

**Asim Faiaz**

**Student ID:3296512**

**Assignment 3**

**SCS Resource Access Database Design Project - Physical Database Design**

Tasks:

Part 1:

* Data Requirements.
* Transaction requirements
  + Data manipulation
  + Queries
* Business Rules.

Part 2:

* EER model
* Documentation
  + Data dictionary

Part 3:

* Reflection piece on Assignment 1 feedback
* EER model mapping
* Normalize using Boyce-Codd Normal Form

Part 4:

* Reflection piece on Assignment 2 feedback
* Write SQL scripts that create the normalised SCS database
* Write SQL statements satisfying the transaction requirements

Table of Contents:

Reflection Piece…………………………………… 5

Data Requirements ………………………………. 6 - 11

Transaction Requirements ……………………… 12

Business Rules ……………………………………13

EER Diagram………………………………………14

Data Dictionary …………………………………….15 - 19

EER Mapping……………………………………… 20 - 22

Normalization……………………………………….23 - 26

SQL script…………………………………………….27 - 45

Reflection Piece:

**Feedback:** Fix the Database Table.

**Response:** I fix the entities Administration and AcquisitionRequest and put all the attributes along with AcquisitionReq from Administration and delete the table.

**Feedback:** Fix the Relation mapping.

**Response:** I did a lot of mistakes in the mapping. After getting the feedback from tutor I correct all the problems. I create two new relations in CourseOffering and Privileges and make it firm. I show them to my tutor and he confirmed that the mapping is good now.

**Feedback:** Normalization process is not right and no list of BCNF

**Response:** I followed the lecture carefully and found out that my normalization was not correct and I tried to make it good. I talked to my tutor about that and he showed me the best way to do that and I followed that and try to make it good.

Data Requirements:

Scenario:

The School of Computer Science (SCS) at the University of Sunshine manages and lends resources to staff and students for assignment and project purposes. The resources include rooms, cameras, speakers, software, phones, etc. Managing such services has become cumbersome and the support services of the School of SCS have requested you to develop a database for their application. The project has been named as “SCS Resource Management”.

Develop a web-based searchable catalogue of all resources. Facilities will be needed to search the catalogue on various criteria including keyword, name, type etc

The loan service provides facilities to issue and to return resources. Resources can be classified as either movable (which can be taken away) or immovable. Resources are loaned to students or staff members of the School, who are also known as borrowers. The different types of borrowers are provided with varying degrees of privileges. The privileges of students will be affected by the courses they are working on.

The School consistently updates its resources. Suggestions for acquiring access to newer and updated materials are elicited from staff and students of SCS. Priority is provided for acquisitions pertaining to the teaching and research needs of the university.

The members should be able to reserve resources (if available) ensuring that they gain access to resources for specific periods of time. Requests for reservations are authorised on a first-come-first-serve basis.

Entities from the data:

1. Resources
   * 1. Movable
     2. Immovable
2. Resources Category
3. Resources Location
4. Members.
   * 1. Students
     2. Staff
5. Course offering
6. Privileges
7. Loan
8. Acquisition
9. Reservation

Requirements of data for entities:

*The first attribute is RESOURCES and Every RESOURCES has the following Entities.*

* Resources

1. Res ID.
2. Description.
3. Status.

*There are two kind of RESOURCES. I am explaining the Attributes and Entities about that below. Those are the subclasses/Child of Resources. They are MOVABLE and IMMOVABLE resources. MOVEABLE resources can be borrowed for a limited time period. It depends on the item. IMMOVABLE resources can also reserve for a limited time period but it should be reserved at least couple of hours ago. Moreover, there should be capacity restriction.*

* Resource Movable

1. Name.
2. Make.
3. Manufacturer.
4. Model.
5. Year.
6. Asset value.

* Resource Immovable

1. Capacity

*Each RESOURCES are categorised as CATEGORY and each CATEGORY has the following Entities mentioning below.*

* Resource Category

1. Cat ID.
2. Name.
3. Description.
4. Max period

*All RESOURCES have a location and that will show where the resource is located. RESOURCE LOCATION is the Attribute for that and the Entities are mentioning below.*

* Resource Location

1. Loc ID.
2. Building.
3. Room.
4. Campus.

*School of SEEC’s staff and students (i.e. students who enrol in courses offered by SEEC) have lending and reservation rights to School’s resources. They known as MEMBERS and This is the name of MEMBERS Attribute. The Entities for MEMBERS are mentioning below. There are two Subclasses/Child of MEMBERS. They are STAFF and STUDENTS. The Entities for STUDENT AND STAFF Attribute mentioned down as well.*

* Members

1. Mem ID.
2. Name.
3. Address.
4. Phone number.
5. Email.
6. Status.

* Student

1. Penalty Point

* Staff

1. Staff Position

*Student members enrol in courses offered by the School. Course information about course offerings and student enrolments are maintained. The Attribute for that is STUDENT’S COURSE OFFERING and the Entities for that Attribute is mentioning below.*

* Course Offering

1. Off ID
2. Course Name.
3. Semester offered.
4. Year offered.

*A course is assigned PRIVILEGES to different categories of resources. The Attribute is named after PRIVILEGES and the Entities are mentioning below.*

* Privileges

1. Priv ID
2. Description.
3. Number of Items

*A member can loan movable resources allowed by his/her privileges. The Attribute is named as LOAN and the Entities are mentioning below as well.*

* Loan

1. Loan ID.
2. Item
3. Pickup date
4. Due date
5. Return date

*A member can request new acquisitions to the School. The Attribute named as ACQUISITION REQUEST and the entities are mentioning below. The administrator of the system assigns a status. The Attribute for that named ADMINISTRATOR and the entities are mentioned below.*

* Acquisition Request

1. Req ID
2. Item name.
3. Item make.
4. Item manufacturer.
5. Item model.
6. Item year.
7. Item description.
8. Items urgency.
9. Admin ID
10. Assign status
11. Fund code.
12. Vendor code.
13. Price.
14. Other request

*Members can reserve resources that their privileges allow them to borrow/book. Once a reservation is made the resource will be booked for pickup/use by the member on the requested date and time. Reservations have the date and time the item is required and a due date & time. The Attribute for that named RESERVATION and the Entities are mentioning below.*

* Reservation

1. Reserve ID.
2. Required time and date.
3. Due time and date.

TRansaction requirement:

Data Manipulation:

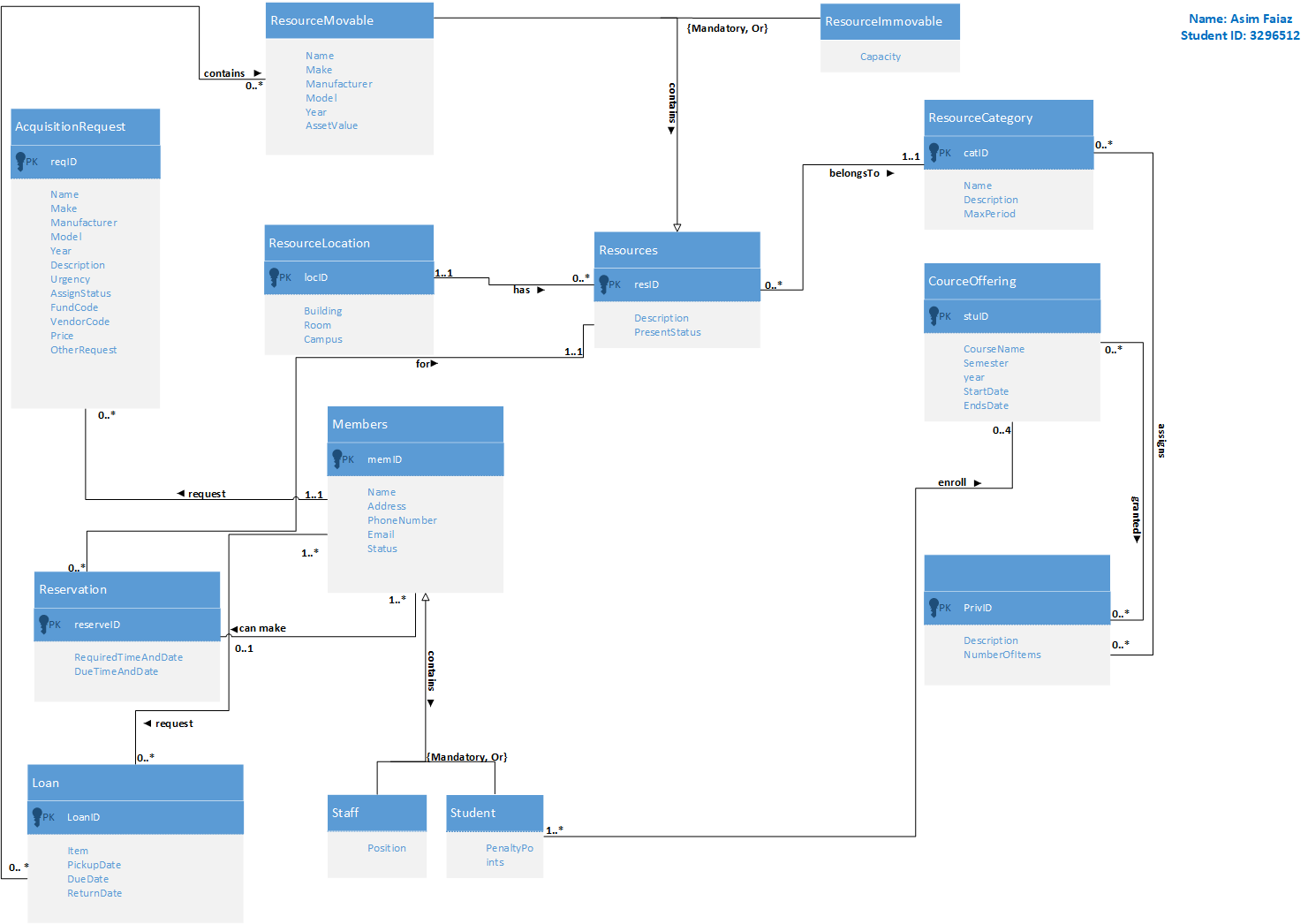
* Insert, update, delete existing resources.
* Insert, update, delete movable resources.
* Insert, update, delete immovable resources.
* Insert, update, delete Resources Catalogue.
* Insert, update, delete Resources Category.
* Insert, update, delete category Resources Location.
* Insert, update, delete existing members.
* Insert, update, delete privileges.
* Insert, update, delete existing loans.
* Insert, update, delete acquisition Request.
* Update administrator.
* Insert, update, delete reservation.
* Update booked.

Queries:

* Search a loaned item based on an employee number, on a particular date.
* Find the student with a late returned item number.
* List all the reservations for a particular item.
* Report of points earned for a particular student during certain period.
* Find resources by using the code.
* Member’s record by using the member ID.
* Display the record who and how many times an immovable item loaned and it should be done by using the Building ID.
* Receipt should be print with every item when borrowed or reserved.
* Administrator resets and amend demerit points.
* Administrator can request amount of points that is in members account and make changes.

Business rules:

* A student’s borrowing privileges are automatically taken away when the current date is later than end date of all his/her enrolled course offerings and update status as “Disabled”.
* A member cannot borrow or reserve more than the maximum number of items specified in his/her privileges at any given time. Penalty for late returns by students and update it.
* There will be a penalty for late return. Each student has 12 points and some fixed amount of points will be reduced from the student account.
* Members have to pay if they damage any items they loaned.
* Members cannot loan/reserve if they have any outstanding charges/ debt.
* Members can request for the extension of the loaned item not more than 3 times.
* A reserved item must pick up by the next day the time members reserved otherwise the reserve will be cancelled.
* Payment can be done by card or cash or PayPal and an amount of surcharge will be applied.
* When a member returns an item, it should be stock in the same place by the office staffs.
* The administrator has rights to reset/amend points.
* Non-cancellation of reservation by member, then 1 demerit point.
* The administrator holds the right to cancel any reservation.
* The duration of borrowing/reservation periods are determined by the category to which the item belongs.

EER Diagram:

Data dictionary:

Entity Types:

|  |  |  |  |
| --- | --- | --- | --- |
| Entity Name | Description | Aliases | Occurrence |
| Resources | Describing resources that members can borrow/reserve | Equipment | When a resource/Equipment taken by a member |
| ResourcesMovable | Describing the movable resources | Equipment | When a resource borrowed by a member |
| ResourcesImmoable | Describing the immovable resources | Equipment | When a resource reserved by a member |
| ResourceCategory | Describing resources are categorised to a category. | Equipment | When resources category used by a member. |
| ResourceLocation | Describing the location of the resources. | Equipment | When resources taken/reserved from different locations. |
| Members | Describing the user of the resources | Staff and students | When members loan a resource |
| Course offering | Describing the courses students enrolled | Students | When student enrolled in a course. |
| Privileges | Describing student members are granted privileges | Students | When student enrolled in a course. |
| Loan | Describing loans that members have made | Resources borrowed | When a loan is made by a member |
| Acquisition Request | Describing acquisition members have made | Members | When members made a request |
| Administrator | Describing the tasks | System | When system requires something. |
| Reservation | Describing the reservation made by members | Members | When reservation made by a member |

Relationship Types:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity Name | Multiplicity | Relationship | Multiplicity | Entity Name |
| Resources | {Mandatory, Or} | has | {Mandatory, Or} | ResourcesMovable |
| {Mandatory, Or} | has | {Mandatory, Or} | ResourcesImmoable |
| 0.. \* | has | 1..1 | ResourceLocation |
| 0.. \* | belongsTo | 1..1 | ResourceCategory |
| 0.. \* | for | 1..1 | Reservation |
| Members | {Mandatory, Or} | contains | {Mandatory, Or} | Students |
| {Mandatory, Or} | contains | {Mandatory, Or} | Staff |
| 1..1 | can make | 0.. \* | Reservation |
| 1..1 | request | 0.. \* | AcquisitionRequest |
| 0.. \* | offered | 1.. \* | Loan |
| ResourceCategory | 1..1 | assigns | 0.. \* | Privileges |
| Privileges | 0.. \* | for | 0.. \* | CourseOffering |
| Student | 1.. \* | enrolled | 0.. 4 | CourseOffering |
| AcquisitionRequest | 0.. \* | contains | 1..1 | Administrator |
| Loan | 0.. \* | contains | 0.. \* | ResourcesMovable |

Attributes:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Entity Name | Attributes | | Description | Data type | Nulls | Multi  Value | | Derived | Default |
| Resources | resID | Unique  ID | | VarChar(20) | N | N | | N | N |
| Description | Resource  Description | | Char(20) | N | N | | N | N |
| Status | able or  disable status | | Char(10) | N | Y | | N | Y |
| ResourceMovable | Name | Item’s name | | VarChar(20) | N | N | | N | N |
| Make | Year made | | Int | N | N | | N | N |
| Manufacturer | Brand name | | VarChar(20) | N | Y | | N | N |
| Model | Unique model | | VarChar(20) | N | N | | N | N |
| Year | Year made | | Int | N | N | | N | N |
| AssetValue | Current market price | | Int | N | Y | | N | N |
| ResourceImmovable | Capacity | Total number can arrange in the room | | Int | N | N | | N | N |
| ResourceCategory | CatID | Unique Code | | VarChar(20) | N | N | | N | N |
| Name | Item Name | | VarChar(20) | N | N | | N | N |
| Description | Item description | | Char(20) | N | Y | | N | N |
| MaxPeriod | For how long one can be borrowed | | DateTime | N | Y | | N | N |
| ResourceLocation | LocId | Unique ID | | VarChar(20) | N | N | | N | N |
| Building | Building Name | | VarChar(20) | N | Y | | N | N |
| Room | The room where the resource is | | VarChar(20) | N | Y | | N | N |
| Campus | Resource Campus | | Char(20) | N | N | | N | N |
| Members | MemID | Unique ID | | VarChar(20) | N | N | | N | N |
| Name | Name | | Char(20) | N | N | | N | N |
| Address | Address | | VarChar(Max) | N | Y | | N | N |
| PhoneNumber | Phone number | | int | N | Y | | N | N |
| Email | Email | | VarChar(20) | N | Y | | N | N |
| Status | Active or disable | | Char(10) | N | N | | N | Y |
| CourseOffering | OffID | Unique ID | | VarChar(20) | N | Y | | N | N |
| CourseName | Course Name | | VarChar(20) | N | N | | N | N |
| Semester | The semester student is doing | | VarChar(10) | N | Y | | N | N |
| Year | Year doing | | Int | N | N | | N | N |
| Privileges | PrivID | Unique ID | | VarChar(20) | N | Y | | N | N |
| Name | Name | | Char(Max) | N | N | | N | N |
| Description | Personal description | | VarChar(Max) | Y | N | | N | N |
| NumberOfItems | Item Limitation | | Int | N | Y | | N | N |
| Loan | LoanID | Which item is loaned ID | | VarChar(20) | N | Y | | N | N |
| Item | Loaned Item Name | | VarChar(20) | N | Y | | N | N |
| PickUpDate | Pick up date and time | | DateTime | N | N | | N | N |
| DueDate | Loaned due date and time | | DateTime | N | N | | N | N |
| ReturnDate | Return date and time | | DateTime | N | N | | N | N |
| Student | Penalty Points | | Demerit points | Int | N | | N | N | N |
| Staff | Position | | Working field | Char(20) | N | | Y | N | N |
|  | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Entity name | Attributes | Description | | Data Type & Length | Nulls | Multi-valued | Derived | | Default |
| AcquisitionRequest | ReqID | Request ID | | VarChar(20) | N | N | N | | N |
| Name | | Item name | VarChar(20) | N | N | | N | N |
| Make | | Item make | VarChar(20) | N | N | | N | N |
| Manufacturer | | Item made by which brand | VarChar(20) | N | Y | | N | N |
| Model | | Item unique model | VarChar(20) | N | N | | N | N |
| Year | | Year made | Int | N | N | | N | N |
| Description | | Item structural description | VarChar(Max) | N | N | | N | N |
| Urgency | | How soon member needs the item. | VarChar(20) | N | Y | | N | Y |
| AssignStatus | | Assign Current status of the item | Char(20) | N | Y | | N | N |
| FundCode | | Fund code number | Int | N | N | | N | N |
| VendorCode | | Vending code number | Int | N | Y | | N | N |
| Price | | Item Price | Int | N | Y | | N | N |
| OtherRequest | | All of the request | Char(Max) | Y | N | | N | N |
| Reservation | ReserveID | | ID number | VarChar(20) | N | N | | N | N |
| RequiredTimeAndDate | | Reservation time and date | DateTime | N | Y | | N | N |
| DueTimeAndDate | | Reserved item due date | DateTime | N | N | | N | N |

rELATIONAL MODEL MAPPING:

|  |
| --- |
| ***Resource:*** (resID, description, status, locID, catID)  Primary key: resID  Foreign key: locID references resourceLocation(locID) on update cascade on delete no action  Foreign key: catID reference resourceCategory(catID) on update cascade on delete no action |
| ***ResourceMovable***: (resID, name, make, manufacturer, model, year, assetValue)  Primary key: ResID  Foreign key: ResID references resource (ResID) on update cascade on delete no action |
| ***ResourceImmovable:*** (resID, capacity)  Primary key: resID  Foreign Key: ResID references resource (ResID) on update cascade on delete no action |
| ***ResourceLocation:*** (locID, building, room, campus)  Primary Key***:*** locID |
| ***ResourcesCatagory:*** (catID, description, maxPeriod)  Primary Key: catID |
| ***Members:*** memID, name, address, phoneNumber, email, status)  Primary key: memID  Alternate Key: phoneNumber, address, email |
| ***Student:*** (memID, penaltyPoint)  Primary key: memID  Foreign Key**:** memID references Members(memID) on update cascade on delete on action |
| ***Staff:*** (memID, field)  Primary Key:memID  Foreign Key: memID references Members(memID) on update cascade on delete on action. |
| ***AcquisitionRequest:*** (reqID, name, make, manufacturer, model, year, description, urgency, assignStatus, fundCode, vendorCode, price, other request, memID)  Primary Key: reqID  Foreign key: memID references Member(memID) on update cascade on delete on action |
| ***Reservation:*** (reserveID, requiredTimeandDate, dueTimeandDate, resID)  Primary key: reserveID  Foreign key: resID references Resource(resID) on update cascade on delete no action  Foreign key: memID references members(memID) on update cascade on delete no action |
| ***Loan:*** (loanID, item, pickUpDate, dueDate, returnDate, resID, memID)  Primary key: loanID  Foreign Key: resID references moveableResource(resID) on update cascade on delete no action  Foreign key: memID references Member(memID) on update cascade on delete no action |
| ***Privileges:*** (privId, name, description, numberOfItems, catID, offID)  Primary Key:privID  Foreign Key**:** catID references resourceCategory(catID) on update cascade on delete no action  Foreign Key**:** offID references courseOffering(offID) on update cascade on delete no action  *New Relation:*  **CoursePrivilege:** (privID, offID)  Primary key: privID, offID  Foreign key: PrivID references Privileges(privID) on update cascade on delete no action  Foreign key: offID references CourseOffering(offID) on update cascade on delete no action |
| ***CourseOffering*** (offID, courseNameAndID, semester, year, memID)  Primary Key**:** offID  Foreign key: memID references Members(memID) on update cascade on delete no action  *New Relation:*  **Enroll:** (offID, memID)  Primary Key: offID, memID  Foreign key: offID references CourseOffering(offID) on update cascade on delete no action  Foreign key: memID reference Member(memID) on update cascade on delete no action |

NormalizeD Relational Schema:

|  |
| --- |
| ***Resource:*** *(ResID, description, status)*  Primary Key: ResID  FD1: ResID 🡪 description, status  As we can see there is no Partial or Transitive dependency here so we can claim that the relation is already in BCNF form. |
| ***ResourceMovable: (***resID, name, make, manufacturer, model, year, assetValue)  Primary Key: resID  FD1: resID 🡪 name, make, manufacturer, model, year, assetValue  As we can see there is no Partial or Transitive dependency here so we can claim that this relation already in BCNF form. |
| ***ResourceImmovable:*** (ResID, capacity)  Primary key: ResID  FD1: resID 🡪 capacity  As we can see here is no Transitive or Partial dependency here so we can claim that the relation is already BCNF form. |
| ***ResourceLocation:*** (locID, building, room, campus)  Primary Key: locID  FD1: locID 🡪building, room, campus  As we can see there is no Transitive dependency or Partial dependency here so we can claim that the relation is already in BCNF form. |
| ***ResourcesCatagory:*** (catId, description, maxPeriod)  Primary Key: catID  FD1: catID 🡪 description, maxPeriod  As we can see there is no Transitive dependency or Partial dependency here so we can claim that the relation is already in BCNF form. |
| ***Members:*** memID, name, address, phoneNumber, email, status)  Primary key: memID  FD1: memID 🡪 name, address, phoneNumber, email, status  As we can see here is no Transitive or Partial dependency so we can claim that the relation is already in BCNF form. |
| ***MembersStudent:*** (memID, penaltyPoints)  Primary key: memID  FD1: memID 🡪 penaltyPoints  As we can see here is no Transitive or Partial dependency so we can claim that the relation is already in BCNF form. |
| ***MemberStaff:*** (memID, position)  Primary Key:memID  FD1: position 🡪 memID  As we can see here is no Transitive or Partial dependency so we can claim that the relation is already in BCNF form. |
| ***AcquisitionRequest:*** (reqID, name, make, manufacturer, model, year, description, urgency, assignStatus, fundCode, vendorCode, price, otherRequest)  Primary Key: reqID  FD1: reqID 🡪 name, make, manufacturer, model, year, description, urgency, assignStatus, fundCode, vendorCode, price, otherRequest  As we can see there is no Transitive or Partial Dependency so we can claim that the relation is in BCNF |
| ***Reservation:*** (reserveID, requiredTimeandDate, dueTimeandDate)  Primary key: reservationID  FD1:reserveID 🡪 requiredTimeandDate, dueTimeandDate  As we can see there is no Transitive dependency or Partial dependency here so we can claim that the relation is already in BCNF form. |
| ***Loan:*** (loanID, item, pickDate, dueDate, returnDate)  Primary key: loanID  FD1: loanID 🡪item, pickUpDate, dueDate, returnDate  As we can see there is no Transitive dependency or Partial dependency here so we can claim that the relation is already in BCNF form. |
| ***Privileges:*** (PrivID, name, description, numberOfItems)  Primary Key:PrivID  FD1: PrivID 🡪 name, description, numberOfItems  As we can see there is no Transitive dependency or Partial dependency here so we can claim that the relation is already in BCNF form. |
| ***CourseOffering*** (offID, courseNameAndID, semester, year, memID)  Primary Key: offID  Foreign Key: memID references Members(memID) on update cascade on delete no action  FD1: offID 🡪 courseNameandID  FD2: offID, memID, semester 🡪 year  As we can see there is a Partial dependency in FD1.  **Course** (offID, courseName)  Primary Key: offID  **CourseOffeings:** (offID, semester, year, memID)  Primary key: offID  Foreign Key: memID references Members (memID) on update on delete cascade.  Foreign Key: offID references Course(offID) on update on delete cascade.  Now both relations are in 2NF |

List od BCNF relations:

* Resource
* Resource Immovable
* ResourceLocation
* ResourceCatagory
* Members
* MembersStudent
* MembersStaff
* Reservation
* Loan
* Privileges

Physical database:

SQL SCEIPT:

DROP DATABASE IF EXISTS SCSUniOfSunshine

GO

CREATE DATABASE SCSUniOfSunshine

go

USE SCSUniOfSunshine

go

use master

GO

--CREATE TABLES

DROP TABLE Enroll

DROP TABLE CoursePrivilege

DROP TABLE Privileges

DROP TABLE CourseOfferings

DROP TABLE loan

DROP TABLE reservation

DROP TABLE AcquisitionRequest

DROP TABLE MembersStaff

DROP TABLE MembersStudent

DROP TABLE Members

DROP TABLE ResourceImmovable

DROP TABLE ResourceMoveable

DROP TABLE SResource

DROP TABLE ResourceCategory

DROP TABLE ResourceLocation

--ResourceLocation Table

CREATE TABLE ResourceLocation(

locID VarChar(20) NOT NULL, --Location Unique ID as PK

building VarChar(20) NOT NULL, --Building Name AND Number

room VarChar(20) NOT NULL, --Room name and Number

campus Char(20) NOT NULL, --Name of the campus

PRIMARY KEY(locID))

GO

--ResourceCatagory Table

CREATE TABLE ResourceCategory(

catID VarChar(20) NOT NULL, --Catagory Unique ID as PK

name VarChar(30) NOT NULL, --Name of the Item

CaDescription VarChar(30) NOT NULL, -- Description of Catagory

maxPeriod INT NOT NULL, --Max period to hold or reserve

PRIMARY KEY (catID))

GO

--Resource Table

CREATE TABLE SResource(

resID VarChar(20) NOT NULL, --Unique Resource ID

locID VarChar(20), --Location unique ID as Foreign key

catID VarChar(20), --Catagory unique ID as Foreign key

ReDescription Char(20) NOT NULL, --Resource Description

ReStatus Char(10) DEFAULT 'AVAILABLE' CHECK (ReStatus IN ('AVAILABLE','OCCUPIED','DAMAGED')) NOT NULL, --Resource Able/Disable status

PRIMARY KEY(ResID),

FOREIGN KEY(locID) REFERENCES ResourceLocation(locID) ON UPDATE CASCADE ON DELETE NO ACTION,

FOREIGN KEY(catID) REFERENCES ResourceCategory(catID) ON UPDATE CASCADE ON DELETE NO ACTION,

);

GO

--ResourceMovable Table

Create table ResourceMoveable(

resID VarChar(20) NOT NULL, --Unique Resource ID and Foreign key as well

name VarChar(20) NOT NULL, --Name of the Item

Manufacturer VarChar(20) NOT NULL, --Manufacturer of the Resource

Make Char(20) NOT NULL, --Counrty make

Model VarChar(20) NOT NULL, --Name of the model

RmYear VarChar(20) NOT NULL, --Built year

AssetValue Int, --Asset Value of the Resource

PRIMARY KEY(ResID),

Foreign Key (ResID) references SResource(ResID) on update cascade on delete no action

);

GO

--ResourceImmovable Table

CREATE TABLE ResourceImmovable(

ResID VarChar(20) Not Null, --Unique Resource ID and Foreign key as well

Capacity Int, --Max capacity of the resource

PRIMARY KEY (ResID),

FOREIGN KEY (ResID) REFERENCES SResource(ResID) ON UPDATE CASCADE ON DELETE NO ACTION,

);

GO

--Members Table

CREATE TABLE Members(

MemID VarChar(20) NOT NULL, --Unique member ID

Name Char(20) NOT NULL, --Name of the Member

MeAddress VarChar(100), --Address of the member

PhoneNumber Varchar(15) NOT NULL, --Phone Number of the member

Email VarChar(20), --Email of Member

MeStatus Char(10) DEFAULT 'ACTIVE' CHECK (MeStatus IN ('ACTIVE','NOT-ACTIVE')) NOT NULL, --Activity status of Member

PRIMARY KEY (MemID))

GO

--MembersStudent Table

CREATE TABLE MembersStudent(

MemID VarChar(20) NOT NULL, --Unique member ID

penaltyPoints Int --Total Penalty points of Students

PRIMARY KEY (MemID),

FOREIGN KEY (MemID) REFERENCES Members(MemID) ON UPDATE CASCADE ON DELETE NO ACTION,

);

GO

----MembersStudent Table

CREATE TABLE MembersStaff(

MemID VarChar(20) NOT NULL, --Unique member ID

Position Char(20), --Working field of Staff member

PRIMARY KEY (MemID),

FOREIGN KEY (MemID) REFERENCES Members(MemID) ON UPDATE CASCADE ON DELETE NO ACTION,

);

GO

-- AcquisitionRequest Table

CREATE TABLE AcquisitionRequest(

reqID VarChar(20) NOT NULL, --Unique ID

MemID VarChar(20) NOT NULL, --Member unique ID as foreign key

name Char(20) NOT NULL, --Name of the requested item

description Char(20) NOT NULL, --Description of the requested item

manufacturer VarChar(20) NOT NULL, --Manufacturer of the requested item

make Char(20) NOT NULL, --Cunntry make of the item

model VarChar(20) NOT NULL, --Model of the item

ArYear VarChar(20), --What year the item was made

price Int, --Market price of the item

urgency VarChar(20) DEFAULT 'NOW' CHECK (urgency IN ('NOW','BOOK ON AVAILABLE')), --How urgent the item for who requested

assignStatus Char(20), --Current status of the item

fundCode VarChar(20), --FundCode of the Item

vendorCode VarChar(20), --Vendor Code of the item

otherRequest Char(50) NULL, --Special request made by who want the item

PRIMARY KEY (reqID),

FOREIGN KEY(MemID) REFERENCES Members(MemID) ON UPDATE CASCADE ON DELETE NO ACTION

);

GO

--Reservation Table;

CREATE TABLE Reservation(

reserveID VarChar(20) NOT NULL, --Unique ID

resID VarChar(20) NOT NULL, --Resource Unique ID as Foreign Key

MemID VarChar(20) NOT NULL, --Members Unique ID as Foreign key

requiredTimeAndDate Date NOT NULL, --Reservation time and date

DueTimeAndDate Date NOT NULL, --Reservation end time and date

PRIMARY KEY (reserveID),

FOREIGN KEY (MemID) REFERENCES Members(MemID) ON UPDATE CASCADE ON DELETE NO ACTION,

FOREIGN KEY (ResID) REFERENCES SResource(ResID) ON UPDATE CASCADE ON DELETE NO ACTION,

);

Go

--Loan Table

CREATE TABLE Loan(

loanID VarChar(20) NOT NULL, --UniqueID

resID VarChar(20), --Resource Unique ID as Foreign key

MemID VarChar(20), --Member Unique ID as foreign key

item VarChar(20) NOT NULL, --Item Catagory

pickDate Date NOT NULL, --Pick up date

dueDate Date NOT NULL, --Due Date

returnDate Date, --Return Date

PRIMARY KEY(loanID),

FOREIGN KEY (ResID) REFERENCES SResource(ResID) ON UPDATE CASCADE ON DELETE NO ACTION,

FOREIGN KEY (MemID) REFERENCES Members(MemID) ON UPDATE CASCADE ON DELETE NO ACTION,

);

GO

--CourseOfferings Table

CREATE TABLE CourseOfferings(

offID VarChar(20) NOT NULL, --Unique ID

MemID VarChar(20), --Member unique ID as foreign key

courseNameAndID VarChar(20) NOT NULL, --Name of the Course

semester VarChar(20) NOT NULL, --Name of the Course

OffYear VarChar(20) NOT NULL, --Year offered the course

PRIMARY KEY(offID),

FOREIGN KEY (MemID) References Members(MemID) ON UPDATE CASCADE ON DELETE NO ACTION,

);

GO

--Privileges Table

CREATE TABLE Privileges(

privID VarChar(20) NOT NULL, --Unique ID

catID VarChar(20), --Catagory Unique ID as Foreign Key

offID Varchar(20), --Student Unique ID as Foreign key

PDescription Char(50) NULL, --What kind of privilege is that

numberOfItems VarChar(10) NOT NULL, --How many itemes can be loaned

PRIMARY KEY(PrivID),

FOREIGN KEY (catID) REFERENCES ResourceCategory(catID) ON UPDATE CASCADE ON DELETE NO ACTION,

FOREIGN KEY (offID) REFERENCES CourseOfferings(offID) ON UPDATE CASCADE ON DELETE NO ACTION,

);

GO

--CoursePrivilege Table

CREATE TABLE CoursePrivilege(

privID Varchar(20) NOT NULL, --Unique Privileges ID

offID VarChar(20) NOT NULL, --Unique CourseOffering ID

PRIMARY KEY(privID, offID),

FOREIGN KEY (privID) references Privileges(PrivID) ON UPDATE CASCADE ON DELETE cascade,

);

GO

--Enroll Table

CREATE TABLE Enroll(

offID VarChar(20) NOT NULL, --Unique CourseOffering ID

MemID VarChar(20) NOT NULL, --Unique Member ID

PRIMARY KEY (offID, MemID),

);

GO

--DATA Creation

--Insert ResourceLocation

INSERT INTO ResourceLocation VALUES

('L0','ICT', 'ICT337','CALLAGHAN'),

('L1','HUNTER', 'HP197', 'CALLAGHAN'),

('L2', 'ES', 'ES101', 'CALLAGHAN')

GO

--Insert ResourceCategory

INSERT INTO ResourceCategory VALUES

('C0', 'CAMERA','ALL KIND OF CAMERAS','6'),

('C1', 'LAPTOP','ALL KIND OF LAPTOPS', '6'),

('C2', 'SPEAKER','ALL KIND OF HEADPHONES','6')

GO

--Insert Resource

INSERT INTO SRESOURCE VALUES

('MVR10','L0' ,'C0' , 'CANON 60D CAMERA', 'OCCUPIED' ),

('MVR20', 'L1', 'C1' , 'DELL LAPTOP', 'AVAILABLE'),

('MVR30', 'L2', 'C2' , 'BOSS SPEAKER', 'DAMAGED'),

('IMVR40', 'L1', 'C2', 'RADIO STATION', 'AVAILABLE'),

('IMVR50', 'L2', 'C1' , 'STUDIO', 'OCCUPIED'),

('IMVR60', 'L0', 'C1' , 'MEETING ROOM', 'OCCUPIED')

GO

--Insert ResourceMovable

INSERT INTO ResourceMoveable VALUES

('MVR10', 'CAMERA', 'CANON', 'THAILAND','D60', '2006', '650'),

('MVR20', 'LAPTOP', 'DELL','AUSTRALIA' , 'R60', '2010', '1650'),

('MVR30', 'SPEAKER', 'BOSS','CHINA', 'H60', '2016', '250')

GO

--Insert ResourceImmovable

INSERT INTO ResourceImmovable VALUES

('IMVR40', '40'),

('IMVR50', '50'),

('IMVR60', '60')

GO

--Insert Members

INSERT INTO Members VALUES

('MSTU1','ASIM FA', '2 JESMOND STREET', '012410', 'AF@GMAIL.COM','ACTIVE'),

('MSTU2','OUDOM MO', '9 HERALD STREET', '019510', 'OM@GMAIL.COM','ACTIVE'),

('MSTU3','JON SMITH', '19 HEXA STREET', '082950', 'JS@GMAIL.COM','NOT-ACTIVE'),

('MSTF4','ANANDA KA', '50 KING STREET', '014569', 'AK@GMAIL.COM','ACTIVE'),

('MSTF5','SUHUAU LU', '39 HUNTER STREET', '017890', 'SL@GMAIL.COM','ACTIVE'),

('MSTF6','SHAMUS KO', '18 COBRA STREET', '012512', 'SK@GMAIL.COM','NOT-ACTIVE')

GO

--INSERT STUDENT

INSERT INTO MembersStudent VALUES

('MSTU1','0'),

('MSTU2','5'),

('MSTU3','9')

GO

--INSERT STAFF

INSERT INTO MembersStaff VALUES

('MSTF4','TUTOR'),

('MSTF5', 'LECTURER'),

('MSTF6', 'CO-ORDINATOR')

GO

--INSERT ACQREQUEST

INSERT INTO AcquisitionRequest VALUES

('REQ1','MSTU1' , 'CAMERA', 'CANON D60 CAMERA', 'CANON', 'THAILAND', 'D60', '2020', '650', 'NOW', 'IN STOCK', 'FUND001', 'VEND001', 'NULL' ),

('REQ2','MSTF4' , 'LAPTOP', 'DELL R60 LAPTOP', 'DELL', 'AUSTRALIA', 'R60', '2019', '1650', 'NOW', 'NOT IN STOCK', 'FUND002', 'VEND002', 'NULL' ),

('REQ3', 'MSTU3' , 'SPEAKER', 'BOSS H60 SPEAKER', 'BOSS', 'CHINA', 'H60', '2020', '250', 'NOW', 'NOT IN STOCK', 'FUND003', 'VEND003', 'NULL' )

GO

--INSERT RESERVATION

INSERT INTO Reservation VALUES

('Rev1','IMVR50','MSTU1', '2020-06-11', '2020-06-11'),

('Rev2','IMVR40','MSTF4', '2019-07-11', '2019-07-11'),

('Rev3','IMVR60','MSTF5', '2020-05-11', '2020-05-11')

GO

--INSERT LOAN

INSERT INTO Loan VALUES

('LN1','MVR10', 'MSTU1', 'CAMERA', '2020-06-11' ,'2020-06-15','2020-06-15'),

('LN2','MVR20', 'MSTU2', 'LAPTOP', '2020-05-11' ,'2020-05-15','2020-06-02'),

('LN3','MVR30', 'MSTU3', 'SPEAKER', '2020-06-18' ,'2020-06-22','2020-06-24')

GO

--INSERT COURSE OFFERINGS

INSERT INTO CourseOfferings VALUES

('OFF1','MSTU1','INFT1004', 'SEM 1', '2020'),

('OFF2','MSTU2','COMP1140', 'SEM 2', '2020'),

('OFF3','MSTU3','SENG2280', 'SEM 1', '2020')

GO

--INSERT PRIVILEGES

INSERT INTO Privileges VALUES

('PR1', 'C0', 'OFF3', 'ONLY FOR SENG STUDENTS', '1'),

('PR2', 'C2', 'OFF2', 'ONLY FOR INFT STUDENTS', '1'),

('PR3', 'C1', 'OFF1', 'ONLY FOR COMP STUDENTS', '1')

GO

--INSERT COURSE PRIVILEGES

INSERT INTO CoursePrivilege VALUES

('PR1','OFF1'),

('PR2','OFF2'),

('PR3','OFF3')

GO

--INSERT INTO ENROLL

INSERT INTO Enroll VALUES

('OFF1', 'MSTU1'),

('OFF2', 'MSTU2'),

('OFF3', 'MSTU3')

GO

SELECT \* FROM SResource

SELECT \* FROM ResourceCategory

SELECT \* FROM ResourceLocation

SELECT \* FROM ResourceMoveable

SELECT \* FROM ResourceImmovable

SELECT \* FROM Members

SELECT \* FROM MembersStudent

SELECT \* FROM MembersStaff

SELECT \* FROM AcquisitionRequest

SELECT \* FROM Reservation

SELECT \* FROM Loan

SELECT \* FROM CourseOfferings

SELECT \* FROM Privileges

SELECT \* FROM CoursePrivilege

SELECT \* FROM Enroll

SELECT \* FROM Members

SELECT \* FROM AcquisitionRequest

SELECT \* FROM Reservation

------------------------------------------------------------

--Ques NO 1

--Print the name of student(s) who has/have enrolled in the course with course id xxx.

--COURSE ID = COMP1140

SELECT \* FROM Members

SELECT \* FROM CourseOfferings

select m.Name

from Members m, CourseOfferings c

where m.MemID = c.MemID and courseNameAndID ='COMP1140'

--------------------------------------------------------------

--Ques NO 2

--Print the maximal number of speakers that the student with name xxx can borrow. The student is enrolled in the course with course id yyy. Note: speaker is a category

--NAME = OUDOM MO COURSEnAMEaNDiD = COMP1140 CATAGORY = SPEAKER

SELECT \* FROM CourseOfferings

SELECT \* FROM Privileges

SELECT \* FROM ResourceCategory

SELECT \* FROM Members

select p.numberOfItems

from ResourceCategory c, Privileges p, CourseOfferings o, Members m

where m.memId=o.MemID and c.catID=p.catID and m.name='OUDOM MO' and o.courseNameAndID='COMP1140' and c.name='SPEAKER'

-----------------------------------------------------------------

--Question no 3

--For a staff member with id number xxx, print his/her name and phone number, the total number of acquisition requests and the total number of reservations that the staff had made in 2019.

SELECT \* FROM Members

SELECT \* FROM AcquisitionRequest

SELECT \* FROM Reservation

select m.name, m.phoneNumber, count(a.ArYear) as TotalRequest, count(r.requiredTimeAndDate) as TotalReservation

from Members m, AcquisitionRequest a, Reservation r

where m.MemID = a.MemID and m.MemID = r.MemID and a.ArYear = '2019' and r.requiredTimeAndDate = '2019-07-11'

group by m.Name, m.PhoneNumber

-----------------------------------------------------------------

--Que no 4

--Print the name(s) of the student member(s) who has/have borrowed the category with the name of camera, of which the model is xxx, in this year.

--MODEL = D60 ITEM = CAMERA datePart code from google(reference)

SELECT \* FROM ResourceMoveable

SELECT \* FROM Members

SELECT \* FROM Loan

Select m.name

From members m, ResourceMoveable r, loan l

Where m.MemID = l.MemID and r.resID = l.resID and r.Model = 'D60' and l.item = 'CAMERA' and datepart(year, pickDate)=datepart(year, GETDATE())

------------------------------------------------------------------

--que no 6

--For each of the three days, including May 1, 2020, June 5, 2020 and September 19, 2020, print the date, the name of the room with name xxx, and the total number of reservations made for the room on each day.

--USING THE DATES FROM MY DATABASE

SELECT \* FROM Reservation

SELECT \* FROM ResourceLocation

SELECT \* FROM SResource

SELECT l.room, r.requiredTimeAndDate, count(r.requiredTimeAndDate) as TotalReservation

From ResourceLocation l, Reservation r, SResource s

where r.resID = s.resID and s.locID = l.locID and r.requiredTimeAndDate = '2020-06-11' and r.requiredTimeAndDate = '2019-07-11' and r.requiredTimeAndDate = '2020-05-11' and l.room = 'ICT337'

GROUP BY l.room, r. requiredTimeAndDate

---------------------------------------------------------------------

--Que no. 5

--Find the moveable resource that is the mostly loaned in the current month. Print the resource id and resource name.

--datepart code from google(reference)

select m.resID, m.name, count(m.resId) as MostlyLoaned

From ResourceMoveable m, Loan l

where m.resID=l.resID and datepart(month, pickDate)=datepart(month, GETDATE())

group by m.resID, m.name

Having count(m.resID)>= all (select count(resID) from Loan

where datepart(month, pickDate)=datepart(month, GETDATE())

group by resID)

------------------------------------------------------------------------

END OF ASSIGNMENT