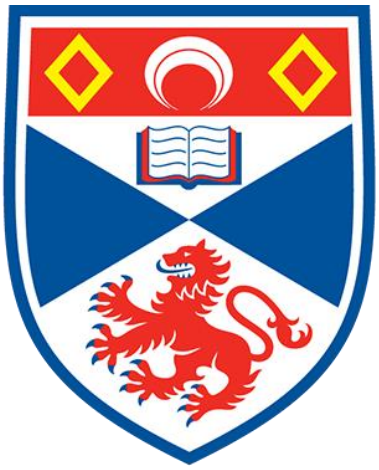


University of
St Andrews

Practical 2: Database management system

Matriculation ID: 230015450

Task1

1. Staff (staff_id, name, email, street, city, postcode)

- Staff id is the primary key. It includes their staff ID, name, email, street address, city, and postal code. Each staff member is uniquely identified by their staff ID.
- The staff id uses the CHAR(9) data type because the employee ID is a fixed-length identifier.
- Name, email address, street, city and postcode are all of VARCHAR type because their length is variable. And the string length setting based on the reality.

2. Phone (staff_id, phone_number, type)

- Phone is Used to store employee phone information, including staff id, phone number and type.
- The employee ID serves as the primary key and is also a foreign key, associated with the Staff table to establish the relationship between employees and phone information.
- Phone numbers use the CHAR (10) data type because phone numbers usually have a fixed length.
- The foreign key constraint ensures the correlation between staff and their phone numbers.

3. Manager (manager_id, annual_salary)

- Manager is used to store manager information, including manager ID and annual salary.
- The manager ID is the primary key and the foreign key and is associated with the Staff table to identify the manager as a type of employee.
- Annual salary uses NUMERIC (8, 2) data type to support accurate salary calculation.

4. Driver (staff_id, hourly_salary)

- Driver is Used to store driver information, including staff id and hourly salary.

The staff id is the primary key and the foreign key.

- Hourly salary uses the INT data type because it is usually an integer.
- Because manager and driver are low-level entity sets of a disjoint generalized model, they will inherit high-level attributes.

5. Station (station_name, town, manager_id)

- Station is used to store station information, including station name, town and manager id.
- The station name is the primary key, and the manager ID is the foreign key, which is related to the Manager table to identify the manager of the station.
- VARCHAR type is chosen here for station name and town as their length is variable.

6. service (service number, departure_station_name, destination_station_name)

- Service is used to store service information, including service number, departure station and destination station.
- The service number is the primary key, and the departure station name and destination station name are foreign keys, respectively associated with the Station

table to identify the relationship between services and stations.

- VARCHAR type is chosen here for the three attributes. But the service number usually has three characters.

7. Drives (Service_number, staff_id, hours_driven)

- Drives is used to store driver driving service information, including staff ID, service number, and driving hours.
- The combination of staff ID and service number is used as the primary key and a foreign key, because the relationship is many to many.
- Numeric (5,1) is chosen here, Because the hours driven usually is not as an interger. Most of the case is have one decimal place.

8. Service time (service_number, start_time)

- It is used to store service time information, including service number and start time.
- The combination of service number and start time is used as the primary key. Because it is a weak entity set. The service number is a foreign key and is associated with the Service table to identify the relationship between service time and services.

9. stop (stop_name)

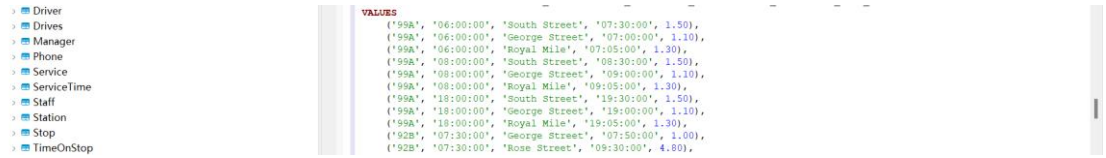
- It is used to store information about stops, including stop names.
- Stop name as primary key.

10. TimeonStop(stop_name,service_number,start_time,arrival_time,fare_from_origin)

- It is used to store service time information at stops, including stop name, service number, start time, arrival time and fare from the origin.
- The combination of stop point name, service number and start time is used as the primary key and at the same time as a foreign key, and is associated with the Stop, Service and Service time table.

Task2

- Create database tables and foreign key relationships between tables. This includes the 10 tables from Task 1. Appropriate foreign key relationships are established between these tables to ensure data consistency and integrity.
- Use the INSERT statement to insert sample data into each table. Those result have been put into the buses.sql document.
- At the beginning of the buses.sql file, writing “PRAGMA foreign keys = TRUE;” to enable foreign key constraints to ensure that foreign key relationships enforce data integrity. And also use the cascading actions when beginning to create table.
- The next step Run the buses.sql file to insert the table structure and data into the database. This will create the database and populate it with data. The result shows in below:



The integrity of the data has been confirmed through a few simple SQL statement queries. However, for the convenience of the SQL script, I did not put the query and deletion into it.

Task3: Data Manipulation

- 1. List all services which have Seagate Bus Station in Dundee as their origin;

```
SELECT service_number, departure_station_name
FROM Service
WHERE departure_station_name  = 'Seagate';
```

	service_number	departure_station_name
1	92B	Seagate
2	32R	Seagate

- 2. Calculate an average monthly salary of a bus station manager

```
SELECT staff_id, annual_salary/12 AS monthly_salary
FROM Manager;
```

	staff_id	monthly_salary
1	230015450	7,041.6941666667
2	230015451	6,379.2116666667
3	230015452	5,000.0166666667
4	230015453	4,170.1933333333
5	230015454	2,920.8791666667

- 3. List the names of all drivers of services which have Edinburgh Bus Station in Edinburgh as their origin or destination, in increasing order of the amount to be paid to them for the hours driven;

```
SELECT STF.name AS driver_name, DRR.hourly_salary, DRR.hourly_salary
* Drives.hours_driven AS total_earnings
FROM Driver DRR
JOIN Staff STF ON DRR.staff_id = STF.staff_id
JOIN Drives ON DRR.staff_id = Drives.staff_id
JOIN Service ON Drives.service_number = Service.service_number
JOIN Station ON Service.departure_station_name = Station.station_name
OR Service.destination_station_name = Station.station_name
WHERE (Station.town = 'Edinburgh' AND (Service.departure_station_name
= 'Edinburgh Bus Station' OR Service.destination_station_name =
'Edinburgh Bus Station'))
```

```
ORDER BY total_earnings;
```

	driver_name	hourly_salary	total_earnings
1	Susan	65.1	130.2
2	James	70.3	210.9
3	Jennifer	80.2	401

- 4. List the manager of the most connected station, measured by the number of services which have that station as their origin or destination

```
SELECT S.manager_id, COUNT(*) AS num_services
FROM Station S
JOIN Service ON S.station_name = Service.departure_station_name OR
S.station_name = Service.destination_station_name
GROUP BY S.manager_id
ORDER BY num_services DESC
LIMIT 1;
```

	manager_id	num_services
1	230015453	3

- 5. For the bus stop "Buchanan Gardens, St Andrews" list in the chronological order arrival times at this stop, origins, destinations, and service numbers of all bus services passing this stop between 10 am and 2 pm

```
SELECT TOS.arrival_time, S.departure_station_name,
S.destination_station_name, S.service_number
FROM TimeOnStop TOS
JOIN Service S ON TOS.service_number = S.service_number
WHERE TOS.stop_name = 'Buchanan Gardens, St Andrews'
AND TOS.arrival_time >= '10:00:00'
AND TOS.arrival_time <= '14:00:00'
ORDER BY TOS.arrival_time;
```

	arrival_time	departure_station_name	destination_station_name	service_number
1	11:15:00	City Square Bus Stop	St Andrews Bus Station	93A
2	13:15:00	City Square Bus Stop	St Andrews Bus Station	93A

- 6. Query the number of all services (Service), according to the number of services departing from a certain Edinburgh Bus Station

```
SELECT departure_station_name, COUNT(*) AS num_services
FROM Service
WHERE departure_station_name = 'Edinburgh Bus Station'
GROUP BY departure_station_name;
```

departure_station_name	num_services
Edinburgh Bus Station	2

- 7. Find the average passenger fare for each service

```
SELECT service_number, AVG(fare_from_origin) AS average_fare
FROM TimeOnStop
GROUP BY service_number
HAVING COUNT(service_number) >= 2;
```

	service_number	average_fare
1	32R	1.5
2	92B	2.9
3	93A	2.95
4	99A	1.3
5	X59	4.9

- 8. Query the number of services at each station and sort them in descending order by the number of services

```
SELECT S.station_name, COUNT(*) AS num_services
FROM Station S
LEFT JOIN Service ON S.station_name = Service.departure_station_name
OR S.station_name = Service.destination_station_name
GROUP BY S.station_name
HAVING COUNT(S.station_name) >= 2
ORDER BY num_services DESC;
```

	station_name	num_services
1	Edinburgh Bus Station	3
2	City Square Bus Stop	2
3	Seagate	2
4	St Andrews Bus Station	2

- 9. Query the number of services and the names of their origin and destination stations, sorted by service number in ascending order

```
SELECT service_number, departure_station_name,
destination_station_name
FROM Service
ORDER BY service_number
LIMIT 2;
```

	service_number	departure_station_name	destination_station_name
1	32R	Seagate	Old Course
2	92B	Seagate	Edinburgh Bus Station

- View1: Displaying Basic information for all employees.

```
CREATE VIEW EmployeeBasicInfo AS
SELECT staff_id, name, email, street, city, postcode
FROM Staff;
```

EmployeeBasicInfo

	staff_id	name	email	street	city	postcode
1	230015450	David	dav83@bus.co.uk	Strathkiness	Fife	KY16 9UW
2	230015451	Tom	tom82@bus.co.uk	Buchanan Gardens	Fife	KY16 9WJ
3	230015452	John	joh55@bus.co.uk	Princes Street	Edinburgh	EH5 8IU
4	230015453	Mary	mar52@bus.co.uk	Haymarket	Edinburgh	EH3 7ST
5	230015454	Michael	mic74@bus.co.uk	Deansgate	Manchester	M20 3YI
6	230015455	Jennifer	jen56@bus.co.uk	Buchanan Street	Glasgow	G8 4OE
7	230015456	Susan	sus80@bus.co.uk	Sauchiehall Street	Glasgow	G7 2OU
8	230015457	James	jam05@bus.co.uk	Bridgeton	Glasgow	G6 5YU
9	230015458	Linda	lin10@bus.co.uk	Redriff Rd	London	E1W 2SL
10	230015459	Robert	rob20@bus.co.uk	Oxford Street	London	SE15 4OI

- View2 Display station information, including station name, city, and manager name.

```
CREATE VIEW StationInfo AS
SELECT S.station_name, S.town, ST.name AS manager_name
FROM Station S
LEFT JOIN Manager M ON S.manager_id = M.staff_id
LEFT JOIN Staff ST ON M.staff_id = ST.staff_id;
```

	station_name	town	manager_name
1	St Andrews Bus Station	St Andrews	David
2	Old course	St Andrews	Tom
3	Seagate	Dundee	John
4	Edinburgh Bus Station	Edinburgh	Mary
5	City Square Bus Stop	Dundee	Michael

- View3 This view contains information about drivers, including their names, hourly salary, and total salary

```
CREATE VIEW DriverEarnings AS
SELECT ST.name AS driver_name, D.hourly_salary, D.hourly_salary *
Drives.hours_driven AS total_earnings
FROM Driver D
JOIN Staff ST ON D.staff_id = ST.staff_id
JOIN Drives ON D.staff_id = Drives.staff_id;
```

DriverEarnings

	driver_name	hourly_salary	total_earnings
1	Jennifer	80.2	401
2	Susan	65.1	130.2
3	James	70.3	210.9
4	Linda	102.5	205
5	Robert	60.5	242

Task4: Reflection

- The good job I did

In this exercise, I think I did a good job in Tasks 1. Especially when converting the ER model into a relational model, I think I did a better job. I can clearly judge strong entity sets, composite attributes, and multi-valued attributes. Weak entity sets, relationship sets, all participation, many-to-one relationships, one-to-many relationships, and disjoint generalize these concepts. When converting the relational model, you can clearly determine how to convert these concepts when they appear. In Task2, I can be more proficient in creating tables, inserting data, and specifying data types.

- Problem I encountered and solution.

First, the first problem encountered in this exercise is to check the data integrity. I was not very clear about this problem at the beginning. Then I returned to the lecture in the classroom and discovered the cascade, so I used the update and deletion of data in the two tables to verify the changes in the related foreign keys.

The second problem is writing complex SQL statements. I feel relatively unskilled here. For this, I can only analyze the SQL connected tables step by step and judge how to write correct SQL statements based on foreign keys and primary keys.

The next problem I found an error when using the sql script to create the database, mainly because of punctuation issues. When I was writing a sql statement, I was prompted with a syntax error. When I looked carefully, I discovered that there was no semicolon at the end of the statement.

The last problem is so important that results me delay for this practical work. When I created the table, I found the error by executing the statement. The TimeOnStop table always prompts me "A foreign key constraint failed". At first, I thought it was a problem with the lab software, but later on I still had the same problem when I used my own computer. So, I carefully checked the value of the foreign key associated first to verify whether it was consistent with the value of the table in TimeOnStop. Finally, I found the error. It was indeed a SQL script execution error caused by inconsistent data. The reason for this situation is that I changed the table data when designing the fifth query in Task 2, so the SQL script failed to execute, which took an entire afternoon.