**HO CHI MINH UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY FOR HIGH-QUALITY TRAINING**

**COURSE NAME: Database Management System**

**🙡🙠✵🙢🙣**



**FINAL PROJECT REPORT**

**Project name:**

BUS TICKET BOOKING MANAGEMENT SYSTEM

**Lecturer:** Prof. Nguyễn Thành Sơn

**Course ID:** DBMS330284E\_22\_2\_01FIE

**Group:** 6

**Date:** 2nd Sem/2022-2023

*Ho Chi Minh city, May, 2023*

**LIST OF STUDENTS – GROUP 6**

**Project: Bus ticket booking management system**

| ***ID*** | ***Full name*** |
| --- | --- |
| 21110758 | Lê Xuân Cường |
| 21110092 | Bùi Quốc Thông |
| 21110066 | Phạm Vũ Bảo Nhân |
| 21110785 | Mai Nguyễn Nhật Nam |

**Professor’s comment**

Ho Chi Minh city, May …, 2023

Grading

**Contents**

[Prologue 4](#_heading=h.z337ya)

[INTRODUCTION 5](#_heading=h.3j2qqm3)

[CHAPTER I: SYSTEM OVERVIEW 6](#_heading=h.4i7ojhp)

[1.](#_heading=h.2xcytpi) Specifications 6

[1.1.](#_heading=h.1ci93xb) Problem statement 6

[1.2.](#_heading=h.3whwml4) Overview 6

[2.](#_heading=h.2bn6wsx) Problem process 9

[2.1.](#_heading=h.qsh70q) Booking period 9

[2.2.](#_heading=h.3as4poj) Departure period 9

[2.3.](#_heading=h.1pxezwc) Drop-off period 10

[2.4.](#_heading=h.49x2ik5) Ticket cancellation period 10

[2.5.](#_heading=h.2p2csry) Delivery period 10

[3.](#_heading=h.nmf14n) Main functions 10

[4.](#_heading=h.37m2jsg) Attributes and operations reference 11

[CHAPTER 2: SYSTEM ANALYSIS AND DESIGN 19](#_heading=h.1mrcu09)

[1.](#_heading=h.46r0co2) Conceptual level database design 19

[2.](#_heading=h.sqyw64) Logical level database design 19

[3.](#_heading=h.3cqmetx) Required constraints 21

[4.](#_heading=h.1rvwp1q) Database settings and constraints 25

[5.](#_heading=h.4bvk7pj) Other constraints 28

[6.](#_heading=h.2r0uhxc) Trigger to check for constraints 30

[CHAPTER 3: DESIGNING FUNCTIONS 45](#_heading=h.1664s55)

[1.](#_heading=h.3q5sasy) Connect to database 45

# Prologue

Firstly, we would like to express our gratitude to Prof. Nguyễn Thành Sơn for his whole-hearted instructions that helped us finish our final project for the Database Management System course. Thanks to the knowledge the professor has provided us, we were able to firmly grasp the basic knowledge and foundation for building a database management system. And through this project, our group would like to present the development process of a database management system and demonstrate by programming a related project once again.

During the process of executing this project, it will be hard to avoid mistakes. Because of that, we would love to get the professor’s suggestion on improving our work so it would be more functional and complete. We wish you good health and the best of luck pursuing the path of teaching.

Finally, we would like to thank all the teachers and classmates who studied with us on this course and offered us support while we carried out our final project.

# INTRODUCTION

In recent years, the Information and Technology (IT) area has been integrated into our society and daily lives, regardless of any field and/or occupations. It also plays an important part of booking management in Vietnam and especially in almost every country as there are many applications made to help fix problems that big organizations frequently face.

The creation of the bus ticket booking management system is the result of many developers’ creativity and hard work with the aim of aiding companies in managing their businesses.

With that in mind, to better understand the application and role of Information and Technology (IT) in Database Management, we have decided on the **“Bus ticket booking management system”** as our final project.

# CHAPTER I: SYSTEM OVERVIEW

## Specifications

### Problem statement

The bus ticket booking management system will:

* Manage the employees, passengers, bus, trips, routes easier.
* Convenient for users to check and book trips.
* Check the state and location of the trip more clearly through a map.
* More convenient for the bus company to obtain statistics: revenue, number of passengers, number of trips, employee salary, outcome, etc. per day, per month, per year.

Vehicle management: Manage travel vehicles including their location, date and time of arrival/departure, price, etc.

System management: Manage employees, drivers, customers, travel curriculum.

Statistics: Employee statistics, vehicle statistics, daily sales, etc.

### Overview

A bus company needs to have a bus ticket reservation system. The bus ticket reservation system should contain the following data:

The bus company manages a lot of agents. Each agent has: agent ID, cash reserve ID, address, agent name.

Each agent has only one cash reserve. A cash reserve includes cash reserve ID and counter.

An agent has many employees. Each employee has: employee ID, position ID, account ID, agent ID, name, address, phone number, identity number, salary, email, date of birth. Each employee is provided with an account to access into the system (username and password). Each employee type has a different position.

The information of the position group contains: position ID, type.

There are several types:

* + Administrator
  + Travel planner
  + Travel supervisor
  + Driver
  + Ticket seller
  + Service guide
  + Security guard
  + Porter

Each position group has separate privileges. The information of the privileges group includes: privilege ID, name.

The agent manages passengers. Each passenger has: passenger ID, name, phone number, address, identity number, gender, email.

The gender attribute of passenger above has two options:

* Male
* Female

Easily manage and filter the address of stations in the general local area, there is information of places: place ID, region.

Each passenger can choose a pickup station and drop-off station. Each station has: station ID, detailed address, name, capacity, parked bus number.

The bus of each brand has: bus ID, registration number, model, capacity, status, type.

Status of the bus can be:

* Ongoing
* Idle
* Break
* Incident

Type of the bus can be:

* Interprovince
* Transit

Routes involving the journey have: route ID, start bus station ID, final bus station ID, travel distance.

Each trip is set up by the travel planner which includes: trip ID, drivers ID, bus ID, route ID, departure time, duration, number of booked seats, state.

The state attribute of trip above has three options:

* Waiting
* Going
* Finish

The drivers ID in the trip relation is an attribute of TRIP\_DRIVER relation: trip ID, driver ID. Note that driver ID is a multivalued attribute.

The agent distributes tickets to the passenger. Each ticket has: ticket ID, trip ID, passenger ID, status, fare, type, seat number.

The status of the ticket can be:

* Available
* Bought

The type of ticket has two options:

* Seat ticket
* Sleeper ticket

The agent manages the booking transaction. Each booking transaction includes: transaction ID, ticket ID, passenger ID, employee ID, booking time.

Each driver has an employee ID number, license level and type of driver (long-haul driver and transit driver).

Each employee can take on more than 1 position.

Each passenger can book more than 1 ticket.

Each trip can have more than 1 driver.

The bus company provides a delivery service so that the customers can send a package without booking a ticket. They must provide information about their packages such as: mass, the phone number of sender and receiver. This package will have an ID and price. The package’s price is determined by PACKAGE\_PRICE\_POLICY: ID, mass of package and price\_per\_km.

When a big event happens, the bus company has discount periods to lower the price of tickets.

Besides, the refund policy can help the passengers receive part of the fare when they cancel their trip and tickets.

## Problem process

### Booking period

*\* Offline booking:*

The service guide records the passenger’s full field information including: their name, ID number, phone number, address, gender, email. Then, the ticket seller checks again to guarantee all the required fields are correctly fielded.

Then, the passenger picks a trip by choosing from multiple options: destination, pickup station, drop-off station, departure time, the available seat, ticket type. Options will be planned by the travel planner, so the passenger must follow this template.

Then, the ticket seller verifies the customer’s selection. If valid, the ticket seller informs the passenger and waits for their confirmation. If they confirm, the ticket seller prints the ticket, gives it to them and reminds them to arrive at the correct time on the ticket. Else if they refuse, the customer needs to modify the information.

*\* Online booking:*

First of all, the passengers must have an account to access the bus ticket booking application. If they don’t have an account yet, they have to register and log in to book the ticket. If they have an account, they only need to log in to book.

Afterwards, passengers will access the system to book their ticket. They will fill in the information about their name, ID number, phone number, address, gender, email, destination, pickup station, drop-off station, departure time, the available seat, ticket type. The system will send a verification code through email, then passengers fill in the app to verify their booking action.

Next, the system will provide information about the ticket.

### Departure period

The passengers wait for the agent. 15 minutes before the departure time of the trip, the vehicle will take the passengers to the bus station.

At the bus station, the porter put the passengers’ luggage into the trunk.

When it’s time, the service guide instructs passengers to the vehicle, and provides water and tissues to them.

### Drop-off period

When the bus arrives at the last bus station, the porter takes passengers’ luggage from the bus and gives it to the passenger.

### Ticket cancellation period

### Delivery period

## Main functions

Administrator (global):

* Add, modify, delete, authorize for positions
* Add, modify, delete employee of the position
* Statistic information about trip, the number of sold tickets

Travel planner:

* Add, modify, delete trips
* Add, modify, delete routes
* Add (distribute the tickets of the trip), modify, delete tickets

Travel supervisor:

* Add, delete passengers of the trip
* Report errors (trip, route, passenger, booking)

Ticket-selling:

* Add, modify, delete passenger
* Export bill
* Export ticket
* Change the state attribute of trips

Passenger (when booking online):

* Check price ticket of each route
* Check the information about booked tickets
* Book one or many tickets
* Change the information about ticket (information of passenger, the route, departure time, departure date)
* Cancel their tickets
* Export their tickets

\* Authorization:

* Admin: Full control on the whole system – Global privilege
* […] (Other privileges): Local privilege

## Attributes and operations reference

Bus relation:

\* ID attribute:

*Format:*



[ ] : Ignore this notation

Example: **bu5272f29s28\_5**

* model: 5272f29s28
* number: 5

Note:

* model: Model of the bus
* number: Order of the bus

\* Capacity attribute of bus relation is fixed (default of model).

Trip relation:

\* ID attribute:

*Format:*



[ ] : Ignore this notation

Example: **trr78\_20230918**

* ID\_route: r78
* departure\_time: 2023/09/18

Note:

* ID\_route: ID of route in route relation
* departure\_time: Written in continuous form (No special characters)

\* Duration attribute is calculated using the following formula:



Note:

* distance: The distance attribute of route relation
* average\_speed: The moving value is statistic by system

\* booked\_seat attribute: Updated after every insert or delete statement of the passenger of the trip. It must satisfy the following rule:



Note:

* capacity: Get from the bus relation
* inherent\_seat: This value is set by the planner or ticket seller

Ticket relation:

\* ID attribute:

*Format:*



[ ] : Ignore this notation

Example: **tiseat\_a14**

* type: seat
* seat\_number: a14

Note:

* type: Ticket type
* seat\_number: Number of tickets

\* Fare attribute: Set by the travel planner.

\* Seat number attribute has a limit: 0 < seat\_number < 15

*Format:*



Note:

* A: Floor 1
* B: Floor 2
* C: Rear seats
* 1, 2, …, 5: column 1 (after driver’s seat)
* 6, 7, …,10: column 2
* 11, 12, …, 15: column 3
* For C (rear seat): 1,2, …, 5 (left to right)

\* Status attribute:

* Available
* Bought

\* Type attribute:

* Seat
* Bed

Agent relation:

\* ID attribute:

*Format:*



Employee relation:

\* ID attribute:

*Format:*



\* Salary attribute: Unit in (đồng/VND)

\* Birthdate:

*Format:*



Driver relation:

\* License\_level attribute:

*List of levels:*

* A1
* A2
* B1
* B2
* … (To be updated)

\* Type attribute:

*Available driver types:*

* Transit
* Interprovince

Route relation:

\* ID attribute:

*Format:*



[ ] : Ignore this notation

Example: **r78**

* route\_number: 78

Note:

* route\_number: The route number

\* Distance attribute: Set manually by the system

Booking relation:

\* ID attribute:

*Format:*



[ ] : Ignore this notation

Example: **b20230719**

* booking\_time: 2023/07/19

Note:

* booking\_time: The booking time attribute of this relation. Written in continuous form (No special characters)

\* Booking\_time attribute: Written in GMT+7 format.

Place relation:

\* ID attribute: ID of province

\* Region attribute: Name of province

BusStation relation:

\* Cur\_bus\_number attribute: Updated after the related trip state change.

\* Capacity attribute: Maximum capacity of the station (For buses)

Passenger relation:

\* ID attribute:

*Format:*



\* Phone\_number attribute: 10-11-digit length.

\* Gender attribute: Male/Female

SystemAccount relation:

\* Username attribute:

*Format:*



[ ] : Ignore this notation

\* Password attribute:

*Conditions:*

* At least 6 characters in length
* All characters must not be the same

Position relation:

\* ID attribute:

*Format:*



\* Type attribute:

*List of account types:*

* admin: Administrator
* plan: Travel planner
* visor: Travel supervisor
* driver: Driver
* seller: Ticket seller
* guide: Service guide
* guard: Security guard
* porter: Porter

Privilege relation:

\* Name attribute:

*List of names:*

* Add passenger
* Add trip
* Add route
* Add bus
* Add ticket
* All control
* … (To be updated)

CashReserve relation:

\* Counter: Updated after a transaction (booking) occurs.

TripDriver relation:

\* ID\_driver attribute references the ID\_employee of employee relation where position is driver.

Package relation:

\*ID attribute:

*Format:*



Example: **packa1231bs**

* ID\_package: a1231bs

PackagePricePolicy relation:

\* ID attribute:

*Format:*



Example: **pacp12a32**

* ID\_policy: 12a32

\* Price\_per\_km attribute: Set by system

\* Mass attribute: Set by system

Event relation:

\* Discount\_type attribute:

*Types of discounts:*

* Special
* Normal
* Sale (Can be inserted, deleted, updated in some cases)

\* Discount\_percent attribute is calculated using the following formula:



Refund relation:

\* Refund\_name attribute: Cancel

\* Refund\_percent attribute is calculated using the following formula:



# CHAPTER 2: SYSTEM ANALYSIS AND DESIGN

## Conceptual level database design

From the necessary data in description of the problem, the following Entity Relationship Model (ERD) is formed.

Diagram

Description automatically generated with low confidence

Sharp image: [ERD sharp image (busticketbookingerd.netlify.app)](https://busticketbookingerd.netlify.app/)

## Logical level database design

From the Entity Relationship Model (ERD), we have:

* Bus (id\_bus, registration\_number, model, capacity, status, type)
* Trip (id\_trip, id\_bus, id\_route, departure\_time, duration, booked\_seat, status)
* TripDriver (id\_trip, id\_driver)
* Agent (id\_agent, id\_place, id\_cash\_reserve, address, name)
* CashReserve (id\_cash\_reserve, counter)
* BusStation (id\_bus\_station, id\_place, address, name, bus\_capacity, count\_current\_bus)
* Driver (id\_driver, license\_level, type, state)
* Route (id\_route, id\_start\_station, id\_end\_station, distance)
* Place (id\_place, region)
* Employee (id\_employee, id\_account, id\_agent, name, address, phone\_number, identity\_number, salary, email, birthdate)
* Position (id\_position, type)
* Privilege (id\_privilege, name)
* SystemAccount (id\_account, user\_name, password)
* Ticket (id\_ticket, id\_trip, status, fare, type, seat\_number)
* Event (id\_event, discount\_type, discounted\_price)
* Refund (id\_refund, refund\_name, refund\_percent)
* Passenger (id\_passenger, name, phone\_number, address, identity\_number, gender, email)
* Booking (id\_booking, id\_ticket, id\_passenger, id\_employee, booking\_time)
* Package (id\_package, id\_trip, mass, price, sender\_contact, receiver\_contact, id\_route)
* PackagePricePolicy (id\_policy, price\_per\_km, mass\_unit)
* Agent\_Trip (id\_agent, id\_trip)
* Agent\_Event (id\_agent, id\_event)
* Agent\_Refund (id\_agent, id\_refund)
* Agent\_Policy (id\_agent, id\_policy)
* BusRoute\_BusStation (id\_busRoute, id\_busStation)
* Employee\_Position (id\_employee, id\_position)
* Employee\_Ticket (id\_employee, id\_ticket)
* Position\_Privilege (id\_position, id\_privilege)

## Required constraints

| **No.** | **Table** | **Constraint** |
| --- | --- | --- |
| **1** | Bus | Primary key: id\_bus |
| **2** | Trip | Primary key: id\_trip |
| **3** | TripDriver | Foreign keys:  id\_trip references Trip(id\_trip),  id\_driver references Driver(id\_driver), respectively |
| **4** | Driver | Primary key: id\_driver |
| **5** | Agent | Primary key: id\_agent  Foreign key:  id\_cash\_reserve references CashReserve(id\_cash\_reserve) |
| **6** | CashReserve | Primary key: id\_cash\_reserve |
| **7** | BusStation | Primary key: id\_bus\_station |
| **8** | BusRoute | Primary key: id\_route  Foreign keys:  id\_bus\_station1 and id\_bus\_station2 references BusStation(id\_bus\_station) |
| **9** | Place | Primary key: id\_place |
| **10** | Employee | Primary key: id\_employee  Foreign keys:  id\_account references SystemAccount(id\_account),  id\_position references Position(id\_position),  id\_agent references Agent(id\_agent), respectively |
| **11** | Position | Primary key: id\_position |
| **12** | Privilege | Primary key: id\_privilege  Foreign key:  id\_position references Position(id\_position) |
| **13** | SystemAccount | Primary key: id\_account |
| **14** | Ticket | Primary key: id\_ticket  Foreign key:  id\_trip references Trip(id\_trip) |
| **15** | Event | No constraints |
| **16** | Refund | No constraints |
| **17** | Passenger | Primary key: id\_passenger |
| **18** | Booking | Primary key: id\_booking  Foreign keys:  id\_ticket references Ticket(id\_ticket),  id\_passenger references Passenger(id\_passenger),  id\_employee references Employee(id\_employee), respectively |
| **19** | Package | Primary key: id\_package  Foreign keys:  id\_route references Route(id\_route),  id\_trip references Trip(id\_trip), respectively |
| **20** | PackagePricePolicy | Primary key: id\_policy |
| **21** | AgentAndTrip | Primary key: id\_agent, id\_trip  Foreign key: id\_agent references Agent(id\_agent), id\_trip references Trip(id\_trip) |
| **22** | AgentAndEvent | Primary key: id\_agent, id\_event  Foreign key: id\_agent references Agent(id\_agent)  id\_event references Event(id\_event) |
| **23** | AgentAndRefund | Primary key: id\_agent, id\_refund  Foreign key: id\_agent references Agent(id\_agent), id\_refund references Refund(id\_refund) |
| **24** | AgentAndPPP | Primary key: id\_agent, id\_ppp  Foreign key: id\_agent references Agent(id\_agent), id\_ppp references PackagePricePolicy(id\_policy) |
| **25** | BusRouteAndBusStation | Primary key: id\_busRoute, id\_busStation  Foreign key: id\_busRoute references BusRoute(id\_route), id\_busStation references BusStation(id\_busStation) |
| **26** | EmployeeAndPosition | Primary key: id\_employee, id\_postion  Foreign key: id\_employee references Employee(id\_employee), id\_position references Position(id\_position) |
| **27** | EmployeeAndTicket | Primary key: id\_employee; id\_ticket  Foreign key: id\_employee references Employee(id\_employee), id\_ticket references Ticket(id\_ticket) |
| **28** | PositionAndPrivilege | Primary key: id\_position, id\_privilege  Foreign key: id\_position references Position(id\_position), id\_privilege references Privilege(id\_privilege) |

## Database settings and constraints

[BUS]



[TRIP]



[TRIP\_DRIVER]



[AGENT]



[CASH\_RESERVE]



[BUS\_STATION]



[DRIVER]

[ROUTE]



[PLACE]



[EMPLOYEE]



[POSITION]



[PRIVILEGE]



[SYSTEM\_ACCOUNT]



[TICKET]



[EVENT]



[REFUND]



[PASSENGER]



[BOOKING]

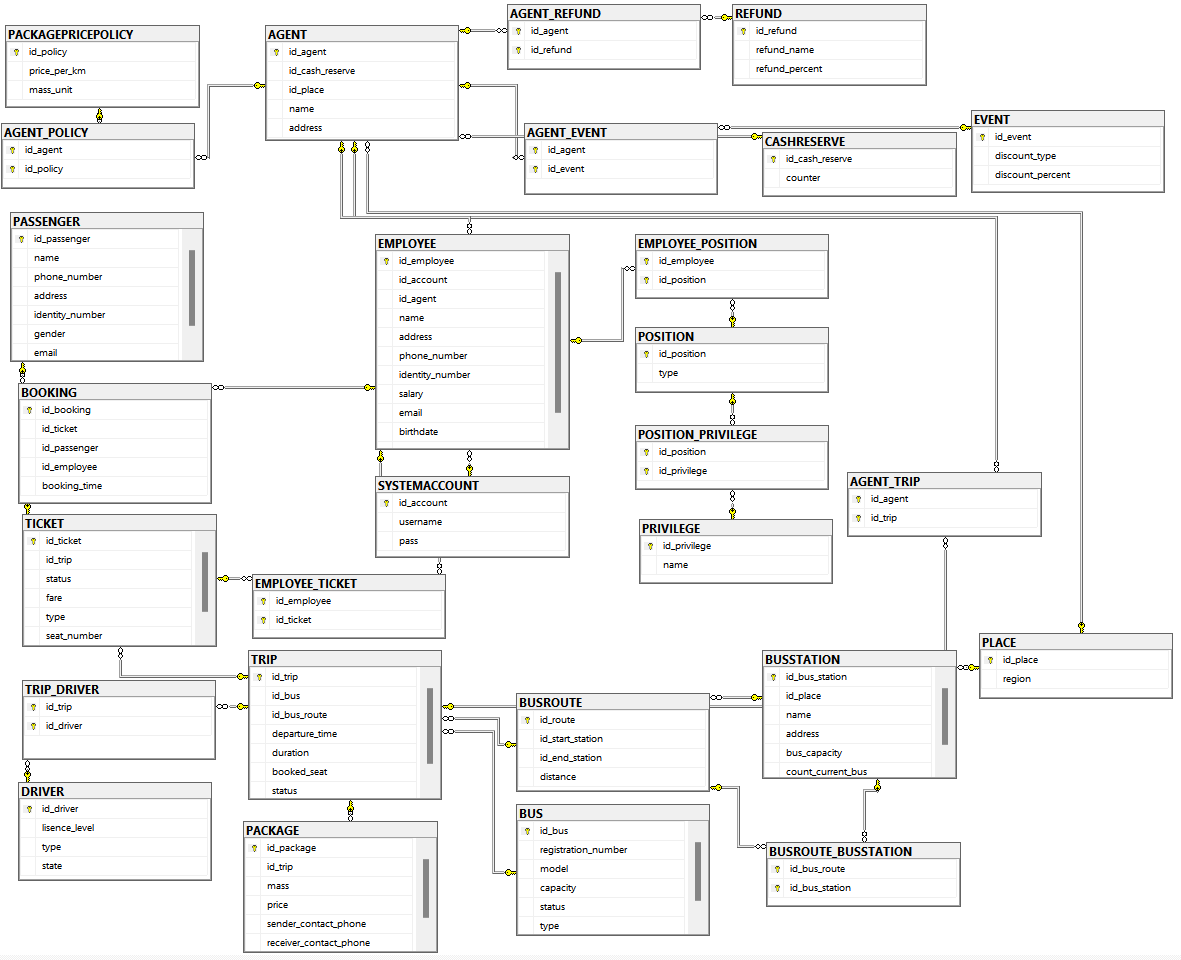


[PACKAGE]



[PACKAGE\_PRICE\_POLICY]





## Other constraints

**Constrain bus identity** **after add one**

-- Set constraint bus identity automatically.

USE BusManagement

ALTER TABLE Bus

ADD CONSTRAINT AUTO\_ID\_Bus

DEFAULT DBO.AUTO\_ID\_Bus() FOR ID\_bus;

GO

**Constrain passenger identity and ticket identity** **after customer buy a ticket**

-- Set constraint ticket identity automatically.

USE BusManagement

ALTER TABLE TICKET

ADD CONSTRAINT AUTO\_ID\_ticket

DEFAULT DBO.AUTO\_ID\_ticket() FOR ID\_ticket;

GO

-- Set constraint passenger identity automatically.

USE BusManagement

ALTER TABLE PASSENGER

ADD CONSTRAINT AUTO\_ID\_passenger

DEFAULT DBO.AUTO\_ID\_passenger() FOR ID\_passenger;

GO

**Constrain trip identity** **after add one**

-- Set constraint trip identity automatically.

USE BusManagement

ALTER TABLE TRIP

ADD CONSTRAINT AUTO\_ID\_trip

DEFAULT DBO.AUTO\_ID\_trip() FOR ID\_trip;

GO

**Constrain route identity** **after add one**

-- Set constraint route identity automatically.

USE BusManagement

ALTER TABLE BUS\_ROUTE

ADD CONSTRAINT AUTO\_ID\_route

DEFAULT DBO.AUTO\_ID\_route() FOR ID\_route;

GO

**Constrain position identity** **after add one**

-- Set constraint route identity automatically.

USE BusManagement

ALTER TABLE POSITION

ADD CONSTRAINT AUTO\_ID\_position

DEFAULT DBO.AUTO\_ID\_position() FOR ID\_position;

GO

**Constrain employee identity** **after add one**

-- Set constraint route identity automatically.

USE BusManagement

ALTER TABLE EMPLOYEE

ADD CONSTRAINT AUTO\_ID\_employee

DEFAULT DBO.AUTO\_ID\_employee() FOR ID\_employee;

GO

**Constrain agent identity** **after add one**

-- Set constraint bus identity automatically.

USE BusManagement

ALTER TABLE Agent

ADD CONSTRAINT AUTO\_ID\_Agent

DEFAULT DBO.AUTO\_ID\_Agent() FOR ID\_agent;

GO

**Constrain booking identity** **after add one**

-- Set constraint bus identity automatically.

USE BusManagement

ALTER TABLE Booking

ADD CONSTRAINT AUTO\_ID\_Booking

DEFAULT DBO.AUTO\_ID\_Booking() FOR ID\_booking;

GO

**Constrain package identity** **after add one**

-- Set constraint bus identity automatically.

USE BusManagement

ALTER TABLE Package

ADD CONSTRAINT AUTO\_ID\_Package

DEFAULT DBO.AUTO\_ID\_Package() FOR ID\_package;

GO

**Constrain package price policy identity** **after add one**

-- Set constraint bus identity automatically.

USE BusManagement

ALTER TABLE PackagePricePolicy

ADD CONSTRAINT AUTO\_ID\_PackagePricePolicy

DEFAULT DBO.AUTO\_ID\_PackagePricePolicy() FOR ID\_policy;

GO

## Trigger to check for constraints

**\*Update state of an employee on Employee and Driver relation**

CREATE TRIGGER tr\_employee\_update\_stateEmployee

ON Employee

AFTER UPDATE

AS

BEGIN

DECLARE @employee\_id CHAR

DECLARE @new\_state CHAR

SELECT @employee\_id = ID\_employee, @new\_state = state FROM inserted

-- Kiểm tra nếu trạng thái mới của nhân viên là 0

IF @new\_state = 0

BEGIN

-- Cập nhật trạng thái của nhân viên trong bảng Employee thành 0

UPDATE Employee SET state = 0 WHERE ID\_employee = @employee\_id

UPDATE Driver SET state = 0 WHERE ID\_driver = @employee\_id

END

SELECT state FROM Driver WHERE ID\_driver = @employee\_id

END

**\* Delete account of employee and update state of Employee when his/her state =0**

CREATE TRIGGER tr\_employee\_deleteAccount

ON Employee

AFTER UPDATE

AS

BEGIN

DECLARE @account\_id CHAR

DECLARE @new\_state CHAR

SELECT @account\_id = ID\_account, @new\_state = inserted.state FROM inserted

IF @new\_state = 0

BEGIN

-- Delete account

DELETE FROM SystemAccount WHERE @account\_id = ID\_account

END

END

1. **Views**

**View list of active employee which is working**

CREATE VIEW [dbo].[ActiveEmployee] AS

SELECT Employee.ID\_employee, Employee\_ID\_account, Employee.name, Employee.address, Employee.phone\_number, Employee.identity\_number, Employee.salary, Employee.email,

Employee.birthday, Agent.name, Position.type

FROM

Employee AS temp1 INNER JOIN Agent AS temp 2

ON temp1.ID\_agent = temp2.ID\_agent

INNER JOIN Position as temp3

ON temp1.ID\_position = temp3.ID\_position

WHERE temp1.status = 1

**View list of waiting trip:**

CREATE VIEW [dbo].[WaitingTrip] AS

temp1.ID\_trip, temp1.departure\_time, temp1.duration, temp1.booked\_seat, temp1.registration\_number, temp1.type,

temp2.name AS start\_point, temp3.name AS end\_point

FROM

((SELECT Trip.\*, Bus.registration\_number, Bus.type

FROM Trip INNER JOIN Bus

ON Trip.ID\_bus = Bus.ID\_bus) AS temp0

INNER JOIN BusRoute

ON BusRoute.ID\_route = temp0.ID\_route) AS temp1

INNER JOIN (

SELECT temp1.ID\_route, temp1.ID\_bus\_station1, BusStation.name

FROM temp1 INNER JOIN BusStation

ON temp1.ID\_bus\_station1 = BusStation.ID\_bus\_station

) AS temp2

ON temp1.ID\_route = temp2.ID\_route

INNER JOIN (

SELECT temp1.ID\_route, temp1.ID\_bus\_station2, BusStation.name

FROM temp1 INNER JOIN BusStation

ON temp1.ID\_bus\_station2 = BusStation.ID\_bus\_station

) AS temp3

ON temp1.ID\_route = temp3.ID\_route

WHERE temp1.status = 'Waiting'

**View list of going trip:**

CREATE VIEW [dbo].[GoingTrip] AS

temp1.ID\_trip, temp1.departure\_time, temp1.duration, temp1.booked\_seat, temp1.registration\_number, temp1.type,

temp2.name AS start\_point, temp3.name AS end\_point

FROM

((SELECT Trip.\*, Bus.registration\_number, Bus.type

FROM Trip INNER JOIN Bus

ON Trip.ID\_bus = Bus.ID\_bus) AS temp0

INNER JOIN BusRoute

ON BusRoute.ID\_route = temp0.ID\_route) AS temp1

INNER JOIN (

SELECT temp1.ID\_route, temp1.ID\_bus\_station1, BusStation.name

FROM temp1 INNER JOIN BusStation

ON temp1.ID\_bus\_station1 = BusStation.ID\_bus\_station

) AS temp2

ON temp1.ID\_route = temp2.ID\_route

INNER JOIN (

SELECT temp1.ID\_route, temp1.ID\_bus\_station2, BusStation.name

FROM temp1 INNER JOIN BusStation

ON temp1.ID\_bus\_station2 = BusStation.ID\_bus\_station

) AS temp3

ON temp1.ID\_route = temp3.ID\_route

WHERE temp1.status = ‘Going’

**View list of finished trip:**

CREATE VIEW [dbo].[FinishTrip] AS

temp1.ID\_trip, temp1.departure\_time, temp1.duration, temp1.booked\_seat, temp1.registration\_number, temp1.type,

temp2.name AS start\_point, temp3.name AS end\_point

FROM

((SELECT Trip.\*, Bus.registration\_number, Bus.type

FROM Trip INNER JOIN Bus

ON Trip.ID\_bus = Bus.ID\_bus) AS temp0

INNER JOIN BusRoute

ON BusRoute.ID\_route = temp0.ID\_route) AS temp1

INNER JOIN (

SELECT temp1.ID\_route, temp1.ID\_bus\_station1, BusStation.name

FROM temp1 INNER JOIN BusStation

ON temp1.ID\_bus\_station1 = BusStation.ID\_bus\_station

) AS temp2

ON temp1.ID\_route = temp2.ID\_route

INNER JOIN (

SELECT temp1.ID\_route, temp1.ID\_bus\_station2, BusStation.name

FROM temp1 INNER JOIN BusStation

ON temp1.ID\_bus\_station2 = BusStation.ID\_bus\_station

) AS temp3

ON temp1.ID\_route = temp3.ID\_route

WHERE temp1.status = ‘Finish'

**View about list of idle interprovince bus:**

CREATE VIEW [dbo].[IdleInterprovinceBus]

AS

SELECT rel.ID\_bus, rel.registration\_number, rel.model, rel.capacity

FROM Bus as rel

WHERE Bus.status = 'idle' AND Bus.type = 'interprovince'

**View about list of break interprovince bus:**

CREATE VIEW [dbo].[BreakInterprovinceBus]

AS

SELECT rel.ID\_bus, rel.registration\_number, rel.model, rel.capacity

FROM Bus as rel

WHERE Bus.status = 'break' AND Bus.type = 'interprovince'

**View about list of incident interprovince bus:**

CREATE VIEW [dbo].[IncidentInterprovinceBus]

AS

SELECT rel.ID\_bus, rel.registration\_number, rel.model, rel.capacity

FROM Bus as rel

WHERE Bus.status = 'incident' AND Bus.type = 'interprovince'

**View about list of ongoing interprovince bus:**

CREATE VIEW [dbo].[OnGoingInterprovinceBus]

AS

SELECT rel.ID\_bus, rel.registration\_number, rel.model, rel.capacity

FROM Bus AS rel INNER JOIN (

Select Trip.ID\_bus, Trip.ID\_trip

FROM Trip

WHERE Trip.status = 'going'

) as rel2

ON rel.ID\_bus = rel2.ID\_bus

WHERE Bus.status = 'ongoing' AND Bus.type = 'interprovince'

**View about list of passenger booking information ( in detail ):**

CREATE VIEW [dbo].[BookingInfor]

AS

SELECT rel.ID\_booking, temp1.ID\_ticket, temp1.ID\_trip, temp1.seat\_number, temp1.type, temp1.fare, temp2.name AS passenger\_name, temp2.phone\_number AS passenger\_phone\_number, temp2.address AS passenger\_address, temp2.email AS passenger\_email, temp2.gender AS passenger\_gender, temp3.name AS employee\_name, temp3.phone\_number AS employee\_phonee\_number

FROM Booking AS rel INNER JOIN Ticket AS temp1

ON rel.ID\_ticket = temp1.ID\_ticket

INNER JOIN Passenger AS temp2

ON rel.ID\_passenger = temp2.ID\_passenger

INNER JOIN Employee AS temp3

ON rel.ID\_employee = temp3.ID\_employee

**View about list of currently bus route information ( in detail ):**

CREATE VIEW [dbo].[BusRouteInfor]

AS

SELECT rel.ID\_route, temp1.start\_point, temp2.end\_point, rel.distance

FROM BusRoute AS rel INNER JOIN (

SELECT rel.ID\_Route, BusStation.name as start\_point

FROM rel INNER JOIN BusStation

ON rel.ID\_bus\_station1 = BusStation.ID\_bus\_station

) AS temp1

ON rel.ID\_route = temp1.ID\_route

INNER JOIN (

SELECT rel.ID\_Route, BusStation.name as end\_point

FROM rel INNER JOIN BusStation

ON rel.ID\_bus\_station2 = BusStation.ID\_bus\_station

) AS temp2

ON rel.ID\_route = temp2.ID\_route

**View about list of Going trip driver:**

CREATE VIEW [dbo].[WaitingTripDriverInfor]

AS

SELECT rel.\*, temp2.ID\_driver, temp2.name AS driver\_name, temp2.phone\_number AS driver\_phone\_number

FROM WaitingTrip AS rel INNER JOIN TripDriver AS temp1

ON rel.ID\_trip = temp1.ID\_trip

INNER JOIN (

SELECT Driver.ID\_driver, Employee.name, Employee.phone\_number

FROM Driver INNER JOIN Employee

ON Driver.ID\_driver = Employee.ID\_employee

) AS temp2

ON temp1.ID\_driver = temp2.ID\_driver

**View about list of Going trip driver:**

CREATE VIEW [dbo].[GoingTripDriverInfor]

AS

SELECT rel.\*, temp2.ID\_driver, temp2.name AS driver\_name, temp2.phone\_number AS driver\_phone\_number

FROM GoingTrip AS rel INNER JOIN TripDriver AS temp1

ON rel.ID\_trip = temp1.ID\_trip

INNER JOIN (

SELECT Driver.ID\_driver, Employee.name, Employee.phone\_number

FROM Driver INNER JOIN Employee

ON Driver.ID\_driver = Employee.ID\_employee

) AS temp2

ON temp1.ID\_driver = temp2.ID\_driver

**View about list of Going trip driver:**

CREATE VIEW [dbo].[FinishTripDriverInfor]

AS

SELECT rel.\*, temp2.ID\_driver, temp2.name AS driver\_name, temp2.phone\_number AS driver\_phone\_number

FROM FinishTrip AS rel INNER JOIN TripDriver AS temp1

ON rel.ID\_trip = temp1.ID\_trip

INNER JOIN (

SELECT Driver.ID\_driver, Employee.name, Employee.phone\_number

FROM Driver INNER JOIN Employee

ON Driver.ID\_driver = Employee.ID\_employee

) AS temp2

ON temp1.ID\_driver = temp2.ID\_driver

**View about list of employee accounts:**

CREATE VIEW [dbo].[EmployeeAccount]

AS

SELECT temp1.ID\_employee, temp1.name, temp2.username, temp2.password

FROM Employee AS temp1, SystemAccount AS temp2

**View about list of waiting trip which is still empty chairs**

CREATE VIEW [dbo].[TripWithAvailableChair]

AS

SELECT temp1.\*, temp2.capacity - temp1.booked\_seat AS available\_position

FROM WaitingTrip AS temp1 INNER JOIN (

SELECT Bus.capacity, Trip.ID\_trip

FROM Trip INNER JOIN Bus

ON Trip.ID\_bus = Bus.ID\_bus

) as temp2

ON temp1.ID\_trip temp2.ID\_trip

WHERE temp2.capacity - temp1.booked\_seat > 0

**View about list of waiting trip which is full chairs**

CREATE VIEW [dbo].[TripWithAvailableChair]

AS

SELECT temp1.\*, temp2.capacity - temp1.booked\_seat AS available\_position

FROM WaitingTrip AS temp1 INNER JOIN (

SELECT Bus.capacity, Trip.ID\_trip

FROM Trip INNER JOIN Bus

ON Trip.ID\_bus = Bus.ID\_bus

) as temp2

ON temp1.ID\_trip temp2.ID\_trip

WHERE temp2.capacity - temp1.booked\_seat = 0

# CHAPTER 3: DESIGNING FUNCTIONS

## Connect to database