Trent Stanley

LMQF353S5

ITDAa3-b22 project

Table of Contents

[Introduction 2](#_Toc77356057)

[Concurrency Control 2](#_Toc77356058)

[Deliverable 1 3](#_Toc77356059)

[1.1 3](#_Toc77356060)

[1.2 4](#_Toc77356061)

[Deliverable 2 5](#_Toc77356062)

[Logical ERD model 5](#_Toc77356063)

[Create database and tables sql query (create base schema.sql) 6](#_Toc77356064)

[Data Dictionary 10](#_Toc77356065)

[Account 10](#_Toc77356066)

[Customer 10](#_Toc77356067)

[Employee 11](#_Toc77356068)

[Office 11](#_Toc77356069)

[Order 12](#_Toc77356070)

[OrderDetail 12](#_Toc77356071)

[Product 13](#_Toc77356072)

[ProductLine 13](#_Toc77356073)

[Deliverable 3 14](#_Toc77356074)

[3.1 14](#_Toc77356075)

[3.2 14](#_Toc77356076)

[3.3 14](#_Toc77356077)

[3.4 15](#_Toc77356078)

[3.5 15](#_Toc77356079)

[References 16](#_Toc77356080)

# Introduction

# Concurrency Control

Concurrency controls ensure the following three criteria:

* Atomicity: a set of operations are grouped as a logical entity. (Oracle, 2021)
* Isolation: this determines how transaction integrity is visible to other systems and users. (GeeksforGeeks, 2019)
* Serializability: A database could be left in an inconsistent state when multiple transactions are running. To keep consistency in the database, a schedule is used to keep track of transaction orders. (Singh, 2018)

The following four protocols used are:

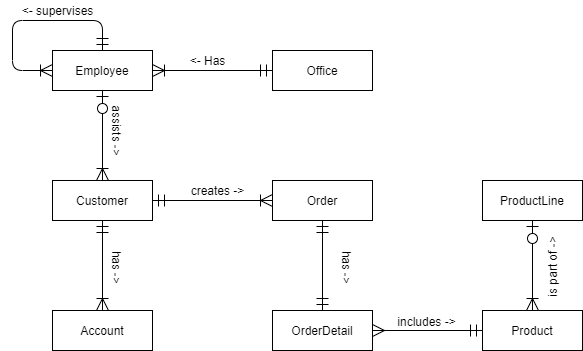
* Two-phase locking Protocol: it applies a lock to data in a transaction while blocking other transactions that are trying to access the same data. At the start, the transaction permission for the lock is requested. The transaction gets all locks. Once a transaction releases a lock, it cannot request more locks. (java T point, 2021)
* Timestamp ordering Protocol: transactions are performed in a sequence based on timestamp. A schedule is made where transactions take part and are serializable. (Goswami, 2019)
* Multi-version concurrency control: The database maintains an old version of itself after something is updated. When a transaction occurs, the correct version is needed to maintain serializability. Some read operations that are rejected in other protocols can now be executed in this protocol using old versions. (Vardhan, 2019)
* Validation concurrency control: mostly used for short transactions. It validates temporary values with actual values and checks view serializability conditions. (Pedamkar, 2021)

# Deliverable 1

## 1.1

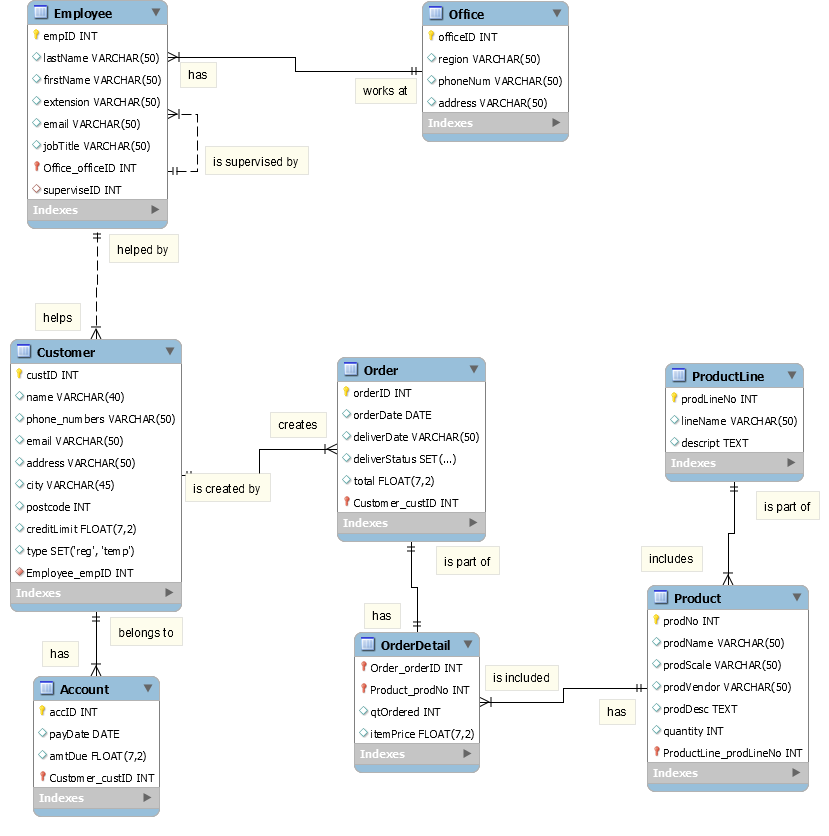
|  |  |  |
| --- | --- | --- |
| Entity name | Relationship | Attributes and attribute domains – key type |
| Customer | * Creates Order: 1:\* * Has Account: 1:\* * Is assisted by Employee: \*:0 | * custID INT - **primary key** * name VARCHAR(40) * phone\_numbers VARCHAR(50) * email VARCHAR(50) * city VARCHAR(50) * postcode INT * address VARCHAR(50) * creditLimit FLOAT(7,2) * type SET (‘reg’, ‘temp’) * password VARCHAR(45) |
| Account | * Belongs to Customer: \*:1 | * accID INT – **primary key** * payDate – DATE * amtDue - FLOAT(7,2) |
| Employee | * assists Customer: 0:\* * supervises Employee: 1:\* * is part of office: \*:1 | * empID INT - **primary key** * officeID INT - **foreign key** * lastName VARCHAR(50) * firstName VARCHAR(50) * extension VARCHAR(50) * email VARCHAR(50) * jobTitle VARCHAR(50) |
| Office | * has Employee: 1:\* | * officeID INT - **primary key** * region VARCHAR(50) * phoneNum VARCHAR(50) * address VARCHAR(50) |
| Order | * Is created by Customer: \*:1 * Has OrderDetail: 1:1 | * orderID INT - **primary key** * custID INT – **foreign key** * orderDate DATE * deliverDate DATE * deliverStatus SET(‘delivered’, ‘to be delivered’) * total FLOAT(7,2) |
| OrderDetail | * Is part of Order: 1:1 * Includes Product: \*:1 | * orderID INT - **foreign key** * prodNo INT - **foreign key** * qtOrdered INT * itemPrice FLOAT(7,2) |
| Product | * Is included in OrderDetail: 1:\* * Is part of ProductLine: \*:0 | * prodNo INT - **primary key** * prodLineNo INT - **foreign key** * prodName VARCHAR(50) * prodScale VARCHAR(50) * prodVendor VARCHAR(50) * prodDesc TEXT * quantity INT |
| ProductLine | * has products: 0:\* | * prodLineNo INT – **primary key** * lineName VARCHAR(50) * descript TEXT |

## 1.2



# Deliverable 2

## Logical ERD model

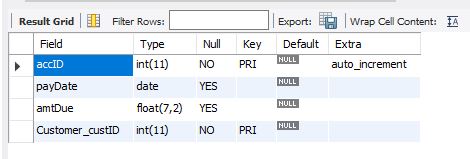


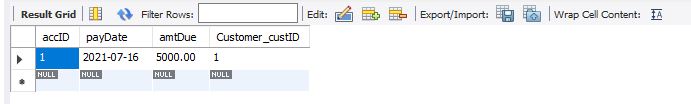
## Create database and tables sql query (create base schema.sql)

|  |
| --- |
| -- MySQL Script generated by MySQL Workbench  -- 07/16/21 17:36:22  -- Model: New Model Version: 1.0  -- MySQL Workbench Forward Engineering  SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;  SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;  SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';  -- -----------------------------------------------------  -- Schema Jarzin  -- -----------------------------------------------------  -- -----------------------------------------------------  -- Schema Jarzin  -- -----------------------------------------------------  CREATE SCHEMA IF NOT EXISTS `Jarzin` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci ;  USE `Jarzin` ;  -- -----------------------------------------------------  -- Table `Jarzin`.`Office`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `Jarzin`.`Office` (  `officeID` INT NOT NULL AUTO\_INCREMENT,  `region` VARCHAR(50) NULL,  `phoneNum` VARCHAR(50) NULL,  `address` VARCHAR(50) NULL,  PRIMARY KEY (`officeID`),  UNIQUE INDEX `officeID\_UNIQUE` (`officeID` ASC))  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `Jarzin`.`Employee`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `Jarzin`.`Employee` (  `empID` INT NOT NULL AUTO\_INCREMENT,  `lastName` VARCHAR(50) NULL,  `firstName` VARCHAR(50) NULL,  `extension` VARCHAR(50) NULL,  `email` VARCHAR(50) NULL,  `jobTitle` VARCHAR(50) NULL,  `Office\_officeID` INT NOT NULL,  `superviseID` INT NULL,  PRIMARY KEY (`empID`, `Office\_officeID`),  UNIQUE INDEX `empID\_UNIQUE` (`empID` ASC),  INDEX `fk\_Employee\_Office1\_idx` (`Office\_officeID` ASC),  INDEX `superviseID\_idx` (`superviseID` ASC),  CONSTRAINT `fk\_Employee\_Office1`  FOREIGN KEY (`Office\_officeID`)  REFERENCES `Jarzin`.`Office` (`officeID`)  ON DELETE NO ACTION  ON UPDATE NO ACTION,  CONSTRAINT `superviseID`  FOREIGN KEY (`superviseID`)  REFERENCES `Jarzin`.`Employee` (`empID`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `Jarzin`.`Customer`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `Jarzin`.`Customer` (  `custID` INT NOT NULL AUTO\_INCREMENT,  `name` VARCHAR(40) NULL,  `phone\_numbers` VARCHAR(50) NULL,  `email` VARCHAR(50) NULL,  `address` VARCHAR(50) NULL,  `city` VARCHAR(45) NULL,  `postcode` INT NULL,  `creditLimit` FLOAT(7,2) NULL,  `type` SET('reg', 'temp') NULL,  `Employee\_empID` INT NOT NULL,  PRIMARY KEY (`custID`),  UNIQUE INDEX `custID\_UNIQUE` (`custID` ASC),  INDEX `fk\_Customer\_Employee1\_idx` (`Employee\_empID` ASC),  CONSTRAINT `fk\_Customer\_Employee1`  FOREIGN KEY (`Employee\_empID`)  REFERENCES `Jarzin`.`Employee` (`empID`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `Jarzin`.`Account`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `Jarzin`.`Account` (  `accID` INT NOT NULL AUTO\_INCREMENT,  `payDate` DATE NULL,  `amtDue` FLOAT(7,2) NULL,  `Customer\_custID` INT NOT NULL,  PRIMARY KEY (`accID`, `Customer\_custID`),  INDEX `fk\_Account\_Customer1\_idx` (`Customer\_custID` ASC),  CONSTRAINT `fk\_Account\_Customer1`  FOREIGN KEY (`Customer\_custID`)  REFERENCES `Jarzin`.`Customer` (`custID`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `Jarzin`.`Order`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `Jarzin`.`Order` (  `orderID` INT NOT NULL AUTO\_INCREMENT,  `orderDate` DATE NULL,  `deliverDate` VARCHAR(50) NULL,  `deliverStatus` SET('delivered', 'to be delivered') NULL,  `total` FLOAT(7,2) NULL,  `Customer\_custID` INT NOT NULL,  PRIMARY KEY (`orderID`, `Customer\_custID`),  INDEX `fk\_Order\_Customer\_idx` (`Customer\_custID` ASC),  CONSTRAINT `fk\_Order\_Customer`  FOREIGN KEY (`Customer\_custID`)  REFERENCES `Jarzin`.`Customer` (`custID`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `Jarzin`.`ProductLine`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `Jarzin`.`ProductLine` (  `prodLineNo` INT NOT NULL AUTO\_INCREMENT,  `lineName` VARCHAR(50) NULL,  `descript` TEXT NULL,  PRIMARY KEY (`prodLineNo`),  UNIQUE INDEX `prodLineNo\_UNIQUE` (`prodLineNo` ASC))  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `Jarzin`.`Product`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `Jarzin`.`Product` (  `prodNo` INT NOT NULL AUTO\_INCREMENT,  `prodName` VARCHAR(50) NULL,  `prodScale` VARCHAR(50) NULL,  `prodVendor` VARCHAR(50) NULL,  `prodDesc` TEXT NULL,  `quantity` INT NULL,  `ProductLine\_prodLineNo` INT NOT NULL,  PRIMARY KEY (`prodNo`, `ProductLine\_prodLineNo`),  UNIQUE INDEX `prodNo\_UNIQUE` (`prodNo` ASC),  INDEX `fk\_Product\_ProductLine1\_idx` (`ProductLine\_prodLineNo` ASC),  CONSTRAINT `fk\_Product\_ProductLine1`  FOREIGN KEY (`ProductLine\_prodLineNo`)  REFERENCES `Jarzin`.`ProductLine` (`prodLineNo`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `Jarzin`.`OrderDetail`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `Jarzin`.`OrderDetail` (  `Order\_orderID` INT NOT NULL,  `Product\_prodNo` INT NOT NULL,  `qtOrdered` INT NULL,  `itemPrice` FLOAT(7,2) NULL,  PRIMARY KEY (`Order\_orderID`, `Product\_prodNo`),  INDEX `fk\_OrderDetail\_Product1\_idx` (`Product\_prodNo` ASC),  CONSTRAINT `fk\_OrderDetail\_Order1`  FOREIGN KEY (`Order\_orderID`)  REFERENCES `Jarzin`.`Order` (`orderID`)  ON DELETE NO ACTION  ON UPDATE NO ACTION,  CONSTRAINT `fk\_OrderDetail\_Product1`  FOREIGN KEY (`Product\_prodNo`)  REFERENCES `Jarzin`.`Product` (`prodNo`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB;  SET SQL\_MODE=@OLD\_SQL\_MODE;  SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;  SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS; |

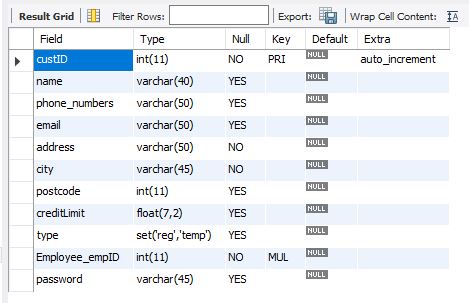
## Data Dictionary

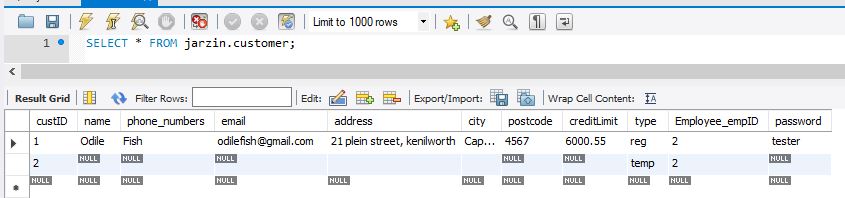
### Account



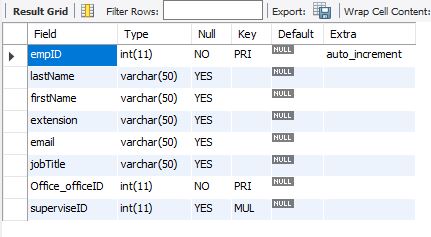


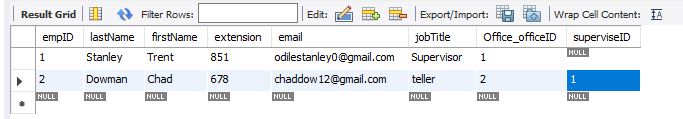
### Customer



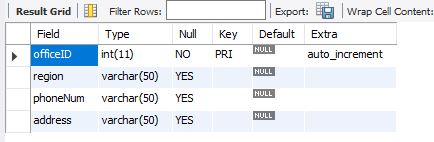


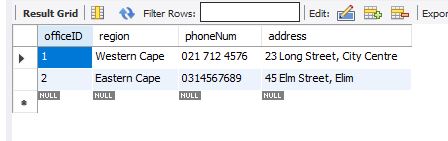
### Employee



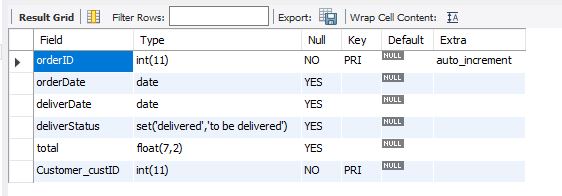


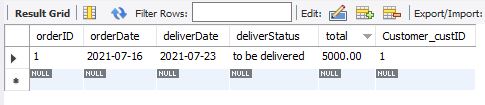
### Office



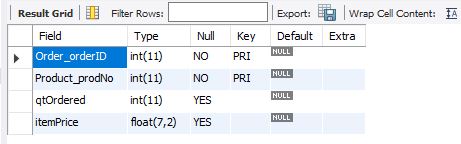


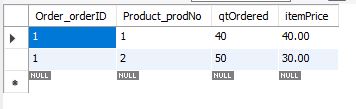
### Order



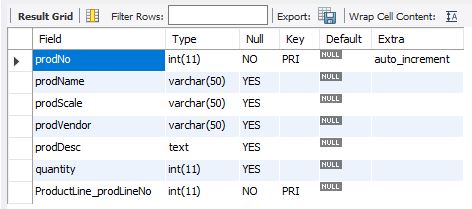


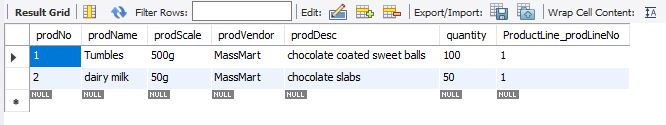
### OrderDetail



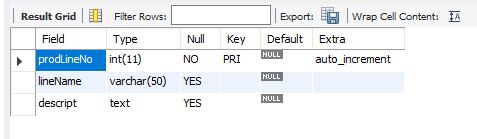


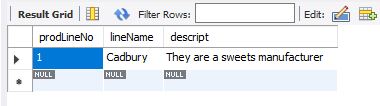
### Product





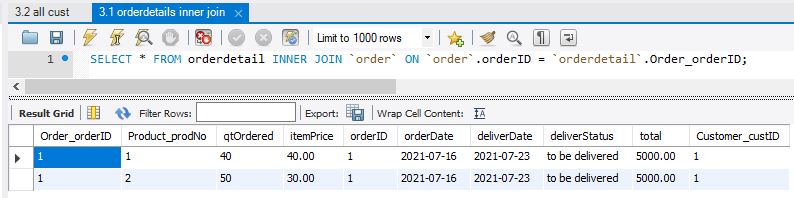
### ProductLine



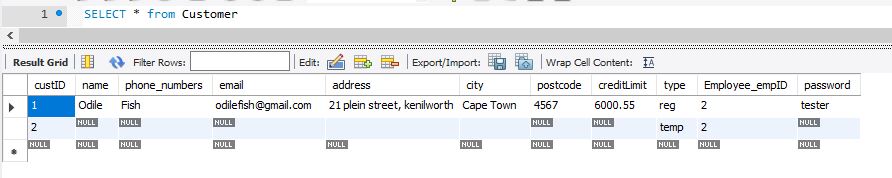


# Deliverable 3

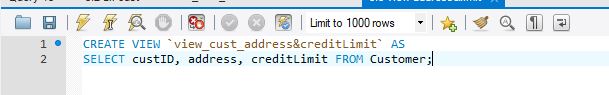
## 3.1

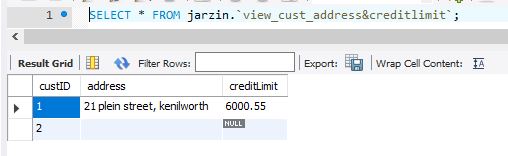


## 3.2

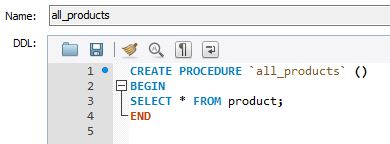


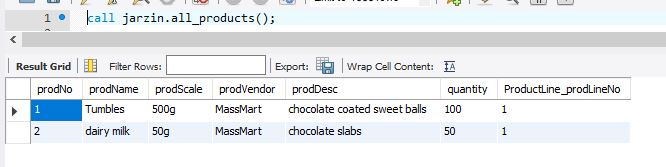
## 3.3



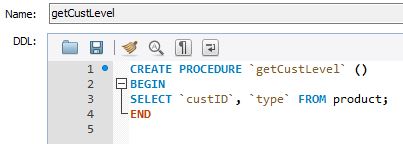


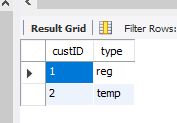
## 3.4





## 3.5





# References

GeeksforGeeks, 2019. *Transaction Isolation Levels in DBMS.* [Online]   
Available at: https://www.geeksforgeeks.org/transaction-isolation-levels-dbms/  
[Accessed 6 July 2021].

Goswami, A., 2019. *Timestamp ordering protocol.* [Online]   
Available at: https://www.includehelp.com/dbms/timestamp-ordering-protocol.aspx  
[Accessed 8 July 2021].

java T point, 2021. *Lock-Based Protocol.* [Online]   
Available at: https://www.javatpoint.com/dbms-lock-based-protocol  
[Accessed 8 July 2021].

Oracle, 2021. *Atomicity.* [Online]   
Available at: https://docs.oracle.com/cd/E17276\_01/html/programmer\_reference/transapp\_atomicity.html  
[Accessed 6 July 2021].

Pedamkar, P., 2021. *Validation Based Protocol in DBMS.* [Online]   
Available at: https://www.educba.com/validation-based-protocol-in-dbms/  
[Accessed 8 July 2021].

Singh, C., 2018. *DBMS Serializability.* [Online]   
Available at: https://beginnersbook.com/2018/12/dbms-serializability/  
[Accessed 6 July 2021].

Vardhan, R., 2019. *Multiversion Concurrency Control Techniques.* [Online]   
Available at: https://www.slideshare.net/rajvardhan7/multiversion-concurrency-control-techniques  
[Accessed 8 July 2021].