Data And Application

Project Phase - 3

Team No. - 57

Team Name - Demolishers

Team members Details

Name	Roll no.	Email ID
Abhinav Anand	2020101054	abhinav.a@students.iiit.ac.in
Lavisha Bhambri	2020101088	lavisha.bhambri@students.iiit.ac.in
Mayank Bhardwaj	2020101068	mayank.bhardwaj@students.iiit.ac.in

Contents

Task 1: Conversion from ER Model to Relational Model

1.1 Mapping Strong entity types to Relations

For every regular strong entity type **E** in the schema, we created a relation that includes all simple attributes of **E.**

Employee	
PK	department_no
PK	serial_no
	aadhar_no
	date_of_birth
	first_name
	middle_name
	surname
	email_id
	street_address
	zip_code
	city
	state
	age

Insurance Policy	
PK	policy_id
	customer_name
	terms_and_conditions
	date_of_issue
	duration
	Premium Value
	Sum assured

Third Party Administrator	
PK	TPA_id
	TPA_name
	street_address
	zip_code
	city
	state

Customer	
PK	aadhar_no
	date_of_birth
	first_name
	middle_name
	surname
	email_id
	customer_status
	street_address
	zip_code
	city
	state
	age

Figure 1.1: Mapping Strong Entites as Relations

1.2 Mapping of Weak Entity types

For every weak entity type **W** in the ER schema with owner entity type **E**, we have created a relation and included all simple attributes as attributes of relations. We also included the primary key of the owner entity type **E** as the foreign key of the relation. The primary key of **W**'s relation is the combination of **E**'s primary key and partial key of **W**.

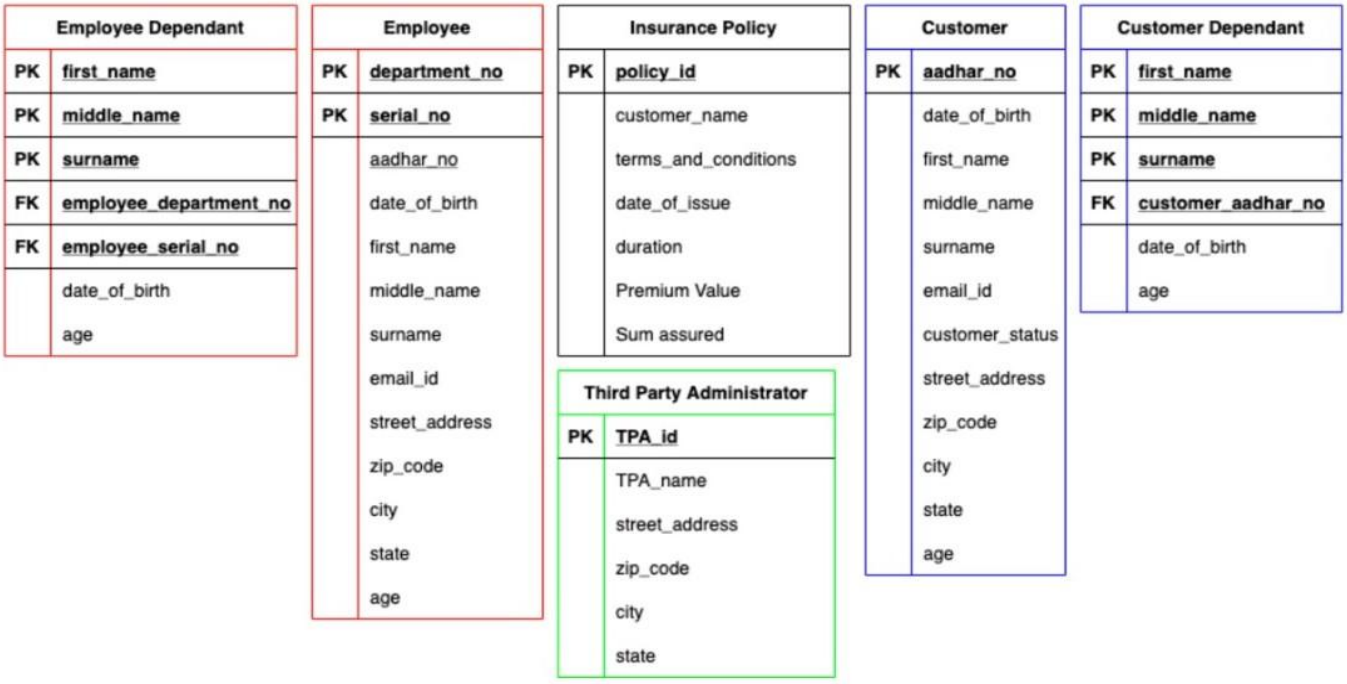

Figure 1.2: Mapping Weak Entities as Relations

1.3 Mapping of Binary 1:1 Relationship Types

Since we don't have any binary 1:1 relationships, we do not need to do anything.

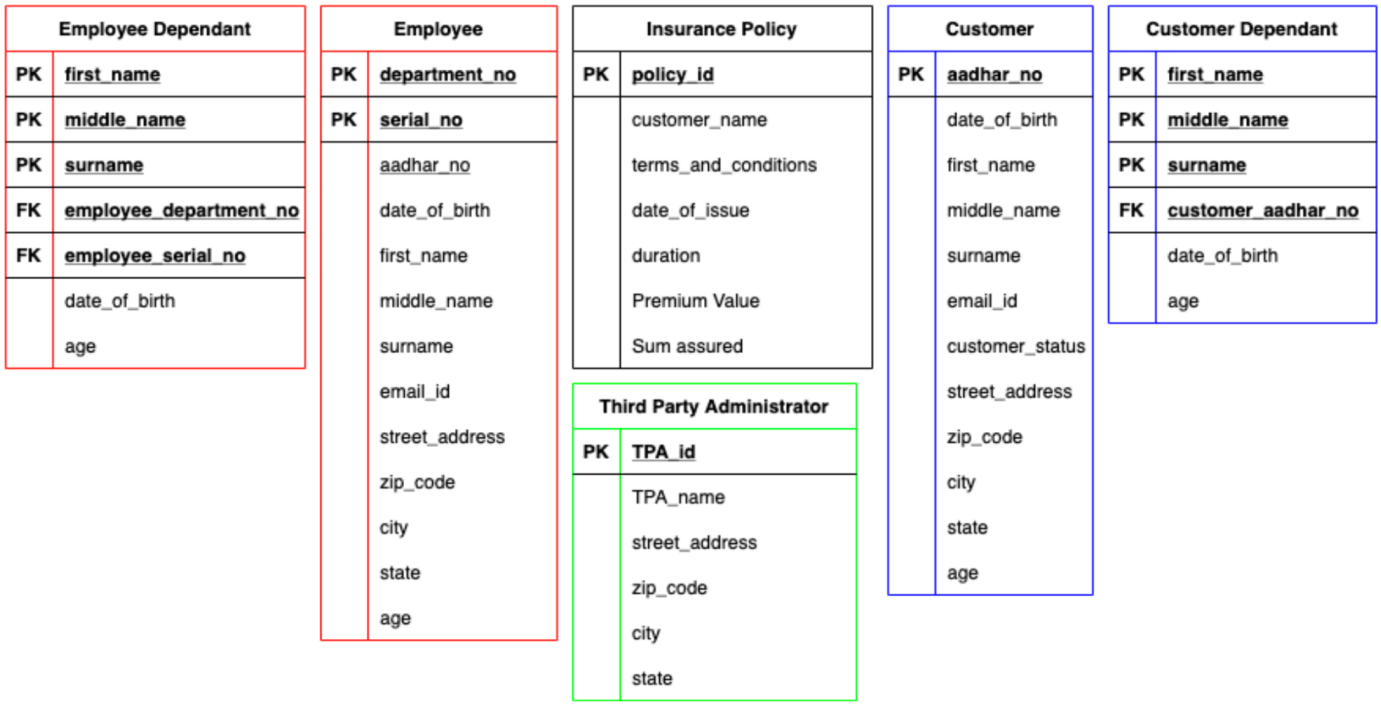

Figure 1.3: Mapping all Binary 1:1 Relationship types

1.4 Mapping of Binary 1:N Relationship types

For each binary 1:N relationship, in the entity with cardinality 1 we add the primary key of the other entity as a foreign key and map these keys to each other.

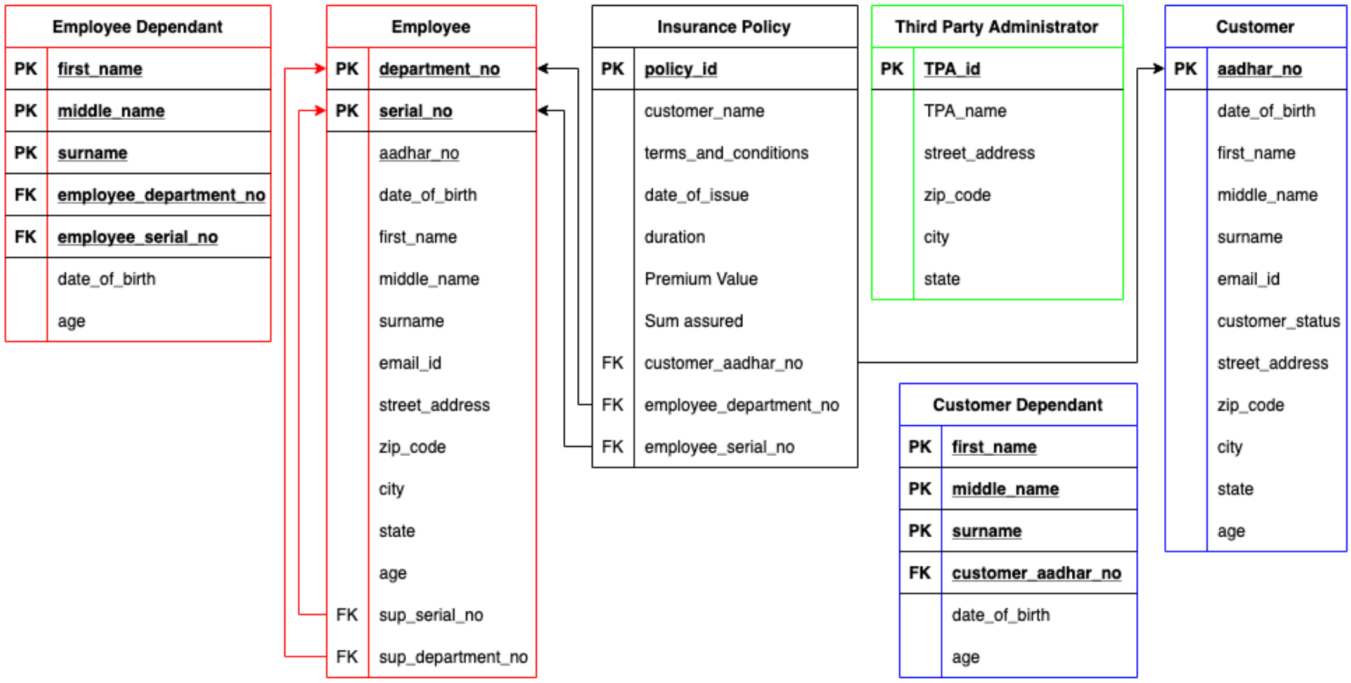

Figure 1.4: Mapping all Binary 1:N Relationship types

1.5 Mapping of Binary M:N Relationship types

For each binary M:N relationship type **R**, we are creating a new relationship relation S which includes the primary keys of participating entities as foreign keys.

Figure 1.5: Mapping all Binary M:N Relationship types

1.6 Mapping of Multi-valued Attributes

For each multi-valued attribute **A**, create a new relation **R** that will include attributes corresponding to **A**, the primary key attribute **K** as the foreign key in **R**. The primary key is a combination of **A** and **R**.

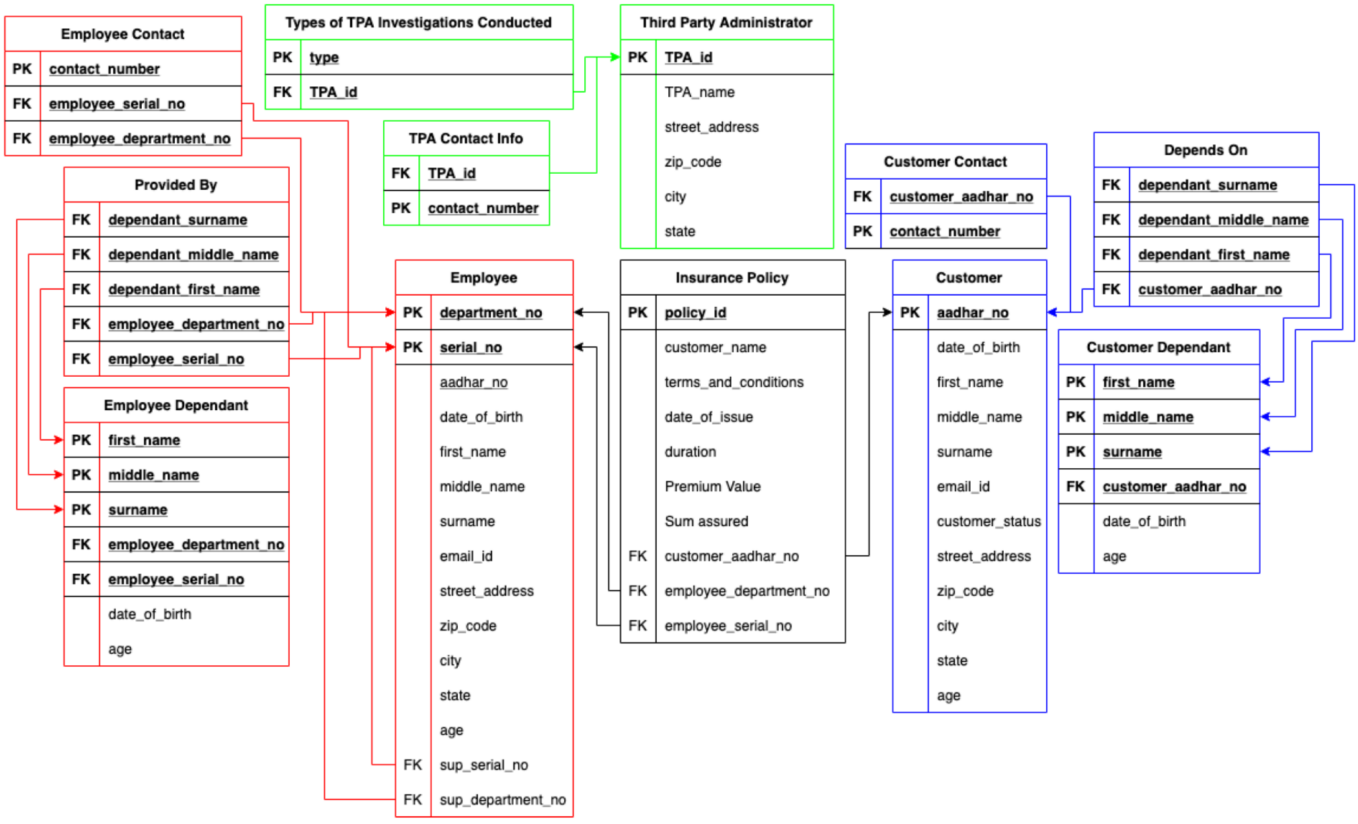

Figure 1.6: Mapping all multivalued attributes

1.7 Mapping of N-ary Relationship Types

For each n-ary relationships, we create a relation **R** which has the primary keys of the participating entities as it's foreign keys, and primary key. **R** also contains the attributes of the n-ary relationship.

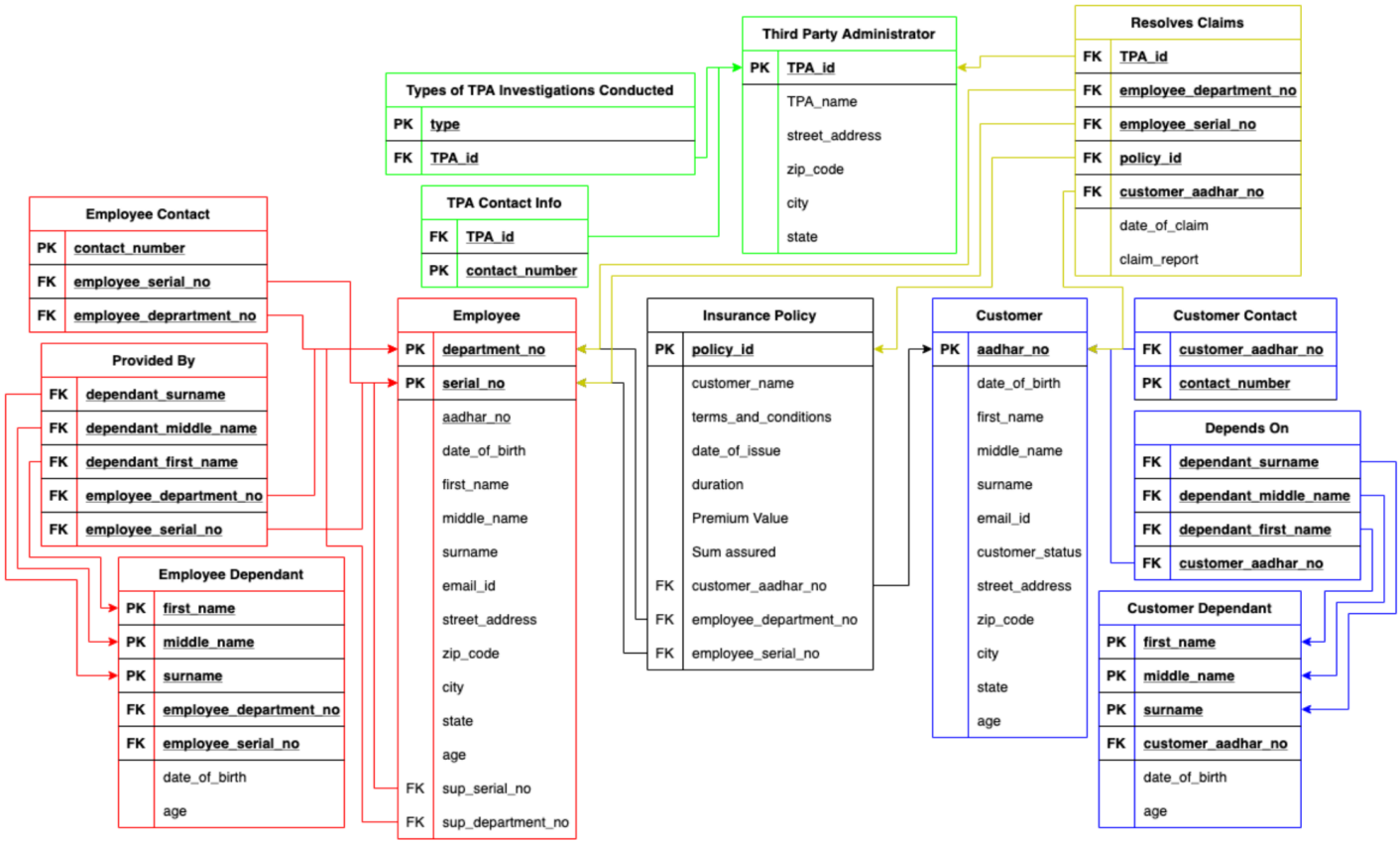

Figure 1.7: Mapping all N-ary Relationship types

1.8 Mapping of Superclasses and Subclasses

For every subclass of the superclass **Insurance Policy** we create a new relation whose primary keys and foreign keys are the primary key of its superclass **Insurance Policy**. We add their simple attributes in these relations and then we follow steps 2-7 for these newly created relations.

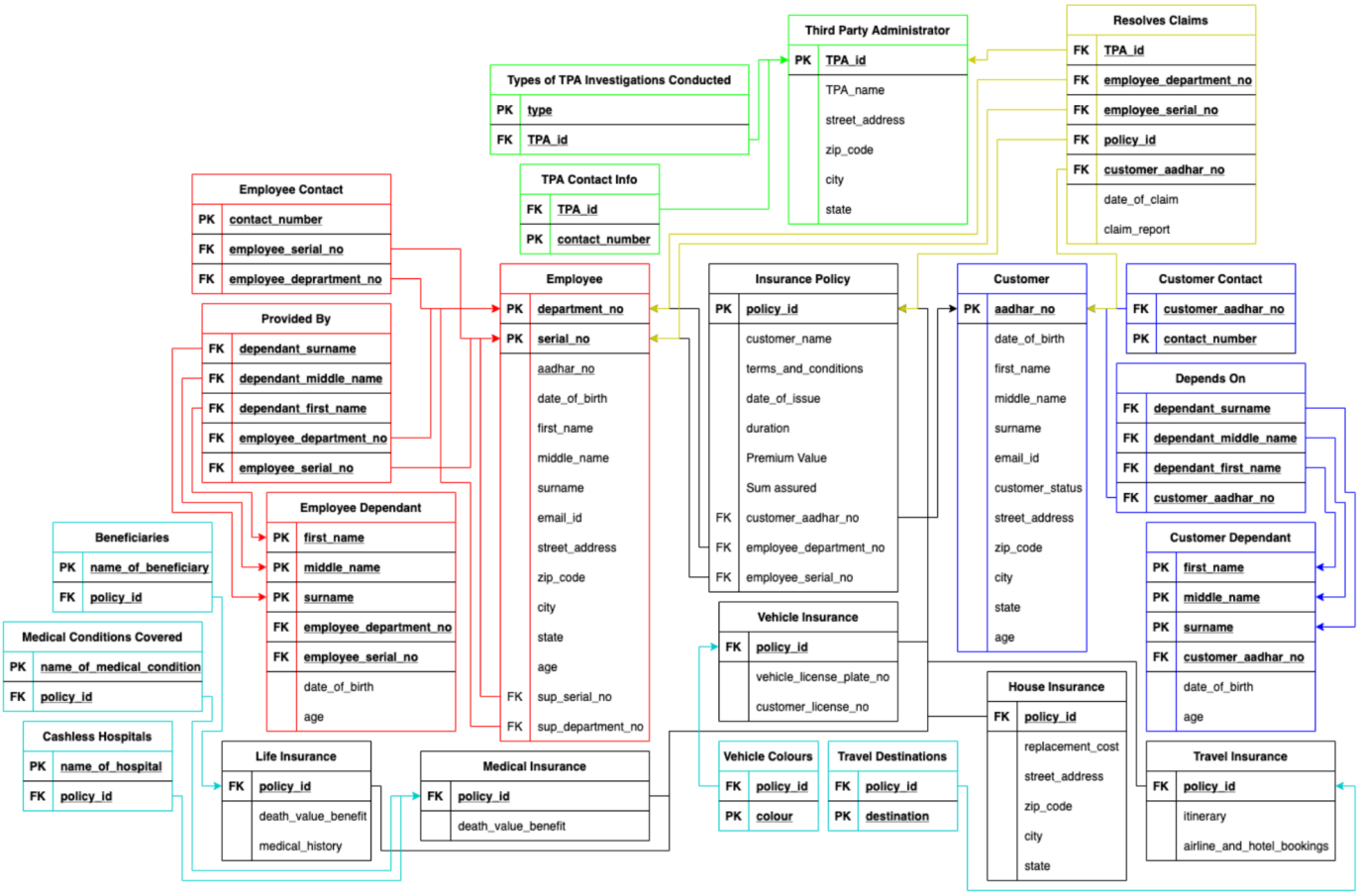

Figure 1.8: Mapping all Superclasses and Subclasses

Task 2: Normal Forms

2.1 1st Normal Form

We have handled all cases of multivalued and composite attributes, and we don't have any nested relations in our model. Hence, our model already is in 1 NF form.

2.2 2nd Normal Form

All the relations that have a single attribute in the primary key do not have an attribute to remove. Hence, these relations are fully functional dependent and can be ignored while checking for partially functional dependent relations.
The only relation that fails the 2 Normal Form test is **Resolves Claims**.
This is because the set of attributes (**claim_report, date_of_claim**) can be functionally determined by the attribute **policy_id** of the primary key.

This relation can be secondly normalized by performing the following steps:
• Delete the attributes **date_of_claim** and **claim_report** from the relation **Resolves Claims.**
• Create a new Relation **Resolves_Claims_Attributes**(policy_id, date_of_claim, claim_report) where the attribute **policy_id** is the primary key and the foreign key.

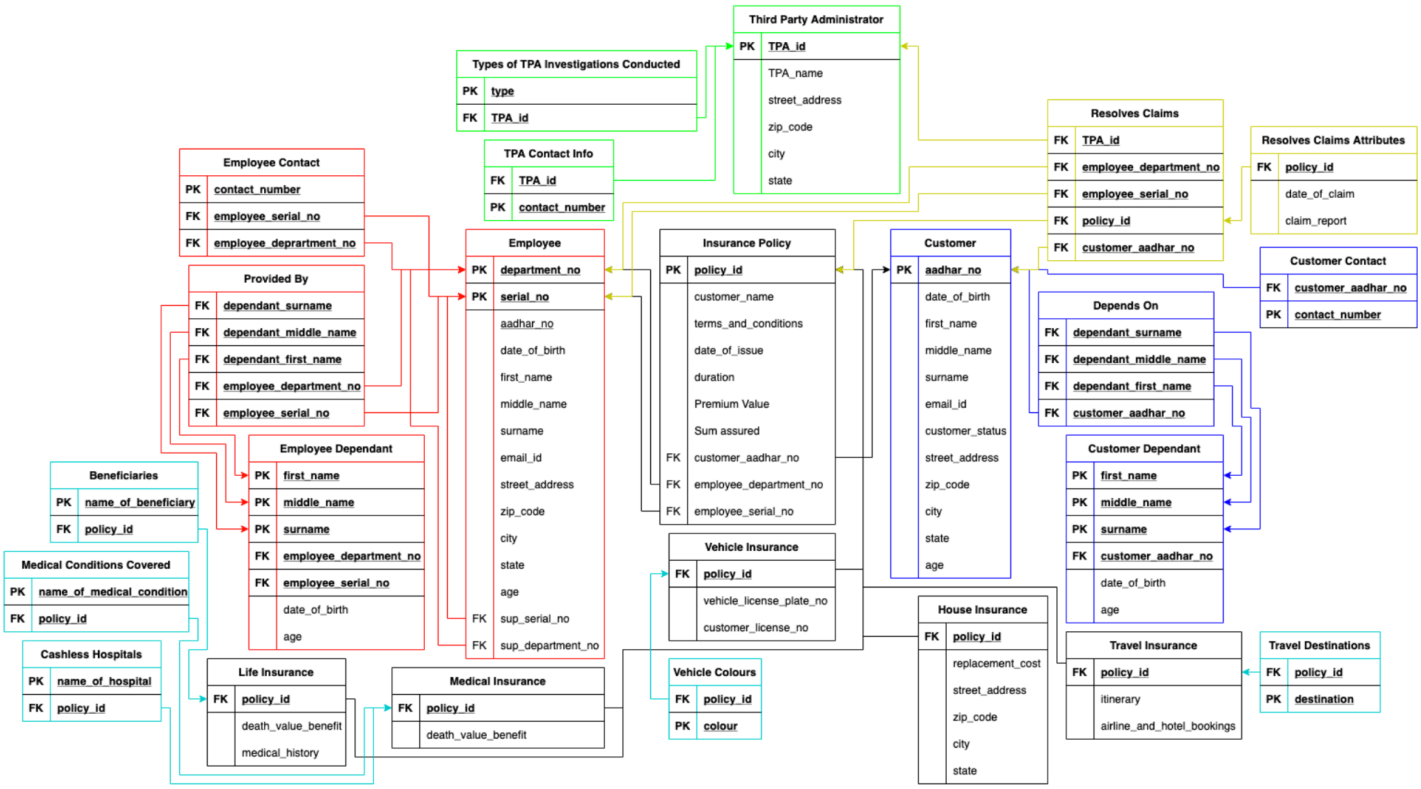

Figure 2.1: Mapping After Second Normalization

2.3 3rd Normal Form

In the relations **Employee, Employee Dependent, Customer** and **Customer Dependent**, the attribute **age** is functionally dependent on the attribute **date_of_birth**, which is in turn functionally dependent on the primary key.
In the relation **Resolves Claims Attributes**, the attribute **claim_report** is functionally dependent on the attribute **date_of_claim**, which is in turn functionally dependent on the primary key.
In relation **Vehicle Insurance**, the attribute **customer_license_info** is functionally dependent on the attribute **vehicle_license_plate_no**, which is in turn functionally dependent on the primary key.

This can be solved by the following steps:

• Delete the attribute **age** from the relation **Employee** and create a new relation **Employee_Age**(age, employee_serial_no, employee_department_no) where all the attributes are the primary key, and the attributes **employee_serial_no** and **employee_department_no** are foreign keys.

• Delete the attribute **age** from the relation **Customer** and create a new relation **Customer_Age**(age, customer_aadhar_no) where all the attributes are the primary key, and the attribute **customer_aadhar_no** is the foreign key.

• Delete the attribute **age** from the relation **Employee Dependent** and create a new relation **Employee_Dependent_Age**(age, dependent_first_name, dependent_middle_name, dependent_surname) where all the attributes are the primary key, and the attributes **dependent_first_name, dependent_middle_name** and **dependent_surname** are foreign keys.

• Delete the attribute **age** from the relation **Customer Dependent** and create a new relation **Customer_Dependent_Age**(age, dependent_first_name, dependent_middle_name, dependent_surname) where all the attributes are the primary key, and the attributes **dependent_first_name, dependent_middle_name** and **dependent_surname** are foreign keys.

• Rename the relation **Resolves Claims Attributes** to **Claim Date**, delete the attribute **claim_report** from this relation and create a new relation **Claim_Report**(policy_id, claim_report) where all the attributes are the primary key, and the attribute **policy_id** is the foreign key.

• Delete the attribute **customer_license_no** from the relation **Vehicle Insurance** and create a new relation **Customer_License_No**(policy_id, customer_license_no) where all the attributes are the primary key, and the attribute **customer_license_no** is the foreign key.