Construct the Spaces© database in SQL Server and load test data. Write queries for the required transactions.

DAT601 Assessment Two Part 2

Physical Implementation

23/6/23

Jayden Houghton – 13524050

Contents

[Introduction 1](#_Toc137126102)

[SQL DDL 1](#_Toc137126103)

[Test Data 1](#_Toc137126104)

[SQL Transactions 2](#_Toc137126105)

[Significant Issues 3](#_Toc137126106)

[Conclusion 3](#_Toc137126107)

[References 3](#_Toc137126108)

# SQL DDL

*Build the database in SQL Server using DDL statements.*

## Tables

Please refer to the attached file *Spaces\_Create.sql*. This file contains all of the table creation and foreign key connection scripts.

I used Visual Paradigm (Visual Paradigm, Retrieved 2023) to logically model my database. I then used the generate database functionality to generate the DDL script. This automatically generated the table creation and foreign key alters. This worked well, although I did have to reformat them to be more readable. I also had to add square brackets to names like data or datetime, as these are keywords.

## Check Constraints

I have implemented three check constraints into my database.

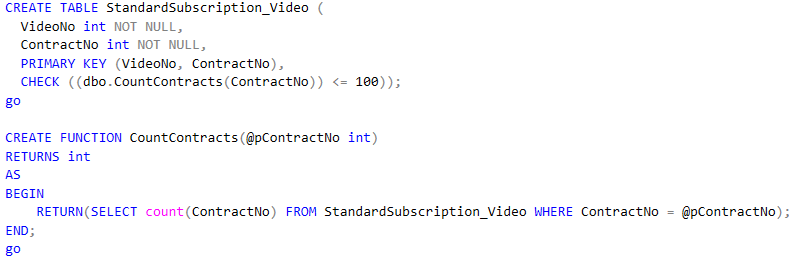
Two of them are for discounts:

1. AdminExecutive maxDiscount
2. Contract Discount

This is because Spaces© has a business rule that only allows discounts unto 3%. By putting in a check, this will ensure that larger values are not accidentally added.



The other discount is to limit the number of video streams a subscriber can view. Spaces© has a business rule that limits subscribers to viewing up to 100 video streams from each sensor. As this is a more complex check, I have implemented it using a function as shown below.



## Roles and Users

I have included five basic roles and added a example user to each role.

1. Administrator
2. Technician
3. Salesperson
4. AdminExecutive
5. Maintainer

These cover the basic business roles needed as well as the technical database roles. These have been granted various permissions depending on the access needed. For example, the maintainer only has access to select, insert and update maintenance records in the database, as that is the only data they will need.

In a professional situation, these permissions would likely be more in-depth to have tighter security of the data. They would also likely have other permissions, such as accessing various transactions, but for the purposes of this assignment, I believe that this demonstrates how this functions.

## Views

I have added a contract details view to the database, which shows contract and subscriber information. This is an easier way to view the data across multiple tables at the same time. It also gives more control over which columns can be accessed by who. Depending on how the different users and roles use the database, it may be beneficial to add other views in the future.

A picture containing text, font, screenshot

Description automatically generated

# Test Data

*Generate and load appropriate test data. There should be a minimum of 100 rows across all the tables and no table should contain less than 5 rows. Provide a list of data in your tables. Please provide the appropriate DML statements for this step in the form of INSERT statements.*

Please refer to the attached file *Spaces\_Populate.sql*. This file contains all of the test data insert scripts.

I used Mockaroo (Mockaroo, Retrieved 2023) to generate my test date. This is a free online tool that contains a lot of features and customisation.

There are approximately 850 records across all tables, with 5-100 per table.

# SQL Transactions

*Write and run a set of SQL queries required to provide the required information for the project case study. These will be resolved during class discussions, but should be determined from the case study documentation. Provide the SQL queries and their results. Include a brief description of the purpose of each query.*

## Analysis

I have analysed each of the transactions that will be implemented in the database using a tabular format.

Since Spaces© is an international company and the database will be interacted with from all around the world, it is hard to judge when peak time will be. I have decided just to include an average estimate.

Code

* Insert = I
* Read = R
* Delete = D
* Update = U
* Entry point = E

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction A | | Frequency: | 10-20/h | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Contact Info | Phone  Email  Secondary Email | I (E)  I  I | 1 |
| - | Address | Street Address  Suburb  City  Country  Postcode | I (E)  I  I  I  I | 1 |
| - | Subscriber | First Name  Last Name  Password  DoB  Address ID  Contact Info ID | I (E)  I  I  I  I  I | 1 |
| - | Contract | Start Date  End Date  Monthly Price  Discount  Salesperson ID  Admin Executive ID  Subscriber ID | I (E)  I  I  I  I  I  I | 1 |
| - | Sensor Subscription | Contract No  Sensor ID | I (E)  I | 1 |
| - | Standard Subscription | Contract No | I (E) | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction B | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Contract | Discount  Salesperson ID  Subscriber ID | R (E)  R  R | 2,000,000\* |
| Contract | Subscriber | First Name  Last Name  Subscriber ID  Address ID | R  R  R (E)\*  R | 2,000,000 |
| Subscriber | Address | Address ID  Street Address  Suburb  City  Country  Postcode | R (E)\*  R  R  R  R  R | 2,000,000 |
| Contract | Salesperson | Employee ID | R (E)\* | 1,000,000\*\* |

\* Estimated average 2 subscriptions per subscriber, at 1,000,000 subscribers.

\*\* Estimated half will subscribe online and wont have a salesperson.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction C | | Frequency: | 15m/s\* | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Data | Shared Audio  Human Voice  Body Shape  Skeletal Points  Texture  Date Time  Longitude  Latitude  Altitude  PContract No  Sensor ID | I (E)  I  I  I  I  I  I  I  I  I  I | 1 |

\* All sensors record data (even if the contract company is not receiving it) as stated in the brief. 500,000 sensors x 30fps = 15,000,000 records/second if all sensors are constantly on.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction D | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Sensor | Sensor ID  Latitude  Longitude | R (E)  R  R | 500,000\* |
| Sensor | Sensor Subscription | Sensor ID  Contract No | R (E)\*  R | 2,000,000\*\* |
| Sensor Subscription | Contract | Contract No  Subscriber ID | R (E)\*  R | 2,000,000 |
| Contract | Subscriber | Subscriber ID  First Name  Last Name | R (E)\*  R  R | 2,000,000 |

\* Over 500,000 sensors are deployed. Assume all have at least one subscription.

\*\* Estimated 2 subscriptions per subscriber.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction E | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Contract | Contract No  Subscriber ID | R (E)  R | 1 |
| Contract | Subscriber | Subscriber ID  First Name  Last Name | R (E)\*  R  R | 1 |
| Contract | Data | Contract No  Human Voice  Shared Audio  Body Shape  Skeletal Points  Texture  Date Time  Longitude  Latitude  Altitude | R (E)\*  R  R  R  R  R  R  R  R  R | 77.76m\* |

\* Assume data is being recorded at 30fps. The sensor stores 1 month of data, but it is unknow how long data is stored on the Spaces© database. Assuming 1 month (30 days) as well -> 77,760,000 or 77.76 million records. This is a very large amount of data that could have a significant impact of the performance of their database. Spaces may with to reduce this or store historical data on a separate database much as a data warehouse for analysis.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction F | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Standard Subscription \_Video | Video No  Contract No | R (E)  R (E) | 1 |
| Standard Subscription \_Video | Video | Video No  Contract No | R (E)\*  R | 100,000\* |
| Standard Subscription \_Video | Contract | Contract No  Subscriber ID | R (E)\*  R | 100,000 |
| Contract | Subscriber | Subscriber ID  First Name  Last Name | R (E)\*  R  R | 100,000 |

\* 1,000,000 subscribers, estimate watching an average of 10% of the time = 100,000.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction G | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Sensor\_Part | PartName  Sensor ID | R (E)  R | 1 |
| Sensor\_Part | Part\_Supplier | PartName  SupplierName | R (E)\*  R | 100\* |

\* This would depend on the number of parts that each sensor will have. Estimated 100 but less with the test data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction H | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Sensor | Sensor ID  Latitude  Longitude | R (E)  U  U | 1 |
| - | Zone \_Sensor | Sensor ID  Zone ID | R (E)  R + U | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction I | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Data | Data No  Shared Audio  Human Voice  Body Shape  Skeletal Points  Texture  Contract No  Sensor ID | D (E)  D  D  D  D  D  D  D | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction J | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Maintenance Record | Sensor ID  Date Time | R (E)  R (E) | 1,000,000\* |
| Maintenance Record | Maintenance Record\_Part | Part Name  Sensor ID  Date Time | R (E)  R  R | 200,000\*\* |
| Maintenance Record\_Part | Part | Part Name  Price | R (E)  R | 500,000\*\*\* |

\* Assume each sensor has had maintenance an average of two times.

\*\* Assuming 20% of maintenance records require new parts.

\*\*\* Assume an average of 1 new parts per two sensors.

## Creation

Please refer to the attached file *Spaces\_Transactions.sql*. This file contains all of the transactions.

Each written query includes comments to explain their purpose and how they work.

## Indexing

After performing transactional analysis to identify high-frequency tables and columns, I have identified four secondary indexes to add:

1. Subscriber: subscriber ID
2. Contract: contract no
3. Video: video no
4. Data: data no

Secondary indexes can be added by adding the following code:



After more extensive testing of database performance, Spaces© may with to add other index, such as to address, contact info and sensor.

## File Storage

File storage will be done using the default Microsoft SQL Server system. The default method is to uses balance trees with clustered indexes. The bulk of the data is stored at each leaf node.

## Controlled Redundancy

Redundancy is the practice of keeping a piece of data in multiple places at the same time. Generally, data redundancy is avoided as it can cause data discrepancies, corruption and increased database sizes, however it can be beneficial in some situations (Gillis, 2021).

Controlled redundancy is the practice of introducing redundant data to improve database read times. While this does use more space, copying stable data can considerably reduce the number of pages accessed and thus reduce the time required (Pattern: Controlled Redundancy, Retrieved 2023). This process require denormalising the database, as redundancy violates first normal form.

This only works or stable data, or data that is unlikely to change. The majority of Spaces© database will be changing frequently, so very little of the data could be implemented with controlled redundancy. Controlled redundancy would likely not be worthwhile implementing in the Spaces© database.

## Estimated Size

I have estimated the database size in the external Excel Spreadsheet. The is a page for each table and a total page at the end. I have estimated the database size to be 373 PetaBytes. This would vary drastically by the actually sizes of the video and data being recorded, as well as how long data is being stored for.

All row counts used for this calculation are the estimates from the tables data dictionary with the exception of the video and data amounts.

* Video: 500,000 sensors \* average streams 50 \* 30fps \* 60 seconds = 45,000,000,000 (45b)
* Data: 500,000 sensors \* 30fps \* 60 seconds = 900,000,000

I have estimated that video and data will be stored on the Spaces© database for 1 minute. This gives room to deal with latency, minor buffering and stream timing issues. If video or data is needed past this, it can be requested from the sensor which will store it for up to 1 month. I think that it is unrealistic for all video and data to be stored much longer than this (373 PB is huge), and this time may be able to be reduced.

# Significant Issues

*Document any significant problem(s) you encountered and the approach you took to solving it/them. This should show you made a genuine effort to solve these problem(s) and the learning you achieved from that process.*

I found that since my logical model was well designed, there were few significant problems during the database implementation. This shows that good planning can save a lot of time and effort during the database constructions.

### Data

While Mockaroo saved me lots of time, it wasn’t perfect. On a couple of occasions, I had to modify the data formatting it outputted or the foreign key values so it would work properly.

I also had to make a few other adjustments to the data over the course of the implementation and transaction building, where a datatype or values needed to be changed. These did slow my progress but were minor overall.

I later learned that Mockaroo is able to same generated data types, which can then be referenced when generating other primary keys. While this would have taken time to learn and set up, it likely would have saved me time in the long run, especially if I was working with a larger database.

### Transactions

I did not have any significant issues will writing or analysing the transactions. While it was not an issue, I did write the SQL transactions first an perform analysis after, which I later learnt was the wrong order. This actually worked well for me, as both steps were relatively easy for the most part. I did however have to make minor adjustments to the database to be able to perform all transactions correctly. For example, I added datetime and location to each data record so this could be displayed.

### Reflection

Overall I am quite happy with how this project has turned out. I believe that I have covered almost all of the requirements to a consistently high standard.

# References

Gillis, A. S. (2021, December). *Redundant*. Retrieved from Tech Target: https://www.techtarget.com/searchstorage/definition/redundant

*Pattern: Controlled Redundancy*. (Retrieved 2023). Retrieved from Object Architects: http://www.objectarchitects.de/ObjectArchitects/orpatterns/Performance/ControlledRedundancy/#:~:text=Controlled%20Redundancy%20is%20a%20technique,speed%20up%20reading%20database%20access.

(Retrieved 2023). Retrieved from Mockaroo: https://www.mockaroo.com/

(Retrieved 2023). Retrieved from Visual Paradigm: https://www.visual-paradigm.com/