DATABASE MANAGEMENT SYSTEM

(10211DS207)

TASK NO: 12

ORPHANAGE MANAGEMENT SYSTEM

TEAM DETAILS:

Team Leader: PRONAY MONDAL(VTU 30566)

Reg No: 24UEDC0054

Team Members:

Names:	VTU:	Registration No:
Talakonda Gopalaswamy	VTU30540	24UEDC0005
Gujjneni Harikrishna	VTU30631	24UESS0019
Annem Naga Gotham Reddy	VTU30576	24UEDC0064
Katreddy Gurunath Reddy	VTU33242	

DATABASE MANAGEMENT SYSTEM

(10211DS207)

TASK NO: 12

ORPHANAGE MANAGEMENT SYSTEM

TEAM DETAILS:

Team Leader: PRONAY MONDAL(VTU 30566)

Reg No: 24UEDC0054

Team Members:

Names:	VTU:	Registration No:
Talakonda Gopalaswamy	VTU30540	24UEDC0005
Gujjneni Harikrishna	VTU30631	24UESS0019
Annem Naga Gotham Reddy	VTU30576	24UEDC0064
Katreddy Gurunath Reddy	VTU33242	

Aim:
To develop a microproject on Orphanage Management System.

1.ER Diagram

The ER diagram models the data structure of an orphanage management system. It helps visualize how different entities—such as children, staff, donors, and rooms—are related, enabling efficient database design and streamlined operations like tracking admissions, donations, events, and medical records.

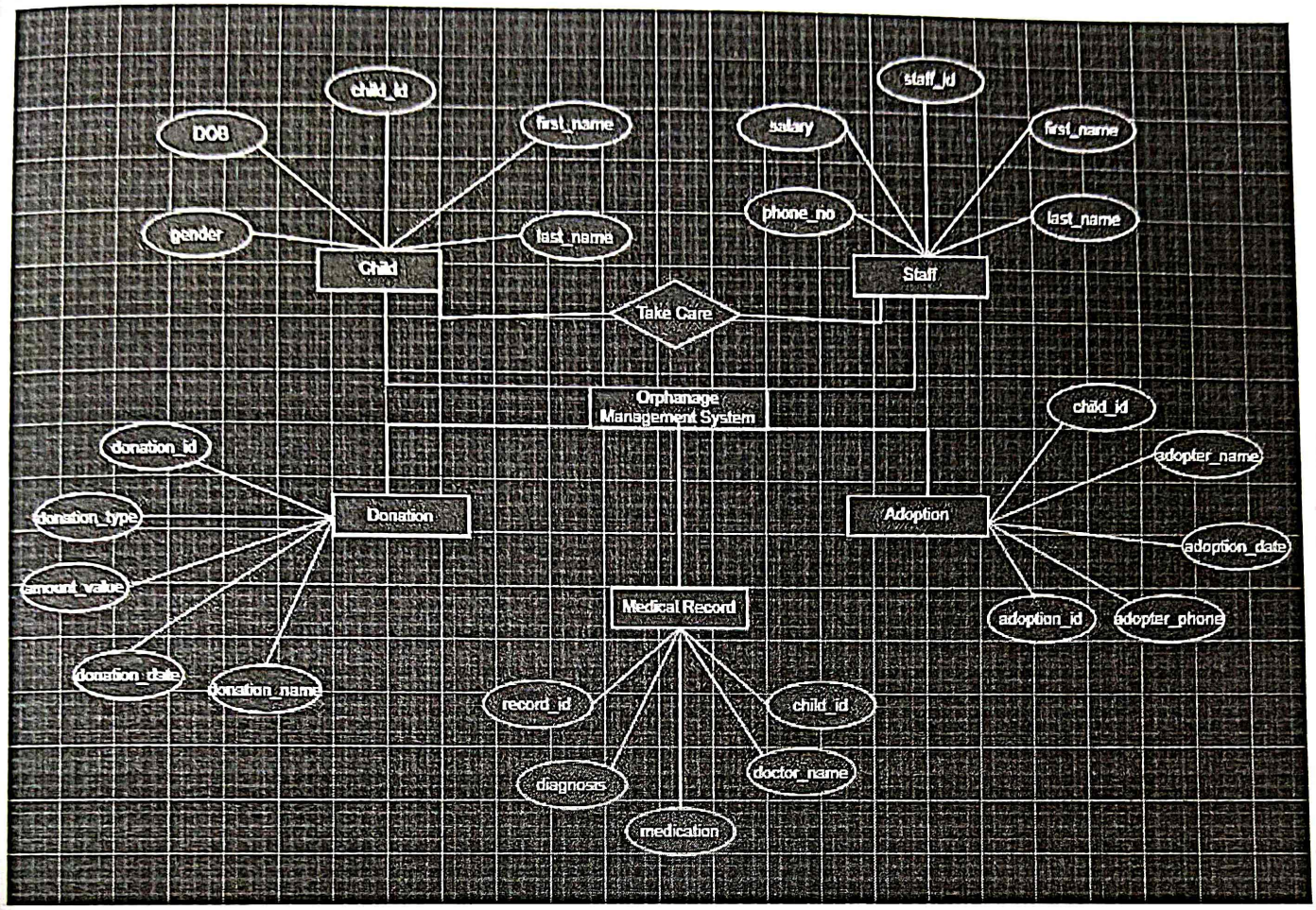
Entities:

1. Child
   -Attributes: Child_ID(PK), First name, Last Name, Gender.
2. Staff
   -Attributes: Staff_ID(PK), First name, Last Name, Phone_no.
3. Adoption
   -Attributes: Adopter_name, Adopter_date, Adopter_phone, Adoption_ID.
4. Medical Record
   -Attributes: Medication, Doctor_name, Diagnosis, Record_ID.
5. Donation
   -Attributes: Donation_ID, Donation_type, Amount_value, Donation_date.

Relationships:

- One Staff member (Caretaker) can care for many Children, but each Child is assigned to one Caretaker.
- One Child has many Medical Records over time.
- One Sponsor can make multiple Donations.
- A Child can only be involved in one final Adoption record (though they may have many pending inquiries).

ER Diagram:


child_id
DOB
first_name
gender
last_name
Child
staff_id
salary
first_name
phone_no
last_name
Staff
Take Care
Orphanage Management System
donation_id
donation_type
amount_value
donation_date
donation_name
Donation
child_id
adopter_name
adoption_date
adoption_id
adopter_phone
Adoption
Medical Record
record_id
child_id
diagnosis
doctor_name
medication

2. SQL Queries and relational operations:

I) Select operation:

SELECT first_name, last_name, gender FROM Child;

first_name	last_name	gender
Alice	Brown	Female
Ben	Smith	Male
Clara	Johnson	Female
David	Lee	Male
Ella	Davis	Female

II) Project operation:

SELECT child_id, date_of_birth FROM Child;

child_id	date_of_birth
1	2014-05-21
2	2013-11-12
3	2015-07-19
4	2016-03-05
5	2012-09-30

III) Union All:

```
SELECT child_id, date_of_birth FROM Child
UNION ALL
SELECT child_id, date_of_birth FROM Child
WHERE date_of_birth > '2015-01-01';
```

child_id	date_of_birth
1	2014-05-21
2	2013-11-12
3	2015-07-19
4	2016-03-05
5	2012-09-30
3	2015-07-19
4	2016-03-05

IV) Union:

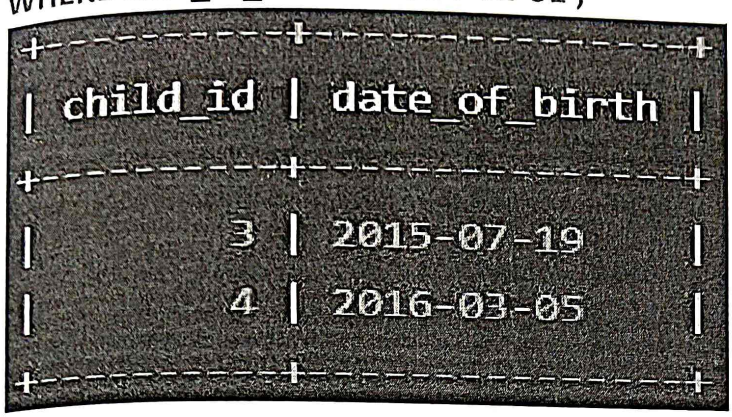
```
SELECT child_id, date_of_birth FROM Child
UNION
SELECT child_id, date_of_birth FROM Child
WHERE date_of_birth > '2015-01-01';
```

child_id	date_of_birth
1	2014-05-21
2	2013-11-12
3	2015-07-19
4	2016-03-05
5	2012-09-30

V) Intersect:

```
SELECT child_id, date_of_birth FROM Child
INTERSECT
SELECT child_id, date_of_birth FROM Child
```

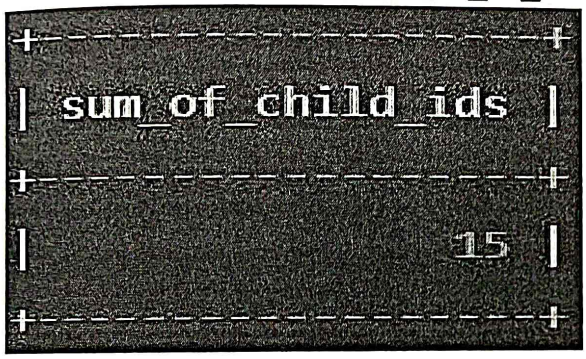
WHERE date_of_birth > '2015-01-01';



child_id	date_of_birth
3	2015-07-19
4	2016-03-05

VI) **Sum:**

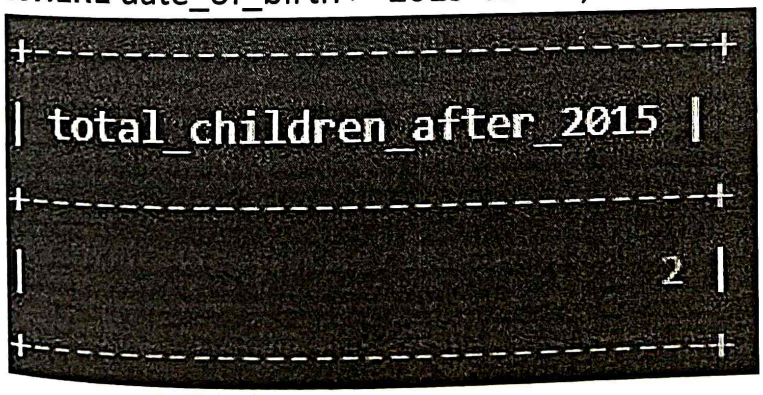
SELECT SUM(child_id) AS sum_of_child_ids FROM Child;



sum_of_child_ids
15

VII) **Count:**

SELECT COUNT(*) AS total_children_after_2015
FROM Child
WHERE date_of_birth > '2015-01-01';



total_children_after_2015
2

VIII) AVG:

```
SELECT AVG(staff_caretaker_id) AS average_caretaker_id FROM Child;
```

average_caretaker_id
102.2

IX) Minimum and Maximum:

```
SELECT MIN(child_id) AS min_id, MAX(child_id) AS max_id FROM Child;
```

min_id	max_id
1	5

X) Nested Queries:

```
SELECT * FROM Child
WHERE child_id = (
  SELECT MAX(child_id) FROM Child
);
```

child_id	first_name	last_name	date_of_birth	date_admitted	gender	status	staff_care
5	Eva	Garcia	2012-09-30	2023-06-05	F	Active	102

XI) PL SQL Procedure:

Average child ID of children born after 2015.

```
DECLARE
    v_avg_child_id NUMBER;
BEGIN
    -- Calculate average child_id for children born after 2015
    SELECT AVG(child_id)
    INTO v_avg_child_id
    FROM child
    WHERE date_of_birth > TO_DATE('2015-01-01', 'YYYY-MM-DD');

    -- Output the result
    DBMS_OUTPUT.PUT_LINE('Average child ID (born after 2015): ' || ROUND(v_avg_child_id, 2));
END;
/
```

Output:

```
Average child ID (born after 2015): 3.5
```

JOINS:

A) INNER JOINS:

```
SELECT
c.child_id,c.first_name,c.last_name,c.date_of_birth,s.staff_name,s.staff_rol
e
FROM
  Child c
INNER JOIN
  Staff s
ON
  c.staff_caretaker_id = s.staff_caretaker_id;
```

child_id	first_name	last_name	date_of_birth	staff_name	staff_role
1	Alice	Brown	2014-05-21	Mr. Adams	Supervisor
2	Ben	Smith	2013-11-12	Ms. Taylor	Caregiver
3	Clara	Johnson	2015-07-19	Mr. Clark	Nurse
4	David	Wilson	2016-03-05	Mr. Adams	Supervisor

B) LEFT OUTER JOIN:

```
SELECT
c.child_id,c.first_name,c.last_name,c.date_of_birth,s.staff_name,s.staff_rol
e
FROM
  Child c
LEFT OUTER JOIN
  Staff s
ON
  c.staff_caretaker_id = s.staff_caretaker_id;
```

child_id	first_name	last_name	date_of_birth	staff_name	staff_role
1	Alice	Brown	2014-05-21	Mr. Adams	Supervisor
2	Ben	Smith	2013-11-12	Ms. Taylor	Caregiver
3	Clara	Johnson	2015-07-19	Mr. Clark	Nurse
4	David	Wilson	2016-03-05	Mr. Adams	Supervisor
5	Eva	Garcia	2012-09-30	NULL	NULL

C) RIGHT OUTER JOINS:

```
SELECT
c.child_id,c.first_name,c.last_name,c.date_of_birth,s.staff_name,s.staff_rol
e
FROM
  Child c
RIGHT OUTER JOIN
  Staff s
ON
  c.staff_caretaker_id = s.staff_caretaker_id;
```

child_id	first_name	last_name	date_of_birth	staff_name	staff_role
1	Alice	Brown	2014-05-21	Mr. Adams	Supervisor
4	David	Wilson	2016-03-05	Mr. Adams	Supervisor
2	Ben	Smith	2013-11-12	Ms. Taylor	Caregiver
3	Clara	Johnson	2015-07-19	Mr. Clark	Nurse
NULL	NULL	NULL	NULL	Ms. Lee	Volunteer

D) **FULL OUTER JOIN:**

```
SELECT
c.child_id,c.first_name,c.last_name,c.date_of_birth,s.staff_name,s.staff_rol
e
FROM
   Child c
FULL OUTER JOIN
   Staff s
ON
   c.staff_caretaker_id = s.staff_caretaker_id;
```

child_id	first_name	last_name	date_of_birth	staff_name	staff_role
1	Alice	Brown	2014-05-21	Mr. Adams	Supervisor
4	David	Wilson	2016-03-05	Mr. Adams	Supervisor
2	Ben	Smith	2013-11-12	Ms. Taylor	Caregiver
3	Clara	Johnson	2015-07-19	Mr. Clark	Nurse
5	Eva	Garcia	2012-09-30	NULL	NULL
NULL	NULL	NULL	NULL	Ms. Lee	Volunteer

4.Normalization :

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity.

We normally go through stages called **Normal Forms (1NF → 2NF → 3NF → BCNF)**.

For our **Orphanage Management System**, we can start from a sample normalized relation and use the

Griffith Normalization Tool to automate the process.

1. First Normal Form (1NF)

A relation (table) is in **1NF** if all its attributes contain **atomic (indivisible) values,** and each row

is unique.

Key point: No repeating groups or arrays.

2. Second Normal Form (2NF)

A relation is in **2NF** if it is in 1NF and every non-prime attribute is fully functionally dependent on the whole primary key.

Key point: No partial dependency on a part of a composite key.

3. Boyce-Codd Normal Form (BCNF)

A relation is in BCNF if it is in 2NF and every determinant is a superkey.

Key point: Even stricter than 2NF; eliminates anomalies caused by non-key attributes determining other attributes.

- In this databse we perform normalization using Griffith university normalization tool.

First Normal Form:

Attributes in Table

Separate attributes using a comma (,)

Adopter_ID, Adopter_name, Adopter_date, Adopter_phone

Functional Dependencies

Enter the functional dependencies that are present in your table.

Adopter_ID ×	→	Adopter_name × Adopter_phone ×	Delete
Adopter_name ×	→	Adopter_ID × Adopter_phone ×	Delete
Adopter_phone ×	→	Adopter_name × Adopter_ID × Adopter_date ×	Delete

Add Another Dependency

Check Normal Form

2NF

The table is in 2NF

3NF

The table is in 3NF

BCNF

The table is in BCNF

Show Steps

2NF

```
find all candidate keys. The candiates keys are {Adopter_ID}, {Adopter_name}, {Adopter_phone}. The set of
key attributes are: {Adopter_ID,Adopter_name,Adopter_phone}
for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not
all key attributes
checking FD: Adopter_ID --> Adopter_name,Adopter_phone
checking FD: Adopter_name --> Adopter_ID,Adopter_phone
checking FD: Adopter_phone --> Adopter_name,Adopter_ID,Adopter_date
```

3NF

```
find all cadnidate keys. The candiates keys are {Adopter_ID}, {Adopter_name}, {Adopter_phone}. The set of
key attributes are: {Adopter_ID,Adopter_name,Adopter_phone}
for each FD, check whether the LHS is superkey or the RHS are all key attributes
checking functional dependency Adopter_ID --> Adopter_name,Adopter_phone
checking functional dependency Adopter_name --> Adopter_ID,Adopter_phone
checking functional dependency Adopter_phone --> Adopter_name,Adopter_ID,Adopter_date
```

BCNF

```
A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.
```

Normalize to 2NF

Attributes in Table

Separate attributes using a comma (,)

Adopter_ID, Adopter_name, Adopter_date, Adopter_phone

Functional Dependencies

Enter the functional dependencies that are present in your table.

Adopter_ID ×	→	Adopter_name ×, Adopter_phone ×	Delete
Adopter_name ×	→	Adopter_ID ×, Adopter_phone ×	Delete
Adopter_phone ×	→	Adopter_name ×, Adopter_ID ×, Adopter_date ×	Delete

Add Another Dependency

Normalize to 2NF

Attributes

Adopter_ID Adopter_name Adopter_date Adopter_phone

Functional Dependencies

Adopter_ID → Adopter_phone

Adopter_name → Adopter_phone

Adopter_phone → Adopter_name Adopter_ID Adopter_date

Show Steps

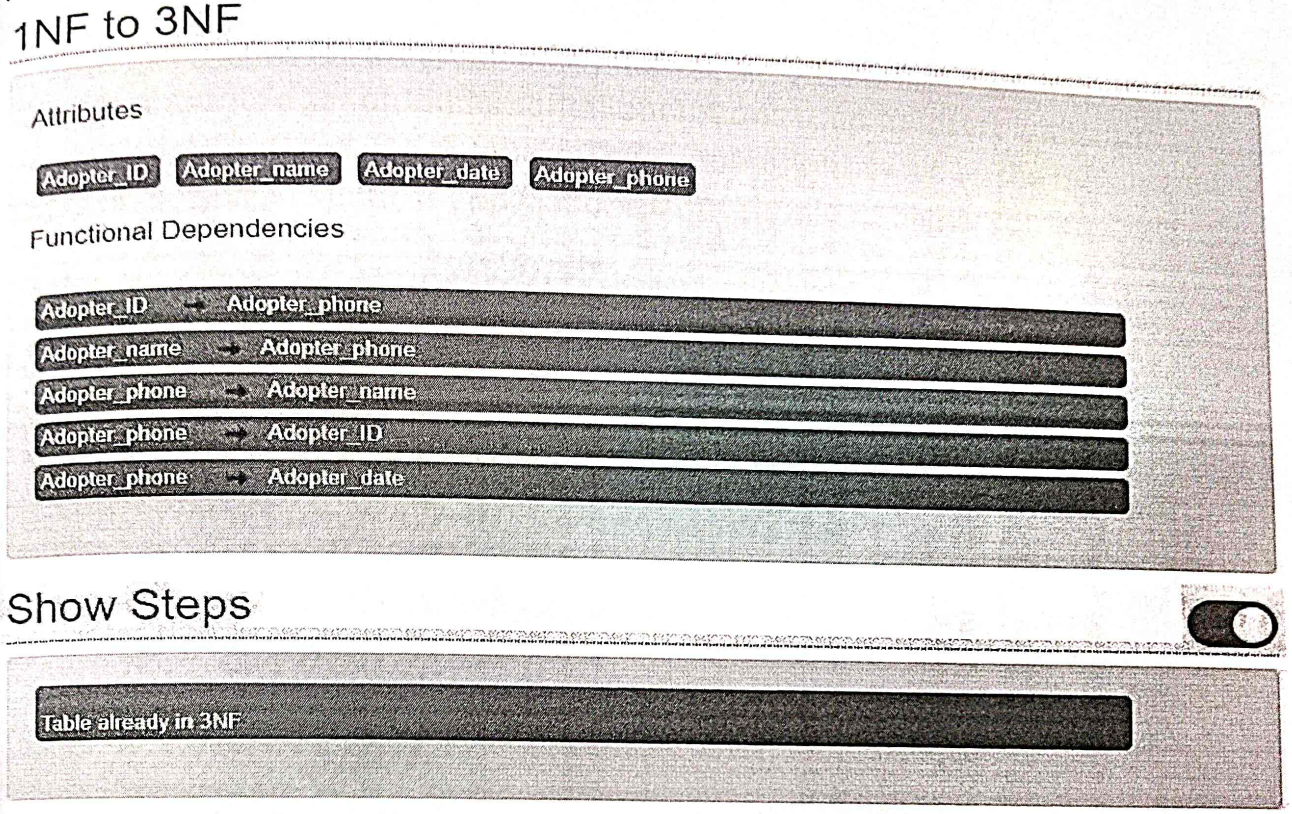
First, find the minimal cover of the FDs, which includes the FDs
Adopter_ID --> Adopter_phone
Adopter_name --> Adopter_phone
Adopter_phone --> Adopter_name
Adopter_phone --> Adopter_ID
Adopter_phone --> Adopter_date

Initially rel[1] is the original table

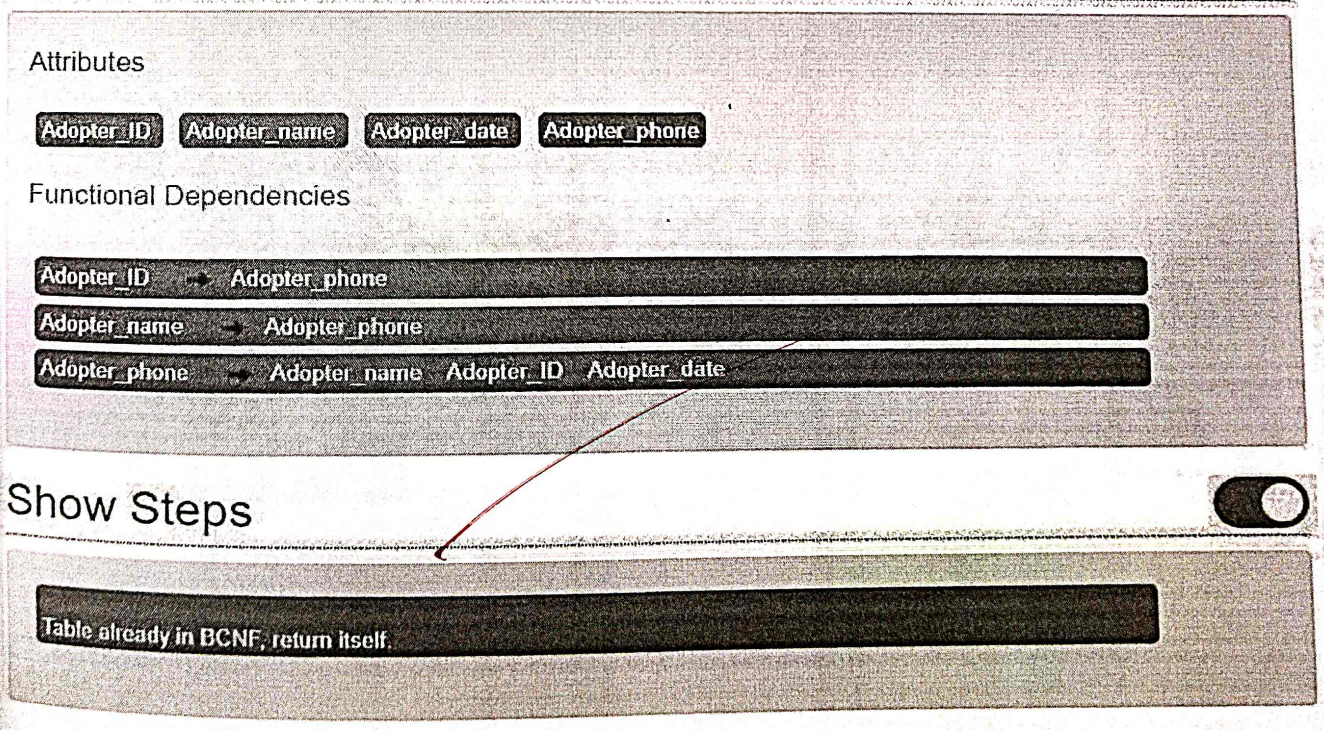
Round1: checking table rel[1]

***** The table is in 2NF already, send it to output *****

Normalize to 3NF:



Normalize to BCNF:



5.Document database using MONGODB :

CRUD,Which stands for Create,Read,Update,and Delete,represents a set of fundamental operations used to insert with and manipulate data stored in a database.These operations serve as the building blocks upon which countless applications,from simple to highly complex,rely.

Create:

```
db.children.deleteOne({ first_name: "Anya", last_name:
 "Johnson"
});
db.children.deleteMany({ status: "Left"
});
```

Output:

```
{
  "ok": 1,
  "acknowledged": true,
  "insertedIds": [
    ObjectId("68efbce91803d6ea926ab44a"),
    ObjectId("68efbce91803d6ea926ab44b")
  ]
}
```

Read:

```
db.children.deleteOne({
 first_name: "Anya",
 last_name: "Johnson"
});
db.children.deleteMany({ status: "Left"
});
```

```
"_id": ObjectId("66e8c4e09f5c4f3468b31a20")
});
```

Output:

```
{
  "acknowledged": true,
  "deletedCount": 1
}
```

6.Graph Database using MONGODB(using neo4j online compiler)

(a) Create a graph database:

MERGE (c1:Child {child_id: 'C001', first_name: 'Anjali', last_name: 'Devi', DOB: '2018-05-15', gender: 'F'})

Output:

Created 1 node, set 2 properties, added 1 label

MERGE (s1:Staff {staff_id: 'S01', first_name: 'Meena', last_name: 'Reddy', salary: 45000, phone_no: '9876543210'})

Output:

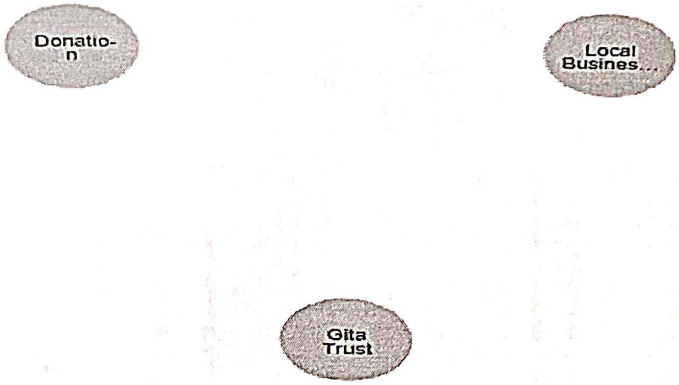
Created 1 node, set 4 properties, added 1 label

MERGE (d1:Donation {donation_id: 'D101', donation_type: 'Cash', amount_value: 50000, donation_date: '2025-09-01', donation_name: 'Gita Trust'})

Output:

Created 1 node, set 3 properties, added 1 label

Output:

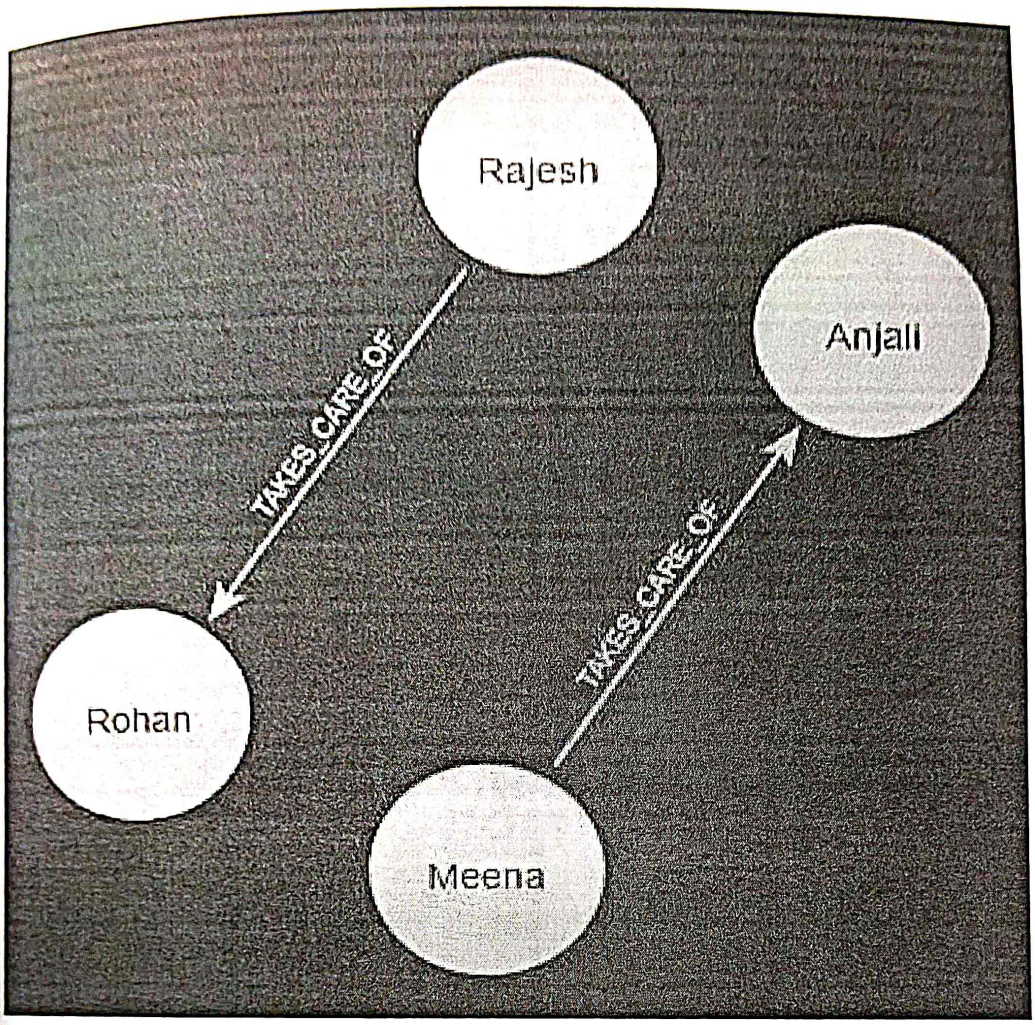



```
MATCH (c1:Child {child_id: 'C001'}), (c2:Child {child_id: 'C002'}),

(s1:Staff {staff_id: 'S01'}), (s2:Staff {staff_id: 'S02'})
MATCH (s:Staff)-[r:TAKES_CARE_OF]->(c:Child)

RETURN s, r, c
```

Output:



(b) Delete a node from Donation:

Syntax:

```
MATCH (d:Donation {donation_name: 'Gita Trust'})
DETACH DELETE d;
```

Output :Deleted 1 node, deleted 1 relationship

Thus,the document database and graph database by using mongodb is implemented.

Result:

Thus,Micro Project for Orphanage Management System was developed and implemented successfully.

VEL TECH	
EX NO.	12
PERFORMANCE (5)	5
RESULT AND ANALYS'S (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	29/10/21