CSG1207 Systems and Database Design

Assignment 2: Database Design & Implementation (Airline)

Assignment Marks: Marked out of 60, (30% of unit)
Task 1 is marked out of 20, (10% of unit)
Task 2 is marked out of 40, (20% of unit)

Due Dates: Inform tutor if working in a pair: 15 April 2024, 9:00AM
Task 1 (Database Design): 22 April 2024, 9:00AM
Task 2 (Implementation): 20 May 2024, 9:00AM

General Assignment Information

Before a database can be implemented and used, it must be designed in a way that ensures it is appropriate for the task at hand. Tools such as Entity-Relationship Diagrams and Data Dictionaries assist in designing and communicating the structure of a database. Once a design has been finalised, the database can be implemented in a DBMS. The Data Definition Language commands of SQL are used to create the database and its tables, attributes, and constraints, after which the Data Manipulation Language commands can be used to manipulate data within the database. This assignment takes you through the design and implementation process, using Microsoft SQL Server.

You may work in **pairs** (maximum of **2** people) to complete this assignment or choose to work **alone**. If you wish to work in a pair, you **must** inform your tutor (by email) of the names and student numbers of both members at *least one week before the due date of Task 1*. If you choose to work alone, be aware that there are *no extra marks* to compensate for the heavier workload. If working in a pair, *work through the whole assignment together*, rather than dividing it and completing sections individually. This will help to ensure that both people learn the content and contribute evenly.

The assignment consists of two tasks. The first task, **Database Design**, requires a word-processed document in PDF format detailing the design of your database. The second task, **Implementation**, is a collection of SQL scripts which create and populate the database designed in the first task, and then query the data it contains. A small number of marks are dedicated to **presentation, notation, and formatting**.

While Task 1 is worth few marks it is the *basis of Task 2*, and therefore it is especially important that it is done well, otherwise further work will be required before Task 2 can be attempted.

Task 1 – Database Design (20 marks)

Your first task is to design a database for the scenario detailed on the following pages. Your final database design should include approximately 10 entities.

> ***Note:*** *The scenario for this assignment is the same as the one from Task 3/4 of Assignment 1. Be sure to incorporate any feedback received in Assignment 1.*

State any **assumptions** (2 marks) you have made regarding your database design at the beginning of the database design document. Do not make any assumptions that significantly change the structure of the scenario, as this may make Task 2 of the assignment more difficult or inapplicable. Only make assumptions that influence your database design. If you are unsure about an assumption you wish to make, ask your tutor.

Once you feel you have identified the entities, attributes, and relationships of the scenario in sufficient depth, you are required to create a **logical ER diagram** (4 marks) and a corresponding **physical ER diagram** (4 marks) to depict your database. Adhere to the distinctions between logical and physical ER diagrams covered in Lecture 3. It is recommended that you draw your diagrams on paper first, to find a layout that is clear and can be created in an electronic format.

Lastly, create a **data dictionary** (8 marks), with an entry for each entity in your database. The entries should list the name of the entity (table), a list of attributes (columns), essential information about the domain of the attributes (e.g. data type, null/not null, identity, default values…), and details of any constraints applied to attributes. List the entries in your data dictionary in an appropriate *table creation order* that can be used to create the database. Include any additional information, if any, that may be needed to implement the database. Remember, a data dictionary should contain *all the information needed* to implement your database design. Use the data dictionary in Lecture 4 and the data dictionary of the "company" database (Module 5) as examples.

Some marks are also awarded for **presentation and notation** (2 marks).

Your complete database design should consist of a list of assumptions, logical and physical ER diagrams, and a data dictionary. This should be submitted as a ***single PDF document and*** be sure to open the PDF file before submitting it to ensure that your diagrams appear as intended. Make sure that your assignment includes the unit code, assignment number/name, year and semester and your name and student number on the first page.

Scenario Details

You are required to design and create a database for an airline. The database must contain details of the airline's planes, flights, flight instances and staff, as well as supporting data as detailed below.

- Details of the **planes** owned by the airline must be stored. This must include the registration number of the plane, the year it was built, a foreign key identifying the model of the plane, and its first, business and economy class passenger capacities (as three separate integers).

- Details of plane **models** must be stored. This must include the model number, manufacturer name, travel range in kilometres and cruising speed in kilometres per hour.

- Details of **flight paths** must be stored. This must include a flight path number, a foreign key identifying the airport the flight departs from, a foreign key identifying the airport the flight arrives at, and the distance between the airports in kilometres.
    - Note: A flight path is *a route that the airline offers*, e.g. Flight path "QF574" is a Perth to Sydney route offered by Qantas.

- Details of **flight instances** must be stored. This must include a flight instance ID, a foreign key identifying the plane making the flight, a foreign key identifying the flight path that is being flown, the date/time that the flight leaves, and date/time that the flight arrives.
    - Note: A flight instance is *a specific trip along a flight path at a certain date and time,* e.g. The 2019-02-06 QF574 flight at 05:45.

- Details of **airports** must be stored. This must include the IATA airport code (e.g. "PER" for Perth), the name of the airport, its latitude and longitude, and a foreign key identifying the country that the airport is in.

- A list of **countries** must be stored. This must simply contain the two-letter country code (e.g. "AU") and the name of the country.

- Details of **pilots** must be stored. This must include a pilot ID, their first name, last name and hire date.
    - The database must record which models of plane each pilot is qualified to fly. Each pilot must be qualified to fly at least one model of plane.
    - The database must record which pilots are aboard each flight instance. Each flight instance requires exactly two pilots – a pilot and co-pilot.

- Details of **flight attendants** must be stored. This must include an attendant ID, their first name, last name, and a list of languages they speak (in one column, e.g. "English, French").
    - For training purposes, some flight attendants' mentor other flight attendants. The database must record each flight attendant's mentor if they have one.
    - The database must record which attendants are aboard each flight instance. Every flight instance must have at least one attendant aboard.
    - The database must record which attendant has been designated the flight service manager (FSM) of each flight instance. Each flight instance must have a FSM.

General Information and Guidelines

The information above describes the entities, attributes and relationships required in the database. Some minor details, such as the cardinality of some relationships, have been omitted. It is up to you to make (*and state*) any assumptions you need to complete the database design. If you are uncertain about any part of the scenario described above, seek clarification from your tutor.

Be sure to specify the most appropriate data type (and length, where applicable) for each attribute in your data dictionary. Where sample data is provided, make sure that the data type and length of your columns are suitable to contain the sample data. Note that when you are storing a date/time, it should be stored as a single column – do *not* split the date and time into two columns.

Many of the entities in this scenario contain an identifying attribute that is suitable to be used as the entity's primary key, such as plane registration numbers, model numbers, flight path numbers, airport codes, etc. These values will *not* be auto-incrementing. For entities that do *not* have a meaningful attribute to use as the primary key, use auto-incrementing integers – for example, flight instance ID numbers, and the ID numbers of staff should be auto-incrementing integers. A compound primary key may be suitable for certain intermediary entities.

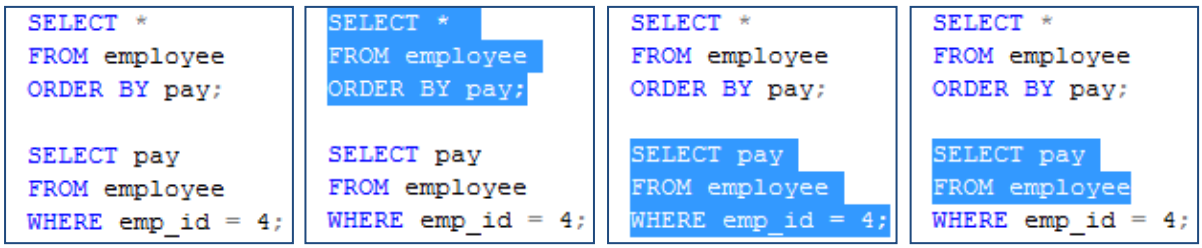
Read the scenario details *several times* to ensure that your database design incorporates all the elements described. *If you desire feedback on your work in progress, send it to your tutor*.

Task 2 – Implementation (40 marks)

Once your database has been designed, it is time to implement it in a DBMS, populate the database, and then manipulate the data via queries. The deliverables of this task are **three files containing SQL statements**. We will be using Microsoft SQL Server 2019 or above – your SQL scripts *must* run in the same environment used in the unit/labs.

Create your scripts as three ".sql" files, with the filenames listed in the following headings. **Templates for the script files are provided with this assignment brief** – please use them. Format your code for **readability and** use **comments** for headings and to provide further detail or information about your code if needed **(2 marks)**.

As each of the script files will contain numerous SQL statements, it is particularly useful to be aware of a particular feature of SQL Server Management Studio (SSMS): If you have selected some text in a query window, *only the selected text will be executed when you press the Execute button*.



Execute both queries | *Execute query 1 only* | *Execute query 2 only* | *Execute part of query 2*

This makes it much easier to test a single statement, or even part of a statement, within a script file.

You are required to create all the scripts detailed below. Please take note of the file names and **use the template files provided with this assignment brief**.

Filename: create.sql

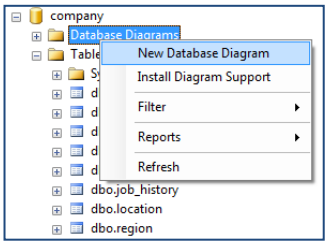
This file is a creation script, like the "company.sql" file (Module 5) used to create the company database.

Database Creation & Population Script (8 marks)

Produce a script to create the database you designed in Task 1 (incorporating any changes you have made since then). Be sure to give your columns the same data types, properties and constraints specified in your data dictionary, and be sure to name tables and columns consistently. Include any suitable default values and any CHECK or UNIQUE constraints that you feel are appropriate.

Make sure this script can be run *multiple times* without resulting in any errors (hint: drop the database if it exists before trying to create it). *You can use/adapt the code at the start of the creation scripts of the sample databases available in the unit materials to implement this.* You will need to follow an appropriate creation order when creating your tables – you cannot create a table with a foreign key constraint that refers to a table which does not yet exist.

Once you have created your database, it is recommended that you use SSMS to create an ER diagram and use this to verify that your implementation matches your design. This can be done by right clicking on the "Database Diagrams" folder of the database in the Object Explorer in SSMS.



Following the SQL statements to create your database and its tables, you must include statements to *populate the database with sufficient test data*. You are only required to populate the database with enough data to make sure that *all views and queries return meaningful results*. You can start working on your views and queries and write INSERT statements as needed for testing as you go.

For example, imagine you are working on a query which joins data from two tables and only displays the rows that meet certain criteria, then orders the results by a certain column. To test this, you will need to insert some data in both tables, making sure that some of the rows meet the criteria and others do not, and making sure that the column used for ordering contains a range of different values. Once you have inserted this data, you can test that your query works.

The final create.sql should be able to create your database and populate it with enough data to make all views and queries return meaningful results.

Make sure referential integrity is observed – you cannot add data to a column with a foreign key constraint if you do not yet have data in the column it references. Remember that the value of an auto-incrementing integer column will be generated automatically when a row is inserted - y*ou cannot specify a value for the column*. Omit such columns from your INSERT statements.

Note: *The data you add is simply for testing purposes, and therefore does not need to be particularly realistic, cohesive or consistent.*

Avoid spending unnecessary amounts of time writing sample data. To assist with this, I have included data for a number of tables in the create_TEMPLATE.sql file.

Filename: views.sql

Views allow you to refer to the result of a SELECT statement as if it were a table, making query writing easier.

Flight Instance View (3 marks)

Create a *view* that selects the following details of all flight instances:

- All the columns in the flight instance table.
- The departure airport code, arrival airport code and distance of the flight path.
- The full name of the pilot, co-pilot, and flight service manager.
  - Concatenate the first name and last name into one column, e.g. "Joe Bloggs."
- The model number of the plane.
- A column with an alias of "attendant_quota" that contains the number of flight attendants expected to be on board based upon the total capacity of the plane.
  - Calculate this by adding together the plane's first, business and economy class capacities and dividing the total by 100. *Note: If one attendant per 100 passengers sounds unrealistic, you are correct – It is to avoid requiring too much sample data!*

Creating this view will require joining the flight instance table to the flight path table, the plane table, the attendant table, and the pilot table (twice) – a total of 5 joins. This view serves as a very convenient replacement for the flight instance table, as it includes the relevant details from numerous other tables – use this view in queries that need this data!

flight_instance_id	rego_num	flight_path_num	pilot_id	copilot_id	fsm_id	departure_time	arrival_time	model_num	departure_airport_code	arrival_airport_code	distance	pilot_name	copilot_name	fsm_name	attendant_quota
1	VH-ABC	SA100	1	2	1	2019-06-01 04:00...	2019-06-01 08:30...	A340 300	PER	SYD	3284	John Smith	Joe Bloggs	Marsha Burton	2
2	VH-DEF	SA101	3	4	2	2019-06-01 09:00...	2019-06-01 12:00...	A380 800	SYD	AKL	2164	Martin Jones	Sue Woods	Denise Perdeau	4
3	VH-GHI	SA102	2	3	3	2019-06-01 11:00...	2019-06-02 02:30...	A380 800	AKL	DEL	12495	Joe Bloggs	Martin Jones	Samuel Forster	4
4	VH-JKL	SA103	4	3	4	2019-06-01 15:00...	2019-06-01 20:00...	737 600	DEL	PEK	3815	Sue Woods	Martin Jones	Sampreet Kaur	1

(example partial output of the Flight Instance View – your data may vary; example illustrates structure only)

You are encouraged to *use the views to simplify the queries* which follow – You can use a view in a SELECT statement in the same way as you can use a table, often avoiding the need to write the same joins and calculations over and over.

When writing a view, it is easiest to write the SELECT statement first, and only add the CREATE VIEW statement to the beginning once you have confirmed that the SELECT statement is working correctly. If you wish to create additional views to use in the queries which follow, include them in this file.

Joins are covered in Module 9, and views are covered in Module 10.

Filename: queries.sql

Write SELECT statements to complete the following queries. If you do not understand or are not sure about exactly what a query requires, contact your lecturer or tutor.

Query 1 – Model Information (2 marks)

Write a query that selects the following information for all rows in the model table:

- The "model name" (the manufacturer name and model number concatenated together).
- The travel range in kilometres and nautical miles (1 km = 0.54 nautical miles).
- The cruising speed in kilometres per hour and miles per hour (1 km/h = 0.62 mph).

Give all columns appropriate aliases and order the results by the model's name.

model_name	travel_range_km	travel_range_nm	cruise_speed_kmh	cruise_speed_mph
Airbus A340 300	13705	7400.70	896	555.52
Airbus A380 800	16112	8700.48	1088	674.56
Boeing 737 600	5649	3050.46	827	512.74
Boeing 777 200LR	17500	9450.00	945	585.90

(expected query results, if provided data used)

Query 2 – Plane Finder (2 marks)

Write a query that selects the registration number, model number, travel range, cruising speed and the passenger capacity of each class for all planes that have a total capacity of at least 250 and a range of at least 14,000kms. Order the results by cruising speed, in descending order.

rego_num	model_num	travel_range	cruise_speed	total_capacity
VH-DEF	A380 800	16112	1088	484
VH-GHI	A380 800	16112	1088	615
VH-PQR	777 200LR	17500	945	266

(expected query results, if provided data used)

Query 3 – Departing Flight Information (2 marks)

Write a query that selects the flight path number, arrival airport code, departure time minus one hour and model number of all upcoming flight instances (departure time in the future). Give the columns aliases of "Flight Number," "Destination," "Boarding Time" and "Plane." Order the results by departure time. Using the Flight Instance View in this query is recommended.

Flight Number	Destination	Boarding Time	Plane
SA100	SYD	2019-06-01 03:00:00	A340 300
SA101	AKL	2019-06-01 08:00:00	A380 800
SA102	DEL	2019-06-01 10:00:00	A380 800
SA103	PEK	2019-06-01 14:00:00	737 600
SA104	PER	2019-06-01 17:00:00	777 200LR

(example of query results – your data may vary; example illustrates structure only)

Query 4 – Qualified Pilots (3 marks)

Write a query that selects the pilot ID number and last name of all pilots who are qualified to fly the plane used in flight instance 3. Use a subquery to determine the model of plane for flight instance 3.

pilot_id	last_name
1	Smith
3	Jones

(example of query results – your data may vary; example illustrates structure only)

Query 5 – Domestic Flight Descriptions (3 marks)

Write a query that selects all domestic flight paths (those where both airports are in the same country) and concatenates information about them to produce a single column in this format:

"**[flight path number]** is a domestic flight in **[country name]**, going from **[departure airport name]** (**[departure airport code]**) to **[arrival airport name]** (**[arrival airport code]**)."

domestic_flights
SA100 is a domestic flight in Australia, going from Perth International Airport (PER) to Sydney Kingsford Smith International Airport (SYD).
SA105 is a domestic flight in Australia, going from Sydney Kingsford Smith International Airport (SYD) to Perth International Airport (PER).

(example of query results – your data may vary; example illustrates structure only)

Query 6 – Flight Statistics (3 marks)

Write a query that selects the registration number, number of flight instances, total distance travelled and average flight duration, grouped per plane. The average duration should be shown in hours, ideally to two decimal places of accuracy. Order the results by the number of flight instances, in descending order. Using the Flight Instance View in this query is recommended.

Hint: *It may take some experimentation to select the average duration as described – ask on the Discussion Board for tips!*

rego_num	flight_count	total_distance	average_duration
VH-ABC	3	13422	5.83
VH-JKL	3	10914	4.83
VH-MNO	2	20469	12.75
VH-DEF	1	2164	3.00
VH-GHI	1	12495	15.50

(example of query results – your data may vary; example illustrates structure only)

Query 7 – Attendant Search (4 marks)

Write a query that that selects the attendant ID, full name (by concatenating their first name and last name) and language list of all attendants who do not mentor any other attendants and speak Hindi or Cantonese.

Hint: *The WHERE clause of this statement will involve LIKE comparisons, AND/OR, and careful use of parentheses. Several approaches can be taken to find non-mentoring attendants, most of which involve a subquery.*

attendant_id	full_name	languages
3	Samuel Forster	English, Hindi, Malay
5	Perdeep Singh	English, Hindi
6	Adrian Lim	Hindi, Cantonese

(example of query results – your data may vary; example illustrates structure only)

Query 8 – Pilot Statistics (4 marks)

Write a query that that selects the pilot ID and full name (by concatenating their first name and last name) of all pilots, as well as how many years they have worked for the airline, how many flight instances they have piloted and how many flight instances they have co-piloted. Be sure to include all pilots, even if they have not piloted or co-piloted any flight instances.
Order the results by the number of years worked, in descending order.

Hint: *This will involve using OUTER JOINs and GROUP BY, as well as COUNTing DISTINCT column values.*

pilot_id	full_name	worked_years	pilot_count	copilot_count
1	John Smith	20	4	0
2	Joe Bloggs	18	2	3
3	Martin Jones	18	1	5
4	Sue Woods	14	3	2

(example of query results – your data may vary; example illustrates structure only)

Query 9 – Understaffed Flight Instances (4 marks)

Write a query that selects the flight instance ID, flight path number, departure time, attendant quota and actual number of attendants rostered for any flight instances where the actual number of attendants rostered is less than the quota. See Page 7 (views.sql) for details regarding the attendant quota. Using the Flight Instance View in this query is recommended.

Hint: *This will involve using COUNT, GROUP BY and HAVING.*

flight_instance_id	flight_path_num	departure_time	attendant_quota	rostered_attendants
2	SA101	2019-06-01 09:00:00	4	3
3	SA102	2019-06-01 11:00:00	4	2

(example of query results – your data may vary; example illustrates structure only)

Presentation, Notation and Formatting (2 marks per part)

A small number of marks are awarded for presentation, notation, and formatting. This includes:

- Presentation and appearance of word-processed PDF document for Task 1
- Appropriateness and consistency of notation used for diagrams/data dictionary in Task 1
- Appropriate commenting and formatting of scripts in Task 2

Submission of Deliverables

Submit your database design (Task 1, a PDF document) by the Task 1 due date above. Submit your three script files (Task 2) by the Task 2 due date above. Submit the files to the appropriate locations in the Assessments area of Blackboard.

If you are working in pairs, **both** people should submit the same assignment. Be sure to include the **name and student number of both members** in the assignment content and submission comments. Remember, if you wish to work in a pair, you **must** inform your tutor of the names and student numbers of both members at *least one week before the due date of Task 1*.

Submissions via email are NOT permitted, unless you are specifically instructed to do so. Make sure that your assignment includes the unit code, assignment number/name, year and semester and your name and student number on the first page.

Referencing, Plagiarism and Collusion

The **entirety of your assignment must be your own work or the work of your pair** (unless otherwise referenced) and produced for the current instance of the unit. Any use of unreferenced content you did not create constitutes plagiarism and is deemed an act of academic misconduct. All assignments will be submitted to plagiarism checking software which includes previous copies of the assignment, and the work submitted by all other students in the unit.

Remember that this is an **individual/pair** assignment. *Never give anyone that you are not officially working with any part of your assignment – even after the due date or after results have been released. Do not work together with other students who you are not officially working with on assignments – helping someone by explaining a concept or directing them to the relevant resources is fine, but doing the assignment for them or alongside them, or showing them your code is not appropriate.* An unacceptable level of cooperation between students who are not officially working together on an assignment is collusion and is deemed an act of academic misconduct. If you are uncertain about plagiarism, collusion or referencing, simply contact your tutor, lecturer or unit coordinator and ask.

You may be asked to **explain and demonstrate your understanding** of the work you have submitted. Always remember that the purpose of an assignment is for you to **demonstrate your understanding** of the unit content and **your ability to apply it** to the task presented to you.

Assignment 2 Task 1 Marking Key

Marks are allocated as follows for this assignment.

Criteria	Marks
Assumptions All/Any assumptions that influence the database design clearly stated.	**2**
Logical ER Diagram Diagram accurately depicts the scenario and includes all elements specified in the brief.	**4**
Physical ER Diagram Accurately depicts the scenario and is a correct translation of the logical ER diagram.	**4**
Data Dictionary Includes all entities and details of attributes as specified in brief and correct creation order used.	**8**
Presentation and Notation Assignment is well presented, uses consistent and appropriate notation.	**2**

Total:	**20 (10% of unit)**

Assignment 2 Task 2 Marking Key

Marks are allocated as follows for this assignment.

Criteria All scripts are judged on correctness, appropriateness and readability of SQL code.	**Marks**
Database Creation & Population Script	**8**
Flight Instance View	**3**
Query 1	**2**
Query 2	**2**
Query 3	**2**
Query 4	**3**
Query 5	**3**
Query 6	**3**
Query 7	**4**
Query 8	**4**
Query 9	**4**
Formatting – Scripts are well formatted and use appropriate commenting.	**2**

Total:	**40 (20% of unit)**