE >=25

→ DELETE from customer where AGE IN (Select AGE from customer where AGE >=25)

Relational Algebra :-

* It is a procedural query language which takes instances as input and yields instances of relation as output

* It uses operators to perform queries.

* An operator can be either unary or binary

* The fundamental operations of relational algebra are as follows * SELECT (σ)
* PROJECT (π)
* UNION (U)
* set Difference (−)
* Cartesian product (X)
* Rename (ρ)

Select Operation(σ) :-

It selects tuples (rows) that satisfy the given predicate from a relation

Notation :- $\sigma_p(r)$

where 'r' is a relation
'p' is propositional logic formula. which

may use connectors like AND, OR, NOT, these forms may use relational operators like =, ≠, >, >=, <, <=

Ex:- σ (Books)
subject = 'database'

o/p:- selects tuples from books where subject is database

Project operation (Π):-

It PROJECTS columns (fields) that satisfy a given predicate.

Notation:- $\Pi_{A_1, A_2 \cdots A_n}(\sigma)$

*Duplicate rows are automatically eliminated

eg:- $\Pi_{\text{subject, author}}$ (Books)

UNION operation (U):-

* It performs binary union b/w two given relations and is defined as $r \cup s = \{t / t \in r \text{ or } t \in s\}$

* For a union operation the following conditions must be hold.

1. r and s must have same no. of attributes
2. Attribute domains must be compatable

Ex:- Π_{author} (Books) ∪ Π_{author} (Article)

o/p:- PROJECTS the names of authors who have written a book (or) an article (or) both

Set Difference (−):-

The result set is a tuple which are present in one relation but not in second relation

Notation:- r − s

Ex:- Π_{author} (Books) − $\Pi_{authors}$ (Articles)