City Planning – Public Transportation

Alpha – Sarat Kiran Andhavarapu

Beta – Shanmukh Yadlapalli

OVERVIEW:

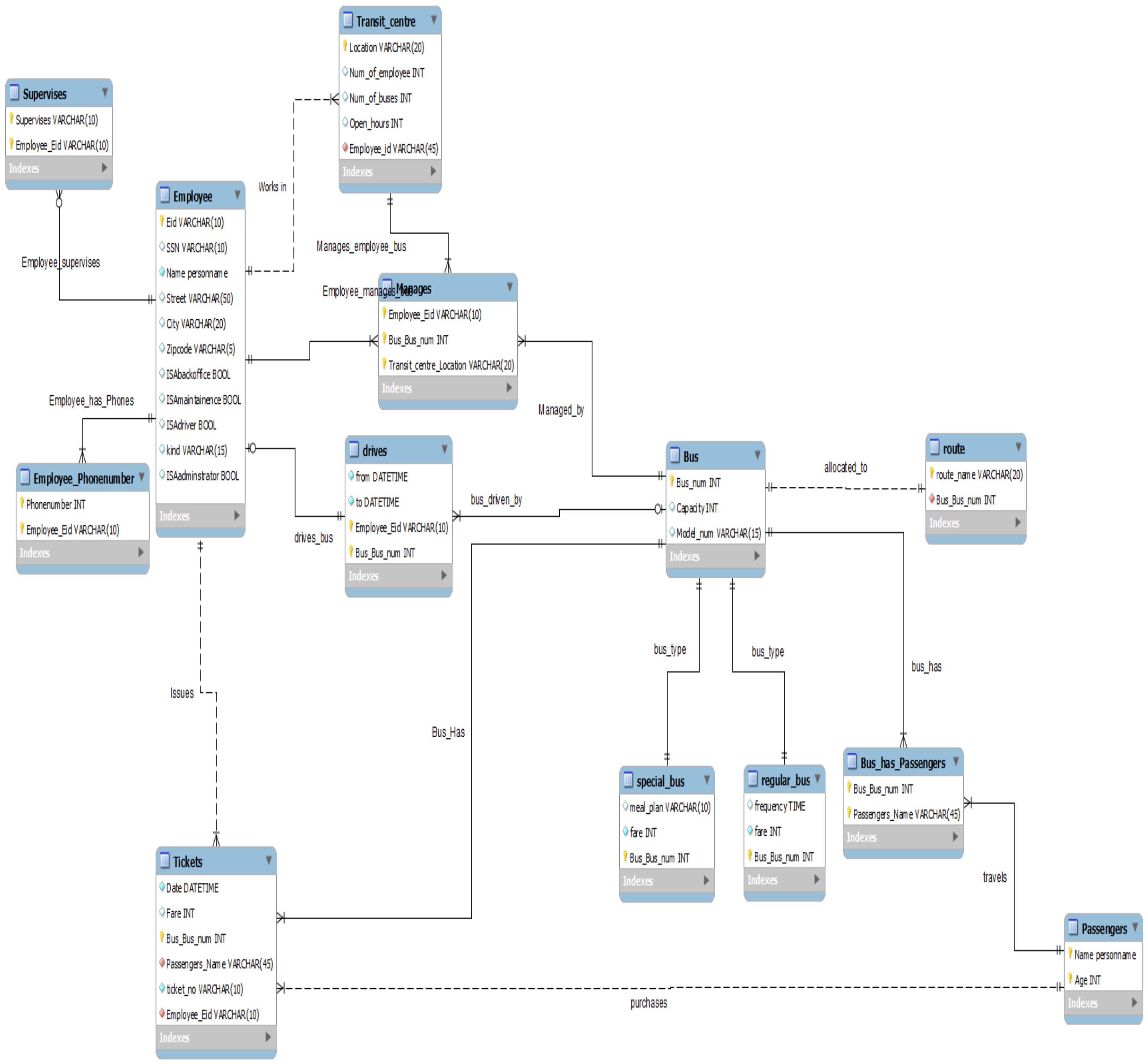
We are trying to describe various things involved in a City public transportation. The different things involved in public transportation are employees, bus, passengers, routes, transit center, tickets etc. These are described as various entities in the model and their characteristics as attributes for the entities. The various relationships between different things in the public transportation system are described using relationship between entities.

In this mini-world, we have some assumptions. Employee's manages different buses and these two are together managed by transit center which also has some employees as administrators. And each bus tickets that are to be purchased by the passengers travelling the bus, all these tickets are issued by the back office employees. Employees are basically three types, they are back office, maintenance, drivers and these three form a disjoint hierarchy of the Employee's. And also an employee can supervise many other employees.

Each bus has a specific route. Buses are also of two type's i.e., special and regular buses depending on fare and other options available in the bus but a same bus can be used as special or regular at different times, this forms an overlapping hierarchy. Each bus may be driven by different drivers based on time i.e., drivers works in shifts system (like from certain time to another). Buses are managed by different employees like a maintenance person to check on bus condition, a back office employee to help the bus to be on route and another employee to keep track of time for a bus etc.

One assumption we make is that no two passengers have the same full name.

ER diagram:


Transit_centre
Location VARCHAR(20)
Num_of_employee INT
Num_of_buses INT
Open_hours INT
Employee_id VARCHAR(45)
Indexes
Supervises
Supervises VARCHAR(10)
Employee_Eid VARCHAR(10)
Indexes
Works in
Employee
Eid VARCHAR(10)
SSN VARCHAR(10)
Name personname
Street VARCHAR(50)
City VARCHAR(20)
Zipcode VARCHAR(5)
ISAbackoffice BOOL
ISAmaintainence BOOL
ISAdriver BOOL
kind VARCHAR(15)
ISAadminstrator BOOL
Indexes
Employee_supervises
Manages_employee_bus
Employee_manages_bus
Manages
Employee_Eid VARCHAR(10)
Bus_Bus_num INT
Transit_centre_Location VARCHAR(20)
Indexes
Employee_has_Phones
Employee_Phonenumber
Phonenumber INT
Employee_Eid VARCHAR(10)
Indexes
Managed_by
drives
from DATETIME
to DATETIME
Employee_Eid VARCHAR(10)
Bus_Bus_num INT
Indexes
bus_driven_by
drives_bus
Bus
Bus_num INT
Capacity INT
Model_num VARCHAR(15)
Indexes
allocated_to
route
route_name VARCHAR(20)
Bus_Bus_num INT
Indexes
bus_type
bus_type
bus_has
Issues
Bus_Has
special_bus
meal_plan VARCHAR(10)
fare INT
Bus_Bus_num INT
Indexes
regular_bus
frequency TIME
fare INT
Bus_Bus_num INT
Indexes
Bus_has_Passengers
Bus_Bus_num INT
Passengers_Name VARCHAR(45)
Indexes
travels
Tickets
Date DATETIME
Fare INT
Bus_Bus_num INT
Passengers_Name VARCHAR(45)
ticket_no VARCHAR(10)
Employee_Eid VARCHAR(10)
Indexes
Passengers
Name personname
Age INT
Indexes
purchases

List of entities:

Employees:

Employee’s entity represents various employees in the public transportation system. Employees are of four types, they are back office, maintenance, Administrators and drivers (disjoint hierarchy). The different attributes of this entity are Eid, name, ssn, phone (a multi-valued attribute) and an address (a composite variable having attributes street, city, zip). Here Eid is the primary key because using that we can uniquely identify each employee and it cannot be null. Other non-null attribute is name.(assumption)

Bus:

Bus entity represents various buses used in the public transportation system. Buses can be special type or regular type or both. The different attributes of this entity are bus_number, capacity and model_no. of the bus. Here bus_num is the primary key because using it we can uniquely identify the bus and it is non-null.

Tickets:

Ticket entity represents tickets that are to be taken by passengers to travel in buses. The different attributes of ticket entity are ticket_no, date and fare. Here **ticket is a weak entity**, so as to uniquely identify each ticket we borrow the bus_num key from the Bus entity and make it as its primary key. Using the bus_num and ticket_no and date, we can uniquely identify each ticket. This entity also has attribute ‘Eid’ from Employee entity as foreign key(borrowed) as to know which back office employee issued the ticket and also attribute ‘name’ from the Passenger entity as foreign key(borrowed) to know which passenger travelled using that ticket. The ticket_no should be non-null. Other non-null attributes are bus_bus_num, Employee_Eid, Passenger_name because they are foreign keys to other entities. Date attribute is also non-null as we need to know which ticket is issued for which date.

Route:

Route entity represents route taken by each different bus. One route is allocated for each bus. The attribute for this entity is route_name, it is also the primary key because using it we can uniquely identify that route. And also it has to be non-null.

Passenger:

Passenger entity represents the passengers travelled in different buses. The different attributes for this entity are Name (passenger's name) and age (special seats are reserved for elderly). Here since we assumed that no two passengers have the same full name, we can consider 'name' as the primary for this entity and so it should be non-null.

Transit center:

Transit center entity represents the center that manages both employees and buses at different locations. The different attributes for this entity are location (of the transit center), no_of_employees (working there) and no_of_buses (operated by them) and open_hours. Here location is the primary key as we can uniquely identify the transit center using the location and also we get attribute 'eid' as foreign key from the employee entity as the people working in transit center are also employees. Location should be non-null.

List of Relationships:

Relationships in ER diagram as tables:

Supervises:

This relationship exists between employee entity and itself. It’s actually a reflexive relationship type. The attribute in this is Supervises (the eid of employees who are being supervised) and we also borrow attribute ‘eid’ from the employee entity (foreign key). Using both we can uniquely identify the supervisor. Both should be non-null.

Drives:

This relationship exists between drivers (a type of employee) and bus entity. The attributes of this relationship are from (time) , to (time) and foreign keys borrowed are ‘eid’ from employee entity and ‘bus_num’ from bus entity. These two foreign keys will be the primary keys. From and to attributes are non-null as each driver works in shift system.

Manages:

This relationship is a ternary relationship in our ER diagram and it links the entities ‘transit center’, ‘employee’, ‘bus’ . It has no specific attributes but the primary keys from each entities mentioned are borrowed by this and all three together make a primary keys. All three attributes should be non-null.

Bus_has_passengers:

Bus_has_passengers is a relationship between bus entity and passenger entity. It has no specific attributes. It borrows keys ‘bus_num’ from bus entity and ‘name’ from passenger entity and these two together are primary entities for this relationship. Both attributes are non-null.

Relationships not as tables in ER diagram:

Allocated to:

This relationship exists between entities Bus and Route, to indicate that unique route is allocated for each bus. It`s a one to one relationship with total participation on both sides.

Bus_Has:

This relationship exists between entities Tickets and Bus, to indicate that each bus has tickets to be purchased by passengers to travel in that bus. This is one to many relationship with many on tickets side.

Works in:

This relationship exist between entities Transit_centre and Employee. It`s a one to many unassociated relationship with many on Transit_centre side. This means each person working in Transit_centre is an employee. And many employees work in transit_centre.

Issues:

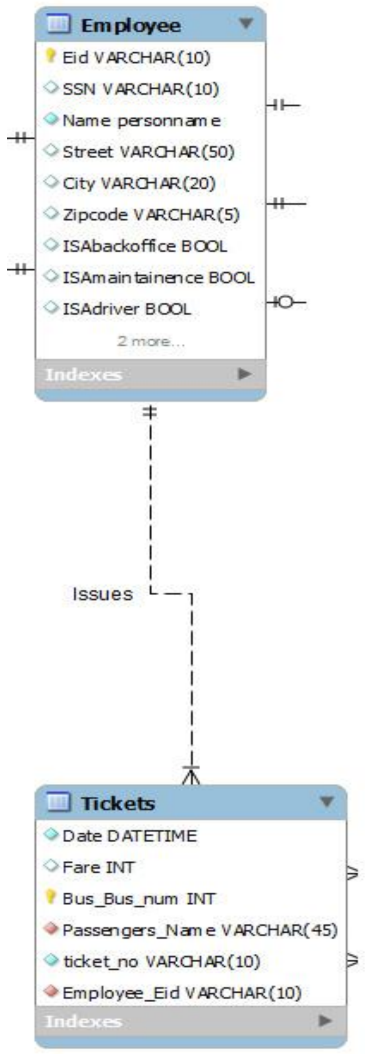
This relationship exists between entities Employees and Tickets. Back office employees issues all tickets. This is a one to many unassociated relationship with many on tickets side.

Purchases:

This relationship exists between entities Passengers and Tickets. This is a one to many unassociated relationship with many on tickets side as a ticket is only for passenger and a passenger can purchase many tickets.

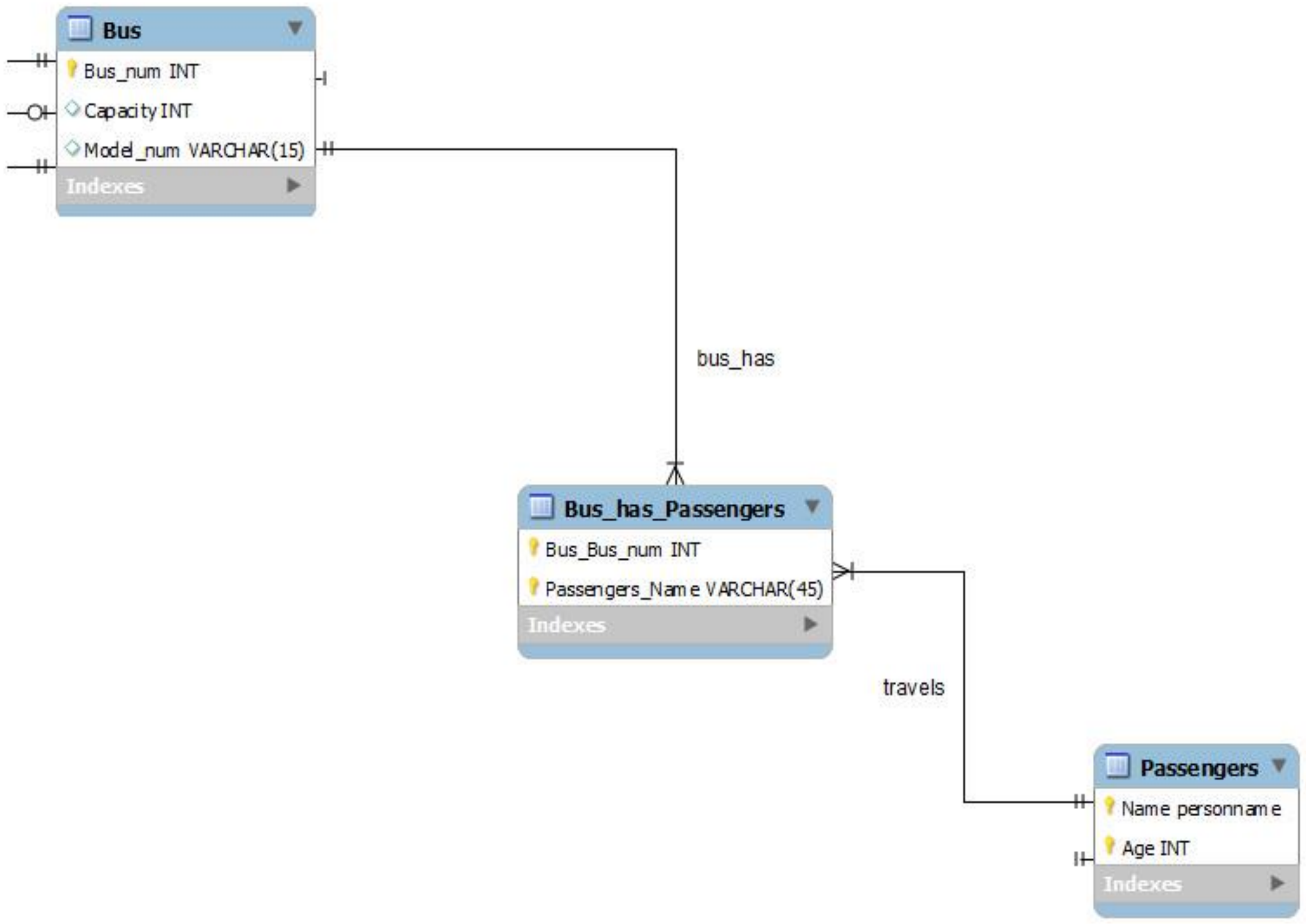
Alpha Contribution:

Many-one relationship type with total participation on the one side:



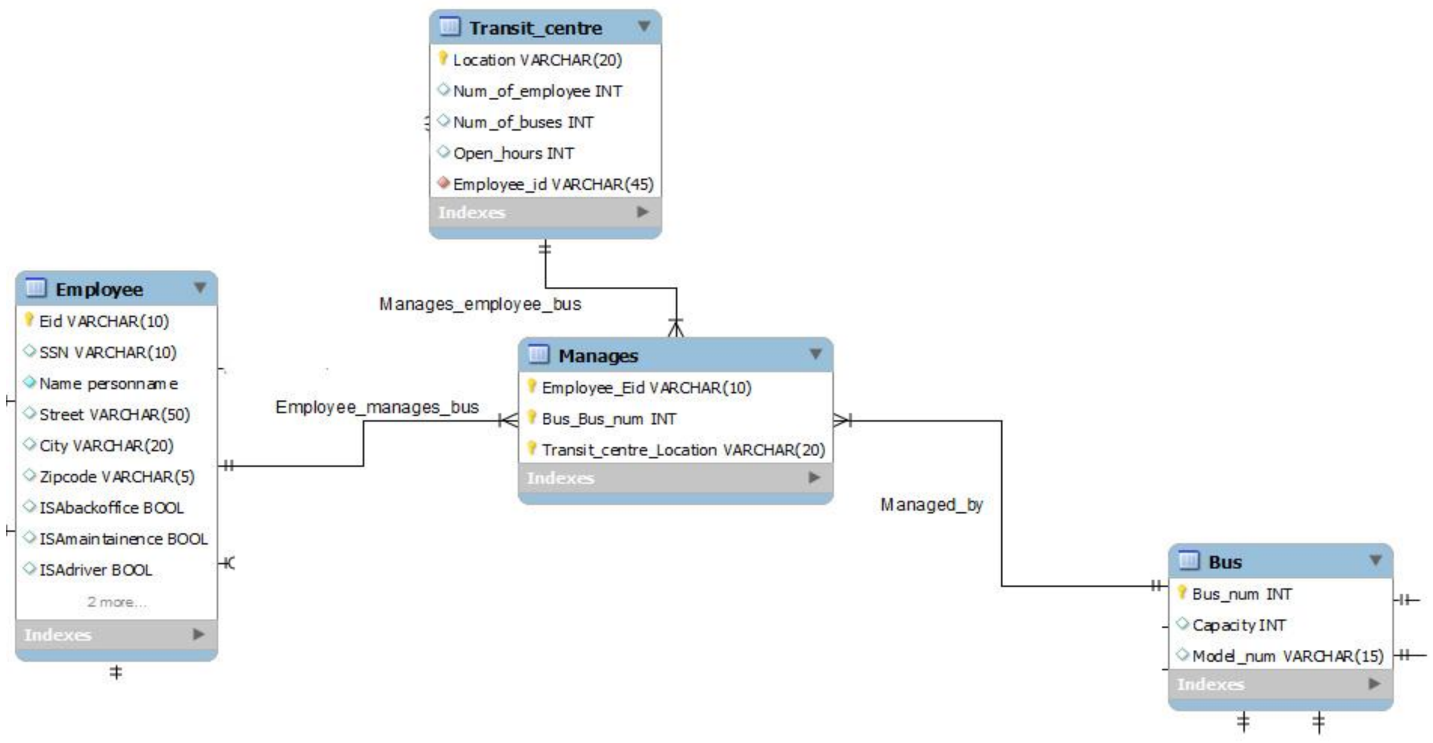
Here Employee entity and ticket entity form many to one relationship with total participation on one side as well as many side. Here employee (back office ,a kind of employee) issues tickets. And tickets (many side) are together issued by the back office always. Therefore there is total participation on both sides. This is mapped using Bus_bus_num which is foreign key to the Bus entity.

Many-many relationship type:



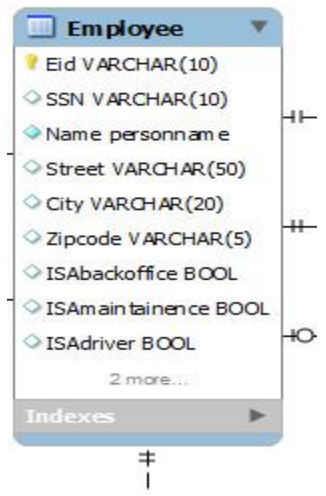
There is a many to many relationship between Passengers entity and Bus entity. Passengers travel in buses. So it means each passenger can travel in many buses and also each bus has many passengers. Therefore Passengers and Bus entities form many to many relationship. This is mapped using Bus_has_Passengers relationship table which is having Bus_bus_num and Passengers_Name as foreign keys to the entities Bus and Passengers respectively.

Ternary relationship:



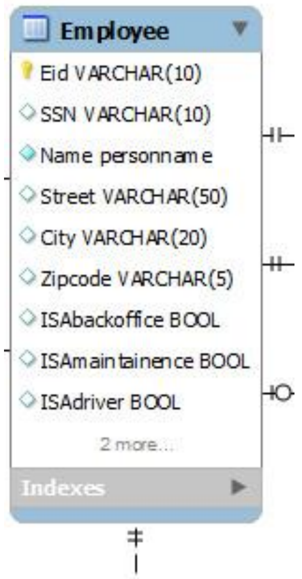
An employee manages many buses and each bus can be managed by many employees (like maintenance, back office etc.). And these both are together managed by Transit_centre. These three entities together form the ternary relationship. This is mapped using manages relationship table with attributes Employee_Eid, Bus_Bus_num, and Transit_centre_location as foreign keys to the entities Employee, Bus and Transit_centre respectively.

Disjoint inheritance hierarchy:



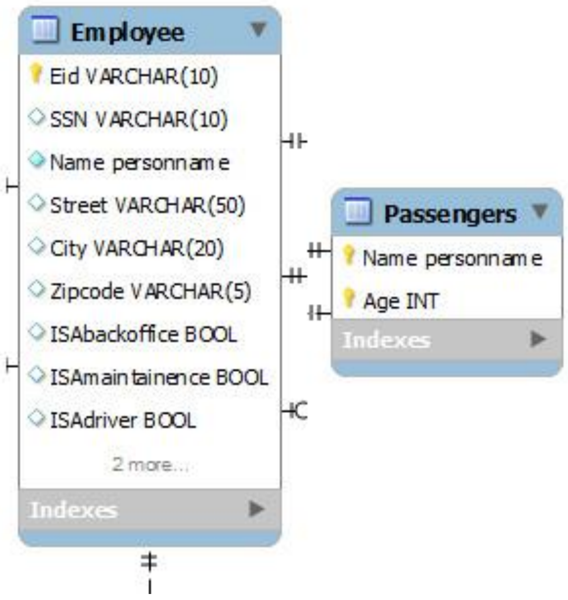
An employee can be of different kinds like Back office, Maintenance, Driver and Administration. So this is a disjoint hierarchy. To map the disjoint hierarchy, sub entities are added to the employee entity to make it a big entity. Here we are adding them as ISA Bool columns.

Composite attribute:



The Employee entity has attribute Address which is a composite attribute which consists of Street, City and Zip code attributes. To map this attribute we replaced the attribute with Street, City and Zip code attributes.

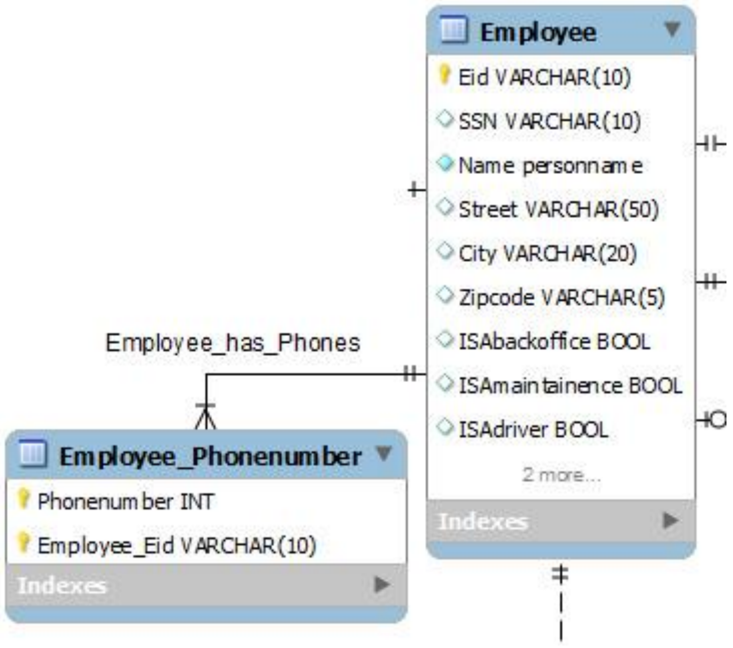
Domain constraint:



In employee entity domain constraint is placed to attribute “Name” as ‘personname’. The domain constraint is the name can be only of varchar type with a maximum of 60 characters.

This domain constraint is also used in Passengers entity.And there are other constraints like we cannot enter “alphabets” in num_of_employees, num_of_buses(since we declared it as INT)

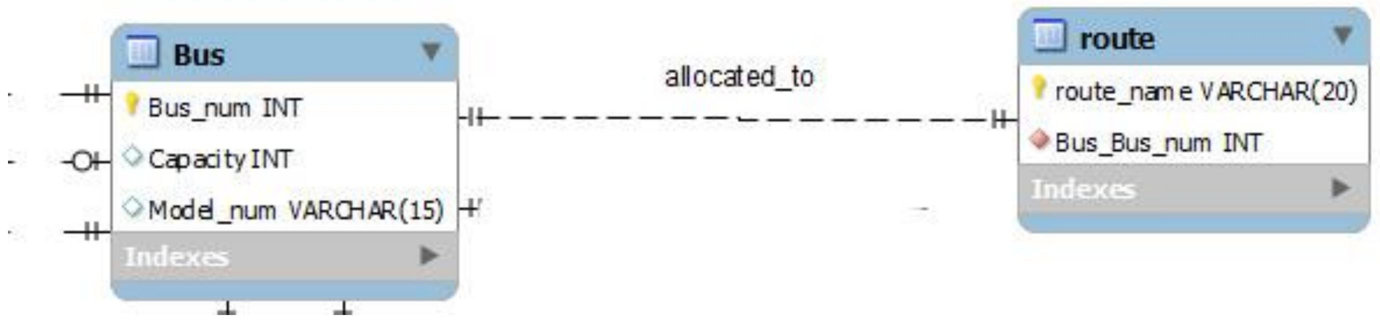
Multivalued attribute:



In employee entity the attribute “phone number” is a multivalued attribute as an employee can have multiple phone numbers. The phone number attribute is mapped using Employee_Phonenumber whose attributes are Phonenumber and Employee_Eid are foreign keys to the entity Employee.

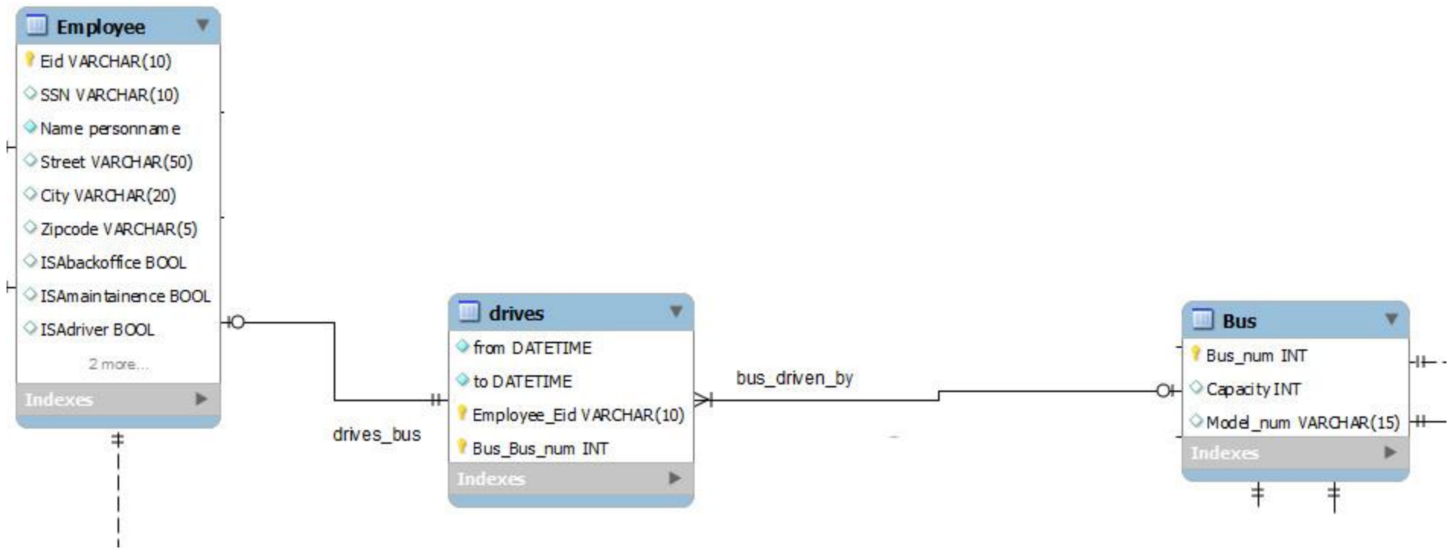
Beta contribution:

One-one relationship type with total participation on at least one side:



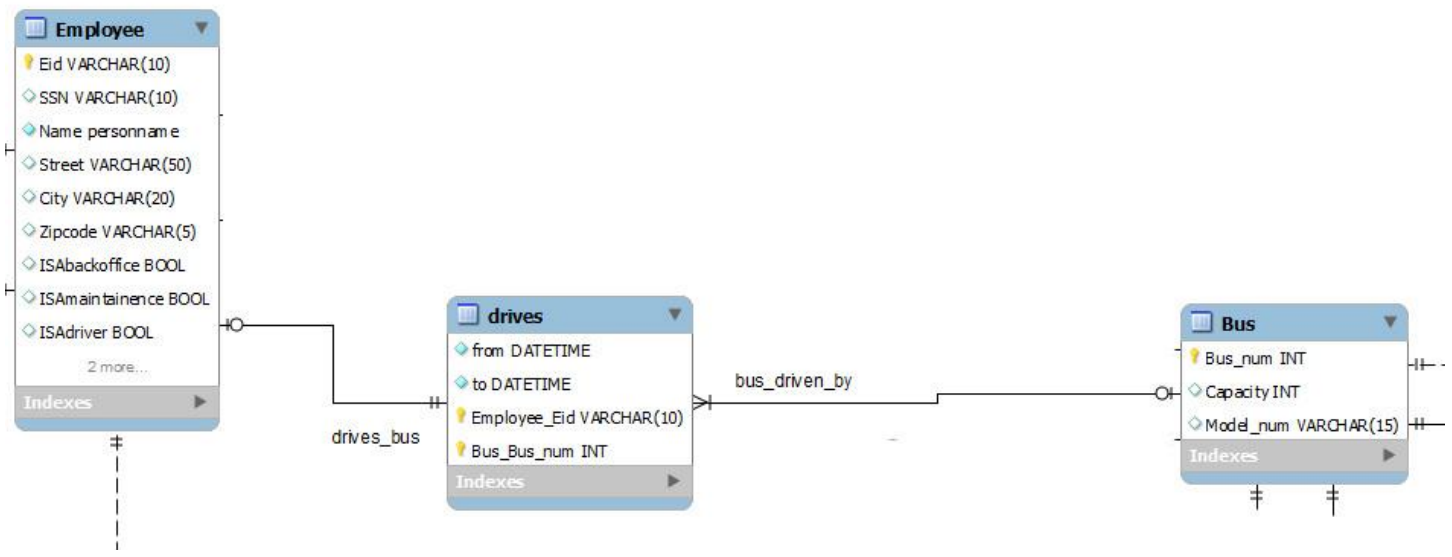
Relationship between entities Bus and Route is one to one relationship with total participation on both sides. Each bus should have unique route and each route is uniquely allocated to one bus only. And so it is one to one relationship with total participation on both sides. The attributes in route entity are route_name which is primary key for that entity and Bus_Bus_name is the foreign key to the entity Bus.

One-many relationship type with partial participation on both sides:



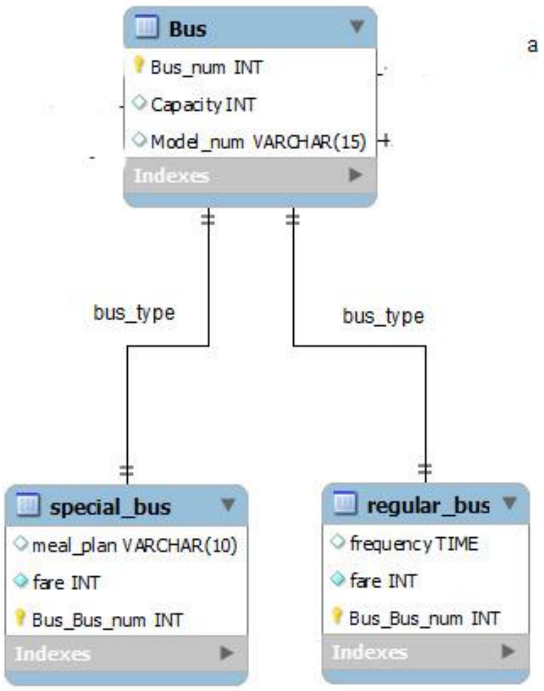
The entities Employee and Bus form a one to many relationship with partial participation on both sides as an Employee (ISAdriver) may drive one bus and each bus may be driven by many employees. So there is partial participation on both sides. This relationship is mapped using drives relationship table which has attributes “from”, “to” and Employee_Eid and Bus_Bus_num as foreign keys to the entities Employee and Bus respectively.

Relationship with an attribute(s):



Bus is driven by many employees and an employee (ISAdriver) can drive a bus. Therefore these entities form one to many relationship. Drives relationship table has attributes because an employee (ISAdriver) can drive a bus for only a period of time and so the attributes are “from” and “to” for this relationship. This can be mapped using those attributes.

Overlapping inheritance hierarchy:

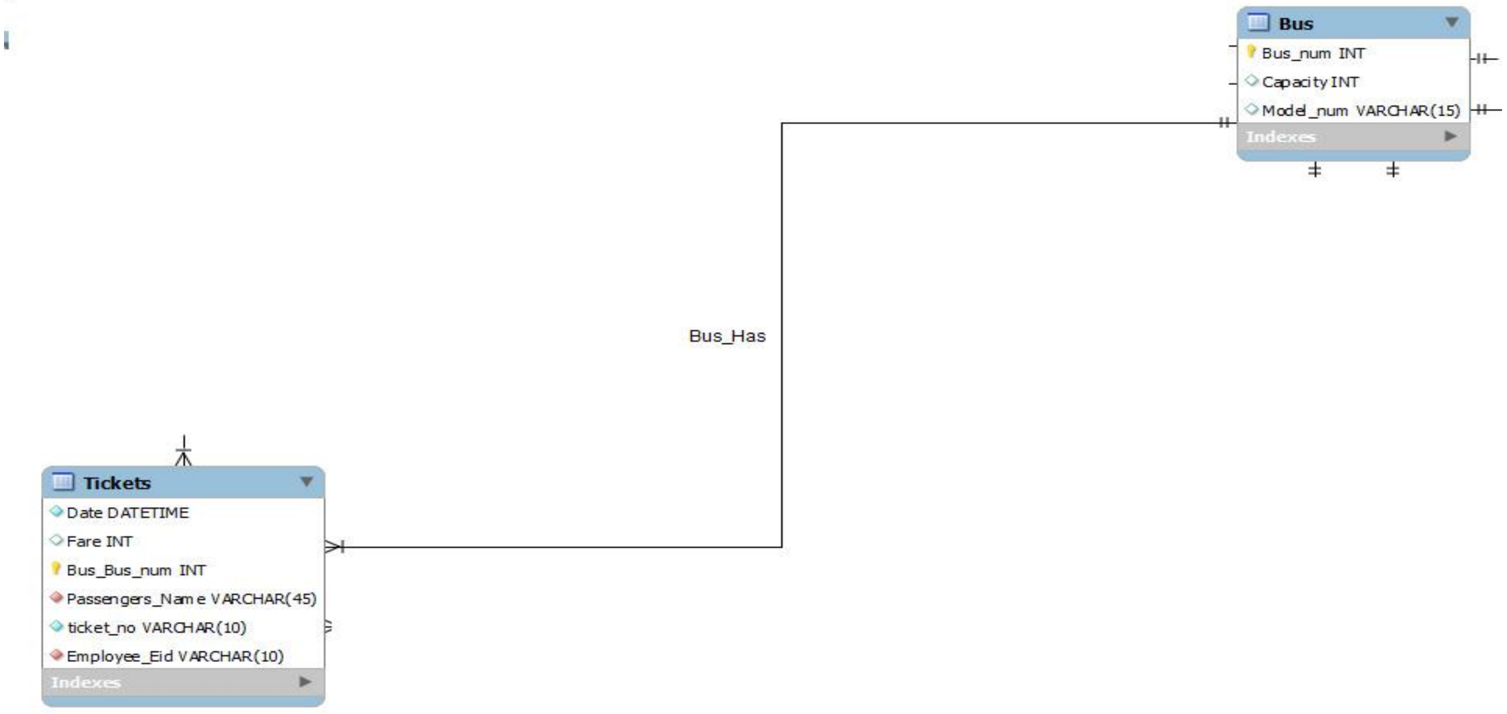


A bus can be either one of the type or both special and regular bus. Therefore they form overlapping hierarchy. To map this we created two different entities special_bus and regular_bus. These two entities use primary key of Bus entity as their primary keys i.e. Bus_num. Bus_bus_num and fare in both entities are non-null.

Weak-entity type:

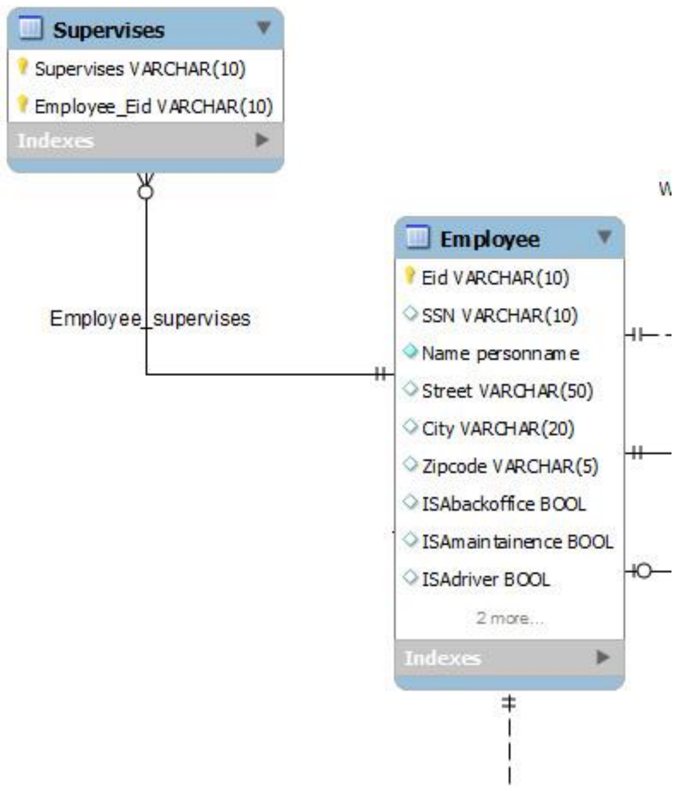
-

-



Bus has tickets issued by back office. The attributes of tickets entity are ticket_no, fare, date and three other attributes Bus_Bus_num, Passengers_name and Employee_Eid are foreign keys to the entities Bus, Passengers and Employee respectively. But the borrowed key Bus_Bus_num from Bus entity is the primary key for the Tickets entity as we cannot uniquely identify tickets without the borrowed keys. We can uniquely identify the Ticket entity using ticket_no,date and Bus_Bus_num attributes. So tickets is a weak entity.

Reflexive relationship type:



Many of the employees may be supervisors, who supervises other employees. This forms a reflexive relationship type. This is mapped using supervises table which has attributes Supervises and Employee_Eid. Both constitute the primary key for the relationship table and both are foreign keys for the entity Employee.

Check (integrity constraint on an attribute):

The capacity attribute of Bus entity cannot extend 80. So check is placed on that attribute. So there is integrity constraint on capacity attribute of Bus entity.

Physical Schema:

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`Employee`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Employee` (
  `Eid` VARCHAR(10) NOT NULL,
  `SSN` VARCHAR(10) NULL,
  `Name` VARCHAR(60) NOT NULL,
  `Street` VARCHAR(50) NULL,
  `City` VARCHAR(20) NULL,
  `Zipcode` VARCHAR(5) NULL,
  `ISAbackoffice` TINYINT(1) NULL,
  `ISAmaintainence` TINYINT(1) NULL,
  `ISAdriver` TINYINT(1) NULL,
  `kind` VARCHAR(15) NULL,
  `ISAadminstrator` TINYINT(1) NULL,
  PRIMARY KEY (`Eid`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Employee_Phonenumber`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Employee_Phonenumber` (
  `Phonenumber` INT NOT NULL,
  `Employee_Eid` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`Phonenumber`, `Employee_Eid`),
  INDEX `fk_Phonenumber_Employee_idx` (`Employee_Eid` ASC),
  CONSTRAINT `fk_Phonenumber_Employee`
    FOREIGN KEY (`Employee_Eid`)
    REFERENCES `mydb`.`Employee` (`Eid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Transit_centre`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Transit_centre` (
  `Location` VARCHAR(20) NOT NULL,
  `Num_of_employee` INT NULL,
  `Num_of_buses` INT NULL,
  `Open_hours` INT NULL,
  `Employee_id` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Location`),
  INDEX `fk_Transit_centre_1_idx` (`Employee_id` ASC),
  CONSTRAINT `fk_Transit_centre_1`
    FOREIGN KEY (`Employee_id`)
    REFERENCES `mydb`.`Employee` (`Eid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Bus`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Bus` (
  `Bus_num` INT NOT NULL,
  `Capacity` INT NULL,
  `Model_num` VARCHAR(15) NULL,
  PRIMARY KEY (`Bus_num`))
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`Supervises`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Supervises` (
  `Supervises` VARCHAR(10) NOT NULL,
  `Employee_Eid` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`Supervises`, `Employee_Eid`),
  CONSTRAINT `fk_Supervisor_Employee1`
    FOREIGN KEY (`Supervises` , `Employee_Eid`)
    REFERENCES `mydb`.`Employee` (`Eid` , `Eid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`Passengers`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Passengers` (
  `Name` VARCHAR(60) NOT NULL,
  `Age` INT NOT NULL,
  PRIMARY KEY (`Name`, `Age`))
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`Tickets`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Tickets` (

  `Date` DATETIME NOT NULL,

  `Fare` INT NULL,

  `Bus_Bus_num` INT NOT NULL,

  `Passengers_Name` VARCHAR(45) NOT NULL,

  `ticket_no` VARCHAR(10) NOT NULL,

  `Employee_Eid` VARCHAR(10) NOT NULL,

  PRIMARY KEY (`Bus_Bus_num`),

  INDEX `fk_Tickets_Passengers1_idx` (`Passengers_Name` ASC),

  INDEX `fk_Tickets_Employee1_idx` (`Employee_Eid` ASC),

  CONSTRAINT `fk_Tickets_Bus1`

    FOREIGN KEY (`Bus_Bus_num`)

    REFERENCES `mydb`.`Bus` (`Bus_num`)

    ON DELETE NO ACTION

    ON UPDATE NO ACTION,

  CONSTRAINT `fk_Tickets_Passengers1`

    FOREIGN KEY (`Passengers_Name`)

    REFERENCES `mydb`.`Passengers` (`Name`)

    ON DELETE NO ACTION

    ON UPDATE NO ACTION,

  CONSTRAINT `fk_Tickets_Employee1`

    FOREIGN KEY (`Employee_Eid`)

    REFERENCES `mydb`.`Employee` (`Eid`)

    ON DELETE NO ACTION

    ON UPDATE NO ACTION)

ENGINE = InnoDB;



-- -----------------------------------------------------

-- Table `mydb`.`Manages`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Manages` (
  `Employee_Eid` VARCHAR(10) NOT NULL,
  `Bus_Bus_num` INT NOT NULL,
  `Transit_centre_Location` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`Employee_Eid`, `Bus_Bus_num`, `Transit_centre_Location`),
  INDEX `fk_Manages_Bus1_idx` (`Bus_Bus_num` ASC),
  INDEX `fk_Manages_Transit_centre1_idx` (`Transit_centre_Location` ASC),
  CONSTRAINT `fk_Manages_Employee1`
    FOREIGN KEY (`Employee_Eid`)
    REFERENCES `mydb`.`Employee` (`Eid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Manages_Bus1`
    FOREIGN KEY (`Bus_Bus_num`)
    REFERENCES `mydb`.`Bus` (`Bus_num`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Manages_Transit_centre1`
    FOREIGN KEY (`Transit_centre_Location`)
    REFERENCES `mydb`.`Transit_centre` (`Location`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`route`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`route` (
  `route_name` VARCHAR(20) NOT NULL,
  `Bus_Bus_num` INT NOT NULL,
  PRIMARY KEY (`route_name`),
  INDEX `fk_route_Bus1_idx` (`Bus_Bus_num` ASC),
  CONSTRAINT `fk_route_Bus1`
    FOREIGN KEY (`Bus_Bus_num`)
    REFERENCES `mydb`.`Bus` (`Bus_num`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`table1`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`table1` (
)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`drives`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`drives` (
  `from` DATETIME NOT NULL,
  `to` DATETIME NOT NULL,
  `Employee_Eid` VARCHAR(10) NULL,
  `Bus_Bus_num` INT NULL,
  PRIMARY KEY (`Employee_Eid`, `Bus_Bus_num`),
  INDEX `fk_drives_Bus1_idx` (`Bus_Bus_num` ASC),
  CONSTRAINT `fk_drives_Employee1`
    FOREIGN KEY (`Employee_Eid`)
    REFERENCES `mydb`.`Employee` (`Eid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_drives_Bus1`
    FOREIGN KEY (`Bus_Bus_num`)
    REFERENCES `mydb`.`Bus` (`Bus_num`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`special_bus`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`special_bus` (
  `meal_plan` VARCHAR(10) NULL,
  `fare` INT NOT NULL,
  `Bus_Bus_num` INT NOT NULL,
  PRIMARY KEY (`Bus_Bus_num`),
  CONSTRAINT `fk_special_bus_Bus1`
    FOREIGN KEY (`Bus_Bus_num`)
    REFERENCES `mydb`.`Bus` (`Bus_num`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`regular_bus`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`regular_bus` (
  `frequency` TIME NULL,
  `fare` INT NOT NULL,
  `Bus_Bus_num` INT NOT NULL,
  PRIMARY KEY (`Bus_Bus_num`),
  CONSTRAINT `fk_regular_bus_Bus1`
    FOREIGN KEY (`Bus_Bus_num`)
    REFERENCES `mydb`.`Bus` (`Bus_num`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`Bus_has_Passengers`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Bus_has_Passengers` (
  `Bus_Bus_num` INT NOT NULL,
  `Passengers_Name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Bus_Bus_num`, `Passengers_Name`),
  INDEX `fk_Bus_has_Passengers_Passengers1_idx` (`Passengers_Name` ASC),
  INDEX `fk_Bus_has_Passengers_Bus1_idx` (`Bus_Bus_num` ASC),
  CONSTRAINT `fk_Bus_has_Passengers_Bus1`
    FOREIGN KEY (`Bus_Bus_num`)
    REFERENCES `mydb`.`Bus` (`Bus_num`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Bus_has_Passengers_Passengers1`
    FOREIGN KEY (`Passengers_Name`)
    REFERENCES `mydb`.`Passengers` (`Name`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```