Appendix A

**ASSIGNMENT COVER SHEET**

|  |  |  |
| --- | --- | --- |
| **Student’s name** | (First name)  Joshua | (Last name)  Morton |
| **Module name** | Data Modelling and Database Design | |
| **Title of assignment** | Joshua Morton DMDD Assessment | |
| **Complete Word Count in my assignment** | Partial Submission | |
| **Date submitted** | 11/09/2023 | |

All work must be submitted by the due date. If an extension of time to submit work is required, a Mitigating Circumstances request must be submitted.

X

Has an extension been approved? Yes No

If yes, please give the new submission date ….…/..…./…….

|  |
| --- |
| IMPORTANT: THIS STATEMENT MUST BE READ & SIGNED  **Academic Integrity Statement**  Academic integrity and honesty are fundamental to the academic work you produce at the University of Roehampton. You are expected to complete coursework which is your own and which is referenced appropriately. The university has in place measures to detect academic dishonesty in all its forms. If you are found to be cheating or attempting to gain an unfair advantage over other students in any way, this is considered academic misconduct and you will be penalised accordingly.   ​  **I declare that the work I am submitting is my own work, is properly referenced and has not been submitted elsewhere.** |
| **Student Signature (Full Name):**  **Date: 11/09/2023** |

Contents

[Table of Figures 2](#_Toc142162532)

[Introduction 2](#_Toc142162533)

[Task 1 - Database Technologies 2](#_Toc142162534)

[Relational Database Management Systems (RDBMS) 2](#_Toc142162535)

[NoSQL – Not Only SQL 4](#_Toc142162536)

[Hierarchical Databases: 4](#_Toc142162537)

[Document Database 5](#_Toc142162538)

[Key-Value Database 6](#_Toc142162539)

[Graph Database 6](#_Toc142162540)

[Wide-Column Database 7](#_Toc142162541)

[Data Lakes 8](#_Toc142162542)

[Task 2 - Database Modelling 8](#_Toc142162543)

[Scenario Overview 8](#_Toc142162544)

[Conceptual Modelling 8](#_Toc142162545)

[Entity Identification 9](#_Toc142162546)

[Relation Identification 9](#_Toc142162547)

[Conceptual Diagram 10](#_Toc142162548)

[Logical Modelling 11](#_Toc142162549)

[Attribute Identification 11](#_Toc142162550)

[Primitive Logical Diagram 12](#_Toc142162551)

[Appendices 22](#_Toc142162552)

[Appendix A: 22](#_Toc142162553)

[Figure 1 – RDBMS Table 22](#_Toc142162554)

[Figure 2 – Stack Overflow Database Environments Survey: 22](#_Toc142162555)

[Figure 3 – ACID Principles: 23](#_Toc142162556)

[Figure 4 – Hierarchical Data Tree 23](#_Toc142162557)

[Figure 5 – Document Model Example 23](#_Toc142162558)

[Figure 6 – Key-Value Database Example 23](#_Toc142162559)

[Figure 7 – Shortest Path Key-Value Data Access 23](#_Toc142162560)

[Figure 8 – Graph Database example Schema 23](#_Toc142162561)

[Figure 9 – Wide Column Database Example 23](#_Toc142162562)

[Figure 10 – The Data Lake Design Pattern 24](#_Toc142162563)

[Figure 11 – Entity Diagram 24](#_Toc142162564)

[Figure 12 – Attribute Entity Diagram 24](#_Toc142162565)

[Appendix B: 24](#_Toc142162566)

[Figure 1 – RDBMS Table: 24](#_Toc142162567)

[Figure 2 – Hierarchical Data Example: 25](#_Toc142162568)

[References 30](#_Toc142162569)

[Glossary 31](#_Toc142162570)

# Table of Figures

[Figure 1 Example RDBMS SQL Table Relations (W3Resource 2022) 3](#_Toc142162571)

[Figure 2 - Database Environments used by Professional Developers - Stack Overflow Survey 2023 3](#_Toc142162572)

[Figure 3 - ACID Principles 4](#_Toc142162573)

[Figure 4 - Hierarchical Data Structure - MariaDB 2015 5](#_Toc142162574)

[Figure 5 Document Data Model Representation (Pore, 2018) 5](#_Toc142162575)

[Figure 6 Example of a Key Value database. (Redis 2023) 6](#_Toc142162576)

[Figure 7 DB Providers & DB Types Read Time for Related Nodes (ArangoDB 2018) 6](#_Toc142162577)

[Figure 8 A simple Graph schema example (Morgante 2020) 7](#_Toc142162578)

[Figure 9 A simple example of a Wide-Column Database 7](#_Toc142162579)

[Figure 10 Data Lake Design Pattern (DataKitchen.io, 2017) 8](#_Toc142162580)

[Figure 11 Entity Diagram 9](#_Toc142162581)

[Figure 12 Entity Relations Identification 10](#_Toc142162582)

[Figure 13 Conceptual Diagram 11](#_Toc142162583)

[Figure 14 Entity Diagram with Attributes Identified 12](#_Toc142162584)

[Figure 15 First Stage Logical Model **Error! Bookmark not defined.**](#_Toc142162585)

# Introduction

This report uses a variety of specialized terminology and abbreviations. Explanations are defined in the Glossary.

# Task 1 - Database Technologies

## Relational Database Management Systems (RDBMS)

The RDBMS uses a structure based on the relational model proposed by E.F. Codd in 1970 (Connolly 2015 Part 2 – Chapter 4), to allow us to identify and access data attributes via table-based (Appendix B - Figure 1) relations using unique keys.

A screenshot of a computer

Description automatically generated

Figure 1 Example RDBMS SQL Table Relations (W3Resource 2022)

They are designed to represent complex data schemas and minimize data redundancy through normalization whilst offering performant transactions using SQL[[1]](#endnote-1).

RDBMS vendors, such as *PostgreSQL, MySQL & SQLite* are amongst the most common databases, evidenced below.

A screenshot of a graph

Description automatically generated

Figure 2 - Database Environments used by Professional Developers - Stack Overflow Survey 2023

An excellent reason to use an RDBMS’ is when your transactions must adhere to the ACID principles[[2]](#endnote-2) which most vendors support implicitly, and if your organisation has complex querying requirements.

A screenshot of a computer

Description automatically generated

Figure 3 - ACID Principles

Finance, education & ecommerce systems’ complex structure prefer RDBMSs for their reliable and consistent data storage.

However,’ RDBMS’ performance is hindered by ACID, making them unsuitable for platforms like YouTube (Shivang, 2019). They’re unsuited for unstructured JSON data, prevalent in the Web and schema changes add significant complexity, also leading to a poor representation of “real-world” entities (Connolly 2015, Part 2 – Chapter 9).

## NoSQL – Not Only SQL

### Hierarchical Databases:

Hierarchical Databases, like JSON, display data in a tree-form[[3]](#endnote-3) with unidirectional parent-child links (Appendix B - Figure 2).

A diagram of a network

Description automatically generated

Figure 4 - Hierarchical Data Structure - MariaDB 2015

Highly efficient for read/write operations due to their structure, they excel in explicit hierarchical relationships like File Systems (MariaDB, 2015). Their limitation lies in an inability to support multi-dimensional or many-to-many relationships.

### Document Database

These don’t store entities like Spreadsheets, Instead, they store structured formats like JSON, BSON[[4]](#endnote-4) or XML[[5]](#endnote-5) (Hoffer, 2019).

Each named, key-addressed record stores its data as a “value”. Unlike hierarchical databases, they facilitate many-to-many relationships by referencing another document’s key.

A close-up of a computer code

Description automatically generated

Figure 5 Document Data Model Representation (Pore, 2018)

They are scalable, can handle complex data and have flexible schemas, making them great for CMS’[[6]](#endnote-6) using *MongoDB* or *Amazon DynamoDB* however they lack proper structure and complex queries can be slow.

### Key-Value Database

KVP [[7]](#endnote-7) databases, fully non-relational, are favored by Twitter (Pandori, 2022) & LinkedIn for rapid lookup in distributed cloud environments.

A screenshot of a computer

Description automatically generated

Figure 6 Example of a Key Value database. (Redis 2023)

Utilizing a hash-table[[8]](#endnote-8) to store data, they offer near instantaneous access but don’t support querying the unstructured values (Hoffer, 2019).

A graph with different colored bars

Description automatically generated

Figure 7 DB Providers & DB Types Read Time for Related Nodes (ArangoDB 2018)

Different from other databases, KVDs excel in speed and scalability, yet lack query versatility.

### Graph Database

Employing graph theory[[9]](#endnote-9) to store, and query relationships, they’re advantageous for interconnected relationships, making them ideal for recommendation engines[[10]](#endnote-10), advanced analytics (Neo4J, n.d.) and fraud detection.

A diagram of a person

Description automatically generated

Figure 8 A simple Graph schema example (Morgante 2020)

They’re unique in NoSQL realm for their relational nature, but struggle with scalability and require complex data modelling.

### Wide-Column Database

Excellent for large datasets with simple querying requirements, favoring tables with numerous, varying column structures (Hoffer, 2019) over RDBMS-like table-joins. However, they have a steep learning curve and struggle with complex querying.

A diagram of a personal information

Description automatically generated

Figure 9 A simple example of a Wide-Column Database

They’re often found in data-warehousing for analytics, aggregation [[11]](#endnote-11)and data mining[[12]](#endnote-12).

### Data Lakes

A data lake is an architectural approach, involving a centralized repository that allows storage of structured and unstructured data at any scale.

A diagram of data processing

Description automatically generated

Figure 10 Data Lake Design Pattern (DataKitchen.io, 2017)

Unlike traditional systems, data lakes store raw data, offering flexibility and the ability to integrate with multiple data sources, but demands robust governance for efficient usage.

They enable diverse analytic methods like machine learning[[13]](#endnote-13).

# Task 2 - Database Modelling

## Scenario Overview

Gemini Jeweler’s (GJ), a producer and wholesaler of Jewelry, implements a pre-approval process for individual item orders by company as well as authorized users for ordering. Items, which come in various color finishes and material selections (presently silver, gold and platinum), are currently housed in two depots. However, GJ envisions future expansion and desires to add more locations.

Currently operating a paper-based system to track stock at each depot, customers approved for ordering by item and material options and prices for each item combination, they desire to move to a web-based CRM [[14]](#endnote-14) & ordering system built on a Relational Database.

## Conceptual Modelling

As the scenario provided contains business requirements with no sample raw data, a ‘top-down’[[15]](#endnote-15) data modelling approach will be implemented.

### Entity Identification

Extracted from the specification, the below figure displays the raw entities, without consideration for relationships or attributes.

A grid of rectangles with text

Description automatically generated

Figure 11 Entity Diagram

### Relation Identification

At a high level, the attribute analysis has revealed a relation of the ‘ItemPrice’ with multiple entities, hence its re-extraction to its own entity.

These relations have started to represent tangible meaning and responsibility of each entity and the control-flow of business processes have emerged.

A diagram of a company

Description automatically generated

Figure 12 Entity Relations Identification

### Conceptual Diagram

Using the relations identified above, annotated in the crows-foot notation, a conceptual diagram has been established.

A diagram of a data flow

Description automatically generated with medium confidence

Figure 13 Conceptual Diagram

A list of assumptions has been made, based on the current state of modelling:

1. The Price entity can be subsumed into our Item Entity
2. GJ Employee’s require zero relations to the greater CRM & Order Management system.
3. Companies can exist without Users but not vice versa.
4. Items are dependent on at least a single Material and Colour entity.

## Logical Modelling

### Attribute Identification

Adapted from Entities discovered in the Conceptual Model, several primitive attributes have been identified & we have assessed the potential of each as Candidate Key (CK)[[16]](#endnote-16).

A screenshot of a computer screen

Description automatically generated

Figure 14 Entity Diagram with Attributes Identified

‘CK’ denotes a Candidate Key, on all entities they are present, a fallback ‘ID’ attribute was preferred as all CK attributes were potentially non-uniquely defining or susceptible to change.

### Primitive Logical Diagram

In the diagram below, the Conceptual Diagram and Attributes identified above have been adapted into a first stage logical model.

A diagram of a computer

Description automatically generated

Figure 15 First Stage Logical Model

User has been renamed Customer to add clarity to the Entity.

Additional attributes have been added as business data requirements have been fully scoped, such as InvoiceAddress & DeliveryAddress on Orders. This process highlighted the requirement of an additional Address Entity.

Multi-valued attributes on existing Entities have been split into single values.

### Removing Many-To-Many Relationships

Below, many-to-many relationships have been extracted into joining tables that preserve the relation between entities while maintaining a clean data structure for easier querying.

A diagram of a company

Description automatically generated with medium confidence

Figure 16 - Many to Many Relationships have been removed.

### Normalization

To fully normalize the database, across the schema, Addresses need to be altered as they exist in two places currently, they also need to be moved from Order to the Company, as the CompanyID is already referenced in the Order. Finally, common attributes representing a Person need to be extracted from Employee and Customer into a single table.

A diagram of a computer

Description automatically generated

Figure 17 - Normalised Database Schema Representation

Optionally, joining tables could be condensed to a Composite key by removing their Primary Key, however, to simplify the data querying process and ensure uniqueness, they will remain.

Alternatively, the ‘Description’ from Material, Item & Colour could be moved to a single Descriptions lookup table, but for the sake of additional complexity the decision was made to retain the individual attributes per table.

If these changes were made, the normalised model would look like below:

A diagram of a computer program

Description automatically generated with medium confidence

Figure 18 - Theoretical Normalised Model if changes were made for pure Normalisation over ease of querying.

### Logical Model

Below is our completed Logical Database Model, complete with attribute data type annotations.

A screenshot of a computer

Description automatically generated

Figure 19 - Logical Data Model

Note: For the sake of simplification each DepotItem’s StockAmount will be static, with no regard to customization. In a real world system, we would require more tables to keep track of stock by Item, Colour & Material combinations.

# Task 3 – Database Design

## Technology Stack.

MySQL 8.1, Docker 3.9 & MySQL Workbench 8.0 CE are the technologies used in the creation of this database. The DB will be entirely constructed via automated scripts & will run on any operating system utilizing containerization with Docker, in the future we can easily replicate this database across any Cloud-based hosting provider.

## Docker Database-Up

A screenshot of a computer

Description automatically generated

This script will create a Docker Container with the MySQL Image, using basic authentication.

A screenshot of a computer screen

Description automatically generated

Docker has successfully created the Container.

A screenshot of a computer

Description automatically generated

## Connecting with MySql Workbench.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Logging into the containerized database instance with the default credentials used, it’s apparent the database has been appropriately created and is ready for scripting.

## Database-UP Script

Using a MySQL SQL Script (Appendix C, Section 1: Database Creation Script), our DB has been scaffolded with Tables, Data Attributes, Indexing, Unique Column Constraints, appropriate Nullability, Defaults, Relations, and Deletion rules such as Cascade.

A computer screen shot of a black background

Description automatically generated

This “CREATE\_TABLE” statement assigns an unsigned integer (meaning it cannot be negative) that automatically increments.

Varchar lengths have been assigned based on an informed decision of the length of data expected for each column.

A business decision was made to not delete Items, rather mark them as Inactive. A Boolean ‘Inactive’ column has been added which defaults to 0 (false).

We require a ‘Name’ to exist by specifying ‘NOT NULL’, but a description is not mandatory.

The ‘Price’ column uses a ‘Decimal(10, 2)’ data annotation so we can specify the degree of rounding we use for non-whole numbers.

A black screen with white text

Description automatically generated

On the ‘ItemMaterial’ table, ‘UNIQUE (`ItemID`, `MaterialID`) ensures that no two records with identical values for each column can exist.

A black screen with white text

Description automatically generated

For the ‘OrderItem’ table, it is anticipated that this will be by far our largest dataset. An ‘unsigned bigint’ has been assigned as the primary key.

The difference between the two is the storage size & maximum range.

Unsigned Integer:

* 4 Bytes of Storage.
* 0 to 4,294,967,295 value range (Almost 4.3 billion).

Unsigned Bigint:

* 8 Bytes of Storage.
* 0 to 18,446,744,073,709,551,615 (18.4 quintillion).

(Morrison II, 2022)

A screen shot of a computer

Description automatically generated

‘Status’ in the ‘CREATE Order’ script defines itself with an Enum. This ensures that the only values that can be stored are the ones we immediately defined within the Enum, we ensure that by default, an order is ‘PLACED’.

‘OrderDate’ and ‘DeliveryDate’ have different data storage requirements. An ‘OrderDate’ automatically assigns the exact time it was created. Delivery Date is intended to be set when the order has been fulfilled.

Finally, frequent lookups are anticipated in this table, so the CompanyID foreign key is Indexed for rapid lookup performance.

A computer screen shot of text

Description automatically generated

Relations are created automatically; cascading deletion rules are established so child records are deleted when a parent record is deleted. However, for data integrity, we do not delete an Order if a Company is deleted, instead, marking the Foreign Key as null.

## Data Seeding

Please see Appendix C – Section 2 for the SQL Scripts to generate initial data.

# Task 4 – Database Queries

# Appendices

## Appendix A:

### Figure 1 – RDBMS Table

A simple representation of a table in a Relational Database Management System.

Adapted from “RDBMS Table Terminologies”, a blogpost by Wentz Wu. 8/07/2019.

### Figure 2 – Stack Overflow Database Environments Survey:

A chart displaying database environments used by professional developers (60,369 respondents). Answers are multiple choice to encapsulate total usage rather than the greatest usage of a specific vender.

Adapted from the 2023 Developer Survey by Stack Overflow.

### Figure 3 – ACID Principles:

A graphic showing an overview of the ACID Principles and their definitions.

Adapted from an independent blogpost by Dave Pinal. 9/12/2007.

### Figure 4 – Hierarchical Data Tree

A simple graphic showing the tree-like structure of hierarchical databases.

Adapted from an article from MariaDB. 06/06/2015.

### Figure 5 – Document Model Example

An image that represents a collection of documents within a document database. Records can contain different data if the overall structure is adhered to.

Adapted from an article from Akshay Pore, 16/2/2018.

### Figure 6 – Key-Value Database Example

An image that shows a human readable example of a Key-Value data store. Using a Name as a Key & a phone number as a Value

Adapted from an article by Redis DB, 06/07/2023.

### Figure 7 – Shortest Path Key-Value Data Access

A chart that displays the read time of related nodes in an inter-connected Social Media database context. ArangoDB(Rocks) is a Key-Value database.

Adapted from an article of benchmarks by ArangoDB, 14/02/2018.

### Figure 8 – Graph Database example Schema

This chart shows a simple graph database, each line represents a relationship between nodes.

Adapted from an article by Victor Morgante on Towards Data Science, 16/09/2020.

### Figure 9 – Wide Column Database Example

An example of a Wide Column store with an example of a User and the data for said user’s information across “column families”.

Adapted from an article from Database Town 29/1/2023.

### Figure 10 – The Data Lake Design Pattern

A design diagram of an architectural approach to integrating and transforming multiple data through a centralized repository known as a ‘Data Lake’.

Adapted from an article from Data Kitchen 17/01/2017.

### Figure 11 – Entity Diagram

A primitive design diagram built to display the Entities found in the specification overview.

### Figure 12 – Attribute Entity Diagram

This diagram, adapted from Figure 11, has been extended to include obvious attributes of the entities, as well as explicit Primary Keys & attributes chosen as Candidate Keys (CKs).

## Appendix B:

### Figure 1 – RDBMS Table:

A diagram of a table

Description automatically generated

A simple example of an RDBMS table and a visual representation of definitions associated with them.

Adapted from an example from W3Resource’s website. 19/08/2022.

### Figure 2 – Hierarchical Data Example:

A screenshot of a computer screen

Description automatically generated

A visual representation of a Hierarchical data structure.

Organization is the Root of the hierarchy, with Departments & Employees as nested child entities. This demonstrates the limitation of the hierarchical data structure, as children can only have a single parent node.

## Appendix C:

### 1: Database Creation Script:

Note: This is left in a raw paste format as I believe it’s significantly easier to read with syntax highlighting.

CREATE TABLE `Colour` (

  `ColourID` integer unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  `Description` varchar(255)

);

CREATE TABLE `ItemColour` (

  `ItemColourID` integer unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  `ItemID` integer unsigned NOT NULL,

  `ColourID` integer unsigned NOT NULL,

  UNIQUE (`ItemID`, `ColourID`)

);

CREATE TABLE `Item` (

  `ItemID` integer unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  `Name` varchar(100) NOT NULL,

  `Inactive` boolean NOT NULL default 0,

  `Description` varchar(255),

  `price` decimal(10, 2)

);

CREATE TABLE `Material` (

  `MaterialID` integer unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  `Description` varchar(255)

);

CREATE TABLE `ItemMaterial` (

  `ItemMaterialID` integer unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  `ItemID` integer unsigned NOT NULL,

  `MaterialID` integer unsigned NOT NULL,

  UNIQUE (`ItemID`, `MaterialID`)

);

CREATE TABLE `Depot` (

  `DepotID` integer unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  `AddressID` integer unsigned NOT NULL

);

CREATE TABLE `DepotItem` (

  `DepotItemId` integer unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  `StockAmount` integer DEFAULT 0,

  `ItemID` integer unsigned NOT NULL,

  `DepotID` integer unsigned NOT NULL

);

CREATE TABLE `OrderItem` (

  `OrderItemId` bigint unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  `ItemID` integer unsigned NOT NULL,

  `OrderNumber` integer unsigned NOT NULL,

  INDEX (`ItemID`, `OrderNumber`)

);

CREATE TABLE `Order` (

  `OrderNumber` integer unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  `Status` ENUM('PLACED', 'SHIPPED', 'COMPLETED', 'CANCELLED') NOT NULL DEFAULT 'PLACED',

  `OrderDate` datetime DEFAULT CURRENT\_TIMESTAMP,

  `DeliveryDate` datetime,

  `CompanyID` integer unsigned,

  INDEX (`CompanyID`)

);

CREATE TABLE `CompanyAllowedItem` (

  `CompanyAllowedItemID` integer unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  `CompanyID` integer unsigned NOT NULL,

  `ItemID` integer unsigned NOT NULL

);

CREATE TABLE `Company` (

  `CompanyID` integer unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  `AccountNumber` varchar(12) NOT NULL UNIQUE,

  `CreditLimit` decimal(10, 2) DEFAULT NULL,

  `InvoiceAddress` integer unsigned,

  `DeliveryAddress` integer unsigned

);

CREATE TABLE `Customer` (

  `CustomerID` integer unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  `CanCreateOrder` boolean NOT NULL DEFAULT 1,

  `CompanyID` integer unsigned NOT NULL,

  `PersonID` integer unsigned NOT NULL

);

CREATE TABLE `Address` (

  `AddressID` integer unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  `AddressLine1` varchar(100) NOT NULL,

  `AddressLine2` varchar(100),

  `AddressLine3` varchar(100),

  `Town` varchar(100),

  `County` varchar(30),

  `Postcode` varchar(30) NOT NULL

);

CREATE TABLE `Person` (

  `PersonID` integer unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  `FirstName` varchar(100),

  `LastName` varchar(100),

  `ContactNumber` varchar(30) NOT NULL,

  `EmailAddress` varchar(100) UNIQUE NOT NULL

);

CREATE TABLE `Employee` (

  `EmployeeID` integer unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

  `IsAdministrator` boolean NOT NULL DEFAULT 0,

  `PersonID` integer unsigned NOT NULL

);

ALTER TABLE `Depot`

ADD FOREIGN KEY (`AddressID`) REFERENCES `Address` (`AddressID`)

ON DELETE CASCADE;

ALTER TABLE `Order`

ADD FOREIGN KEY (`CompanyID`) REFERENCES `Company` (`CompanyID`)

ON DELETE SET NULL;

ALTER TABLE `Customer`

ADD FOREIGN KEY (`PersonID`) REFERENCES `Person` (`PersonID`);

ALTER TABLE `Company`

ADD FOREIGN KEY (`InvoiceAddress`) REFERENCES `Address` (`AddressID`)

ON DELETE CASCADE;

ALTER TABLE `Company`

ADD FOREIGN KEY (`DeliveryAddress`) REFERENCES `Address` (`AddressID`)

ON DELETE CASCADE;

ALTER TABLE `Employee`

ADD FOREIGN KEY (`PersonID`) REFERENCES `Person` (`PersonID`)

ON DELETE CASCADE;

ALTER TABLE `Customer`

ADD FOREIGN KEY (`CompanyID`) REFERENCES `Company` (`CompanyID`)

ON DELETE CASCADE;

ALTER TABLE `CompanyAllowedItem`

ADD FOREIGN KEY (`CompanyID`) REFERENCES `Company` (`CompanyID`)

ON DELETE CASCADE;

ALTER TABLE `CompanyAllowedItem`

ADD FOREIGN KEY (`ItemID`) REFERENCES `Item` (`ItemID`)

ON DELETE CASCADE;

ALTER TABLE `OrderItem`

ADD FOREIGN KEY (`OrderNumber`) REFERENCES `Order` (`OrderNumber`);

ALTER TABLE `OrderItem`

ADD FOREIGN KEY (`ItemID`) REFERENCES `Item` (`ItemID`);

ALTER TABLE `DepotItem`

ADD FOREIGN KEY (`DepotID`) REFERENCES `Depot` (`DepotID`)

ON DELETE CASCADE;

ALTER TABLE `DepotItem`

ADD FOREIGN KEY (`ItemID`) REFERENCES `Item` (`ItemID`);

ALTER TABLE `ItemColour`

ADD FOREIGN KEY (`ColourID`) REFERENCES `Colour` (`ColourID`);

ALTER TABLE `ItemColour`

ADD FOREIGN KEY (`ItemID`) REFERENCES `Item` (`ItemID`);

ALTER TABLE `ItemMaterial`

ADD FOREIGN KEY (`MaterialID`) REFERENCES `Material` (`MaterialID`);

ALTER TABLE `ItemMaterial`

ADD FOREIGN KEY (`ItemID`) REFERENCES `Item` (`ItemID`);

### 2: Seed Data Insertion Scripts:

Note: This script seeds hard-coded data that matches our pre-defined constraints in the above section. Then, following the relations established in our schema, creates the joining tables with values that match the defined relational constraints set by the data itself.

-- Seed Some Pre-defined Colours

INSERT INTO `Colour` (`Description`)

VALUES

('Gold'),

('Silver'),

('Rose Gold'),

('Emerald Green'),

('Ruby Red'),

('Sapphire Blue'),

('Diamond Clear'),

('Amethyst Purple'),

('Onyx Black'),

('Pearl White');

-- Seed Some Pre-defined Materials.

INSERT INTO `Material` (`Description`)

VALUES

('Gold 24K'),

('Gold 18K'),

('White Gold 18K'),

('Sterling Silver'),

('Platinum'),

('Titanium'),

('Copper'),

('Bronze'),

('Stainless Steel'),

('Diamond'),

('Emerald'),

('Ruby'),

('Sapphire'),

('Amethyst'),

('Onyx'),

('Pearl');

-- Seed 10 Addresses using a Fancy & Fake naming convention.

INSERT INTO `Address` (`AddressLine1`, `AddressLine2`, `AddressLine3`, `Town`, `County`, `Postcode`)

VALUES ('123 Jewel St', 'Diamond District', 'Riverside Plaza', 'Gemsville', 'Ruby County', 'WF1 3ST'),

       ('456 Crystal Ave', 'Emerald Enclave', 'Peachtree Heights', 'Gemsville', 'Ruby County', 'JW5 5LC'),

       ('789 Platinum Pl', 'Silver Shore', 'Elmwood Estates', 'Stoneville', 'Pearl Parish', 'ST7 1DC'),

       ('101 Gold Ln', 'Golden Gate', 'Oakwood Gardens', 'Stoneville', 'Pearl Parish', 'OG1 8FC'),

       ('202 Sapphire Sq', 'Ruby Region', 'Sunflower Suites', 'Crystal City', 'Sapphire State', 'CC1 4FH'),

       ('303 Amethyst Alcove', 'Pearl Park', 'Cedarview Terrace', 'Crystal City', 'Sapphire State', 'CC4 6XS'),

       ('404 Diamond Dr', 'Platinum Point', 'Pine Hill Lane', 'Jewel Junction', 'Topaz Territory', 'JJ4 9JG'),

       ('505 Emerald Esplanade', 'Copper Corner', 'Ivybridge Court', 'Jewel Junction', 'Topaz Territory', 'JJ6 7HT'),

       ('606 Ruby Rd', 'Sapphire Street', 'Maple Meadows', 'Gem Grove', 'Emerald Empire', 'GG6 5MR'),

       ('707 Topaz Tr', 'Amethyst Avenue', 'Birchwood Square', 'Gem Grove', 'Emerald Empire', 'GG7 2XL');

-- Seed our 2 Depos with the first 2 addresses.

INSERT INTO `Depot` (`AddressID`)

VALUES (1), (2);

-- Seed 20 aptly named items.

INSERT INTO `Item` (`Name`, `Description`, `price`)

VALUES

('Golden Ring', 'Elegant golden ring.', 250.00),

('Silver Necklace', 'Beautiful silver necklace.', 150.00),

('Platinum Bracelet', 'Stylish platinum bracelet.', 400.00),

('Diamond Earrings', 'Sparkling diamond earrings.', 550.00),

('Emerald Brooch', 'Vintage emerald brooch.', 320.00),

('Ruby Locket', 'Stunning ruby locket.', 350.00),

('Sapphire Pendant', 'Dazzling sapphire pendant.', 270.00),

('Amethyst Choker', 'Charming amethyst choker.', 210.00),

('Pearl Necklace', 'Classic pearl necklace.', 200.00),

('Golden Studs', 'Shiny golden earrings.', 120.00),

('Silver Anklet', 'Dainty silver anklet.', 80.00),

('Platinum Nose Ring', 'Unique platinum nose ring.', 90.00),

('Diamond Tiara', 'Royal diamond tiara.', 650.00),

('Emerald Cufflinks', 'Exquisite emerald cufflinks.', 180.00),

('Ruby Crown', 'Magnificent ruby crown.', 590.00),

('Sapphire Ring', 'Elegant sapphire ring.', 280.00),

('Amethyst Bracelet', 'Alluring amethyst bracelet.', 210.00),

('Pearl Studs', 'Sophisticated pearl earrings.', 100.00),

('Golden Toe Ring', 'Trendy golden toe ring.', 60.00),

('Silver Brooch', 'Elegant silver brooch.', 110.00);

-- For each item, assign 3 Materials.

INSERT INTO `ItemMaterial` (`ItemID`, `MaterialID`) VALUES

(1, 1), (1, 2), (1, 3), (2, 4), (2, 5), (2, 6), (3, 5), (3, 6), (3, 9),

(4, 10), (4, 4), (4, 5), (5, 11), (5, 4), (5, 3), (6, 12), (6, 1), (6, 2),

(7, 13), (7, 4), (7, 5), (8, 14), (8, 4), (8, 6), (9, 16), (9, 1), (9, 3),

(10, 1), (10, 2), (10, 3), (11, 4), (11, 7), (11, 8), (12, 5), (12, 6), (12, 9),

(13, 10), (13, 1), (13, 3), (14, 11), (14, 1), (14, 4), (15, 12), (15, 1), (15, 3),

(16, 13), (16, 4), (16, 1), (17, 14), (17, 4), (17, 6), (18, 16), (18, 1), (18, 3),

(19, 1), (19, 2), (19, 3), (20, 4), (20, 5), (20, 6);

-- For each item, assign 3 Colours

INSERT INTO `ItemColour` (`ItemID`, `ColourID`)

VALUES (1, 1), (1, 2), (1, 3), (2, 2), (2, 4), (2, 5), (3, 3), (3, 4), (3, 6),

(4, 7), (4, 8), (4, 9), (5, 4), (5, 5), (5, 10), (6, 5), (6, 6), (6, 7),

(7, 6), (7, 7), (7, 8), (8, 8), (8, 9), (8, 10), (9, 1), (9, 3), (9, 4),

(10, 2), (10, 3), (10, 10), (11, 3), (11, 6), (11, 7), (12, 4), (12, 7), (12, 8),

(13, 5), (13, 8), (13, 9), (14, 6), (14, 9), (14, 10), (15, 7), (15, 10), (15, 2),

(16, 8), (16, 1), (16, 3), (17, 9), (17, 3), (17, 5), (18, 10), (18, 4), (18, 6),

(19, 1), (19, 7), (19, 8), (20, 2), (20, 9), (20, 10);

-- For each Depot, Assign 10 items with a StockAmount of 100.

INSERT INTO `DepotItem` (`DepotID`, `ItemID`, `StockAmount`) VALUES

(1, 1, 100), (1, 2, 100), (1, 3, 100), (1, 4, 100), (1, 5, 100), (1, 6, 100),

(1, 7, 100), (1, 8, 100), (1, 9, 100), (1, 10, 100),

(2, 11, 100), (2, 12, 100), (2, 13, 100), (2, 14, 100), (2, 15, 100),

(2, 16, 100), (2, 17, 100), (2, 18, 100), (2, 19, 100), (2, 20, 100);

-- Seed 4 Companies, one with a Credit Limit.

INSERT INTO `Company` (`AccountNumber`, `CreditLimit`, `InvoiceAddress`, `DeliveryAddress`)

VALUES ('ACC00123456', NULL, 3, 4), ('ACC00123457', 10000, 5, 6),

('ACC00123458', NULL, 7, 8), ('ACC00123459', NULL, 9, 10);

-- Seed 10 pseudo-random people.

INSERT INTO `Person` (`FirstName`, `LastName`, `ContactNumber`, `EmailAddress`)

VALUES

('John', 'Doe', '+44 7700 900001', 'john.doe@email.com'),

('Jane', 'Smith', '+44 7700 900002', 'jane.smith@email.com'),

('Emily', 'Johnson', '+44 7700 900003', 'emily.johnson@email.com'),

('Michael', 'Brown', '+44 7700 900004', 'michael.brown@email.com'),

('Sophia', 'Taylor', '+44 7700 900005', 'sophia.taylor@email.com'),

('William', 'Jones', '+44 7700 900006', 'william.jones@email.com'),

('Olivia', 'White', '+44 7700 900007', 'olivia.white@email.com'),

('James', 'Harris', '+44 7700 900008', 'james.harris@email.com'),

('Isabella', 'Clark', '+44 7700 900009', 'isabella.clark@email.com'),

('Benjamin', 'Walker', '+44 7700 900010', 'benjamin.walker@email.com');

-- Seed 2 Employees, one Admin.

INSERT INTO `Employee` (`IsAdministrator`, `PersonID`)

VALUES (1, 1), (0, 2);

-- For each of our companies, assign 2 of our people as Customers, one with ordering privileges.

INSERT INTO `Customer` (`CanCreateOrder`, `CompanyID`, `PersonID`)

VALUES (1, 1, 3), (0, 1, 4), (1, 2, 5), (0, 2, 6),

(1, 3, 7), (0, 3, 8), (1, 4, 9), (0, 4, 10);

-- For each company, seed 10 items that they're approved to purchase.

INSERT INTO `CompanyAllowedItem` (`CompanyID`, `ItemID`)

VALUES

(1, 1), (1, 3), (1, 5), (1, 7), (1, 9), (1, 11), (1, 13), (1, 15), (1, 17), (1, 19),

(2, 2), (2, 4), (2, 6), (2, 8), (2, 10), (2, 12), (2, 14), (2, 16), (2, 18), (2, 20),

(3, 1), (3, 2), (3, 4), (3, 6), (3, 7), (3, 9), (3, 11), (3, 13), (3, 15), (3, 17),

(4, 3), (4, 5), (4, 8), (4, 10), (4, 12), (4, 14), (4, 16), (4, 18), (4, 19), (4, 20);

-- Seed 3 orders for each Company. Order 3 for Company 2 will be special, as it will be cancelled because of going over the credit limit.

INSERT INTO `Order` (`Status`, `OrderDate`, `DeliveryDate`, `CompanyID`)

VALUES

('PLACED', '2023-08-10 10:30:00', NULL, 1),

('SHIPPED', '2023-08-05 11:45:00', '2023-08-12 11:45:00', 1),

('COMPLETED', '2023-07-25 12:15:00', '2023-07-30 14:30:00', 1),

('PLACED', '2023-08-08 09:20:00', NULL, 2),

('SHIPPED', '2023-08-04 15:50:00', '2023-08-11 16:00:00', 2),

('PLACED', '2023-07-27 10:10:00', NULL, 2),

('PLACED', '2023-08-09 14:40:00', NULL, 3),

('SHIPPED', '2023-08-03 13:30:00', '2023-08-10 13:45:00', 3),

('COMPLETED', '2023-07-28 15:25:00', '2023-08-02 16:40:00', 3),

('PLACED', '2023-08-07 12:55:00', NULL, 4),

('SHIPPED', '2023-08-02 14:05:00', '2023-08-09 15:15:00', 4),

('COMPLETED', '2023-07-29 10:55:00', '2023-08-03 12:10:00', 4);

-- Seed 3 OrderItems per Order we just created.

INSERT INTO `OrderItem` (`Quantity`, `ItemID`, `DepotItemID`, `ItemMaterialID`, `ItemColourID`, `OrderNumber`)

VALUES

(1, 1, 1, 1, 1, 1), (2, 2, 2, 4, 2, 1), (3, 3, 3, 7, 3, 1),

(1, 4, 4, 10, 7, 2), (2, 5, 5, 13, 4, 2), (3, 6, 6, 16, 5, 2),

(1, 7, 7, 19, 6, 3), (2, 8, 8, 22, 8, 3), (3, 9, 9, 25, 1, 3),

(1, 10, 10, 28, 2, 4), (2, 11, 11, 31, 3, 4), (3, 12, 12, 34, 4, 4),

(1, 13, 13, 37, 5, 5), (2, 14, 14, 40, 6, 5), (3, 15, 15, 43, 7, 5),

(16, 16, 16, 46, 8, 6), (24, 17, 17, 49, 9, 6), (34, 18, 18, 52, 10, 6),

(1, 19, 19, 55, 1, 7), (2, 20, 20, 58, 2, 7), (3, 1, 1, 1, 3, 7),

(1, 2, 2, 4, 4, 8), (2, 3, 3, 7, 5, 8), (3, 4, 4, 10, 6, 8),

(1, 5, 5, 13, 7, 9), (2, 6, 6, 16, 8, 9), (3, 7, 7, 19, 9, 9),

(1, 8, 8, 22, 10, 10), (2, 9, 9, 25, 1, 10), (3, 10, 10, 28, 2, 10),

(1, 11, 11, 31, 3, 11), (2, 12, 12, 34, 4, 11), (3, 13, 13, 37, 5, 11),

(4, 14, 14, 40, 6, 12), (5, 15, 15, 43, 7, 12), (6, 16, 16, 46, 8, 12);

# References

Akshay Pore, 16/02/2018. NoSQL Data Architecture & Data Governance: Everything You Need To Know [article] Available at: <https://www.dataversity.net/nosql-data-architecture-data-governance-everything-need-know/> [Accessed 01/08/2023]

ArangoDB, 14/02/2018. NoSQL Performance Benchmark 2018 – MongoDB, PostgreSQL, OrientDB, Neo4j and ArangoDB. [article] Available at: <https://www.arangodb.com/2018/02/nosql-performance-benchmark-2018-mongodb-postgresql-orientdb-neo4j-arangodb/> [Accessed 03/08/2023]

Connolly, Thomas, 17/4/2015. Database systems: a practical approach to design. Part 2 Chapter 4 – The Relational Model, 6th Edition. Pearson Education. Rotherham, England. [Accessed 31/07/2023]

Connolly, Thomas, 17/4/2015. Database systems: a practical approach to design. Part 2 Chapter 9 – Object-Relational DBMS, 6th Edition. Pearson Education. Rotherham, England. [Accessed 01/08/2023]

Data Kitchen, 17/01/2017. The Data Lake Is a Design Pattern. [article] Available at: <https://medium.com/data-ops/the-data-lake-is-a-design-pattern-888323323c66> [Accessed 05/08/2023]

Database Town, 29/1/2023. Wide Column Database (Use Cases, Examples, Advantages & Disadvantages) [article] Available at: <https://databasetown.com/wide-column-database-use-cases/#wide-column-database-examples> [Accessed 03/08/2023]

Hoffer, Jeff. Modern database management. Part IV Chapter 10 – Big Data Technologies, page 485. Global Edition. Pearson Education. Rotherham, England. [Accessed 01/08/2023]

MariaDB, 06/06/2015. Understanding the Hierarchical Database Model. [article] Available at: <https://mariadb.com/kb/en/understanding-the-hierarchical-database-model/> [Accessed 01/08/2023]

Morrison, Brian, 31/10/2022. MySQL Integers: INT BIGINT and more. [article] Available at: <https://planetscale.com/blog/mysql-data-types-integers> [Accessed 14/08/2023]

Neo4J, 04/02/2015. Why Graph Database? [online] Available at: <https://neo4j.com/why-graph-databases/> [Accessed 03/08/2023]

Redis, 6/07/2023. What is a Key-Value Database. [article] Available at: <https://redis.com/nosql/key-value-databases/> [Accessed 02/08/2023]

Stack Overflow, June 2023. 2023 Developer Survey. [survey] Available at: <https://survey.stackoverflow.co/2023/#section-most-popular-technologies-databases> [Accessed 01/08/2023]

Sahib Pandori, 28/02/2022. Data transfer in Manhattan using RocksDB – Twitter Blogs. [blog] Available at: <https://blog.twitter.com/engineering/en_us/topics/infrastructure/2022/data-transfer-in-manhattan-using-rocksdb> [Accessed 02/08/2023]

Shivang, 15/12/2019. YouTube Database and how does it store so many videos without running out of storage space. [article] Available at: <https://scaleyourapp.com/youtube-database-how-does-it-store-so-many-videos-without-running-out-of-storage-space/> [Accessed 01/08/2023]

Pinal, Dave, 9/12/2007. SQL Server – ACID (Atomicity, Consistency, Isolation, Durability) [blog] Available at: <https://blog.sqlauthority.com/2007/12/09/sql-server-acid-atomicity-consistency-isolation-durability/> [Accessed 01/08/2023]

Victor Morgante, 16/09/2020. What is a graph database? Towards Data Science. [article] Available at: <https://towardsdatascience.com/what-is-a-graph-database-249cd7fdf24d> [Accessed 03/08/2023]

Wentz Wu, 8/07/2019, RDBMS Table Terminologies, RDBMS Table. [online] Available at: <https://wentzwu.com/2019/07/08/rdbms-table-terminologies/> [Accessed 31/07/2023]

W3Resources, 19/08/2022, SQL Table. [online] Available at: <https://www.w3resource.com/sql/sql-table.php> [Accessed 01/08/2023]

W3Schools, n.d. Introduction to XML. [online] Available at: <https://www.w3schools.com/xml/xml_whatis.asp> [Accessed 01/08/2023]

# Glossary

1. SQL – Structured Query Language

   The standard language of accessing and manipulating Relational Database Management Systems. [↑](#endnote-ref-1)
2. ACID Principles:

   Atomicity, Consistency, Isolation & Durability. A set of principles applied to database transactions to enhance the reliability of data, avoid stateful concurrency issues in transactions and eliminate data loss through critical outages. [↑](#endnote-ref-2)
3. Tree Structure

   A data structure that contains a single root node that can have a recursive number of child elements with only one parent. Called a “Tree” due to its triangular visual representation and because each child node can be referred to as a branch, where each branch has its own branches. [↑](#endnote-ref-3)
4. BSON – Binary JSON

   A binary representation of JSON, often storing metadata about the type & length of data. [↑](#endnote-ref-4)
5. XML – Extensible Markup Language

   A file format for storing, transmitting, and reconstructing arbitrary data. It defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. (W3Schools, n.d.) [↑](#endnote-ref-5)
6. CMS – Content Management System

   A software system that creates, manages, and modifies digital content on a website or application.] [↑](#endnote-ref-6)
7. KVP – Key-Value Pair

   A name & value pairing. Sometimes called an attribute-value pair. These are fundamental for data representation in computing systems and applications. [↑](#endnote-ref-7)
8. Hash Table

   In computing, a hash map is an abstract data type that is used to store an unordered collection of key-value pairs. Each key is unique which offers extremely fast data retrieval if you have the address of the key ahead of time. [↑](#endnote-ref-8)
9. Graph Theory

   Graph theory is a mathematical field studying graph structures, modeling pairwise relations between objects, characterized by vertices (nodes) and edges (connections) [↑](#endnote-ref-9)
10. Recommendation Engine

    This is a system that suggest products, services or information to users based on analysis of their behaviour. [↑](#endnote-ref-10)
11. Aggregation

    In data warehousing, aggregation refers to the process of combining or grouping data together to provide summarized, more digestible information. [↑](#endnote-ref-11)
12. Data Mining

    Applying a statistical method to create measurable, analysed data, revealing patterns or trends, contributing to model development and predictive analytics. [↑](#endnote-ref-12)
13. Machine Learning

    A subset of artificial intelligence that enables computers to learn and improve from experience without being explicitly programmed. [↑](#endnote-ref-13)
14. CRM – Customer Relation Management System

    A system to help manage customer data to support sales management. [↑](#endnote-ref-14)
15. Top-Down Approach

    In Data Modelling, a top-down approach means you only have a set of business requirements to begin modelling. The opposite ‘Bottom Up’ approaches are taken when you have access to raw data. [↑](#endnote-ref-15)
16. CK – Candidate Key

    An attribute on an entity that could be a good choice for a Primary Key. [↑](#endnote-ref-16)