Final Project Documentation
DATABASES COMP 353
WINTER 2018

Project Done by team ezc353_4:

Natalia Dimitrova, 24662407
Max Malderle, 26562906
Javier Fernandez-Rial Portela, 40044982
Eve Raya, 40016485

11th of April 2018
Concordia University

Faculty of Engineering and Computer Science
Expectations of Originality

This form sets out the requirements for originality for work submitted by students in the Faculty of Engineering and Computer Science. Submissions such as assignments, lab reports, project reports, computer programs and take-home exams must conform to the requirements stated on this form and to the Academic Code of Conduct. The course outline may stipulate additional requirements for the course.

1. Your submissions must be your own original work. Group submissions must be the original work of the students in the group.
2. Direct quotations must not exceed 5% of the content of a report, must be enclosed in quotation marks, and must be attributed to the source by a numerical reference citation[1]. Note that engineering reports rarely contain direct quotations.
3. Material paraphrased or taken from a source must be attributed to the source by a numerical reference citation.
4. Text that is inserted from a web site must be enclosed in quotation marks and attributed to the web site by numerical reference citation.
5. Drawings, diagrams, photos, maps or other visual material taken from a source must be attributed to that source by a numerical reference citation.
6. No part of any assignment, lab report or project report submitted for this course can be submitted for any other course.
7. In preparing your submissions, the work of other past or present students cannot be consulted, used, copied, paraphrased or relied upon in any manner whatsoever.
8. Your submissions must consist entirely of your own or your group's ideas, observations, calculations, information and conclusions, except for statements attributed to sources by numerical citation.
9. Your submissions cannot be edited or revised by any other student.
10. For lab reports, the data must be obtained from your own or your lab group's experimental work.
11. For software, the code must be composed by you or by the group submitting the work, except for code that is attributed to its sources by numerical reference.

You must write one of the following statements on each piece of work that you submit:
For individual work: **"I certify that this submission is my original work and meets the Faculty's Expectations of Originality"**, with your signature, I.D. #, and the date.
For group work: **"We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality"**, with the signatures and I.D. #s of all the team members and the date.

A signed copy of this form must be submitted to the instructor at the beginning of the semester in each course.

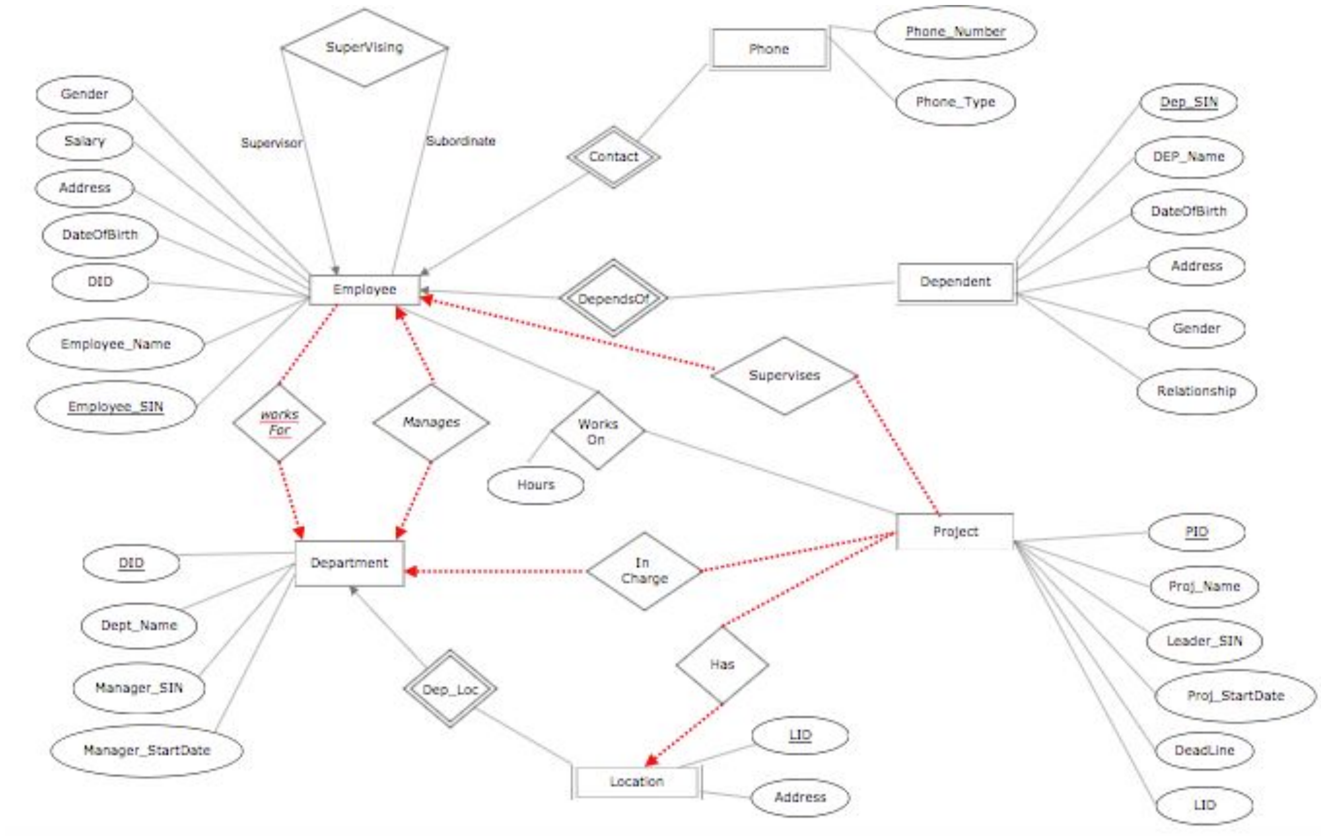
I certify that I have read the requirements set out on this form, and that I am aware of these requirements. I certify that all the work I will submit for this course will comply with these requirements and with additional requirements stated in the course outline.

Course Number: COMP 353 - Winter 2018 Instructor: N. Shiri
Net Names: f_ferna, m_malde, n_dimit, e_raya I.Ds# 40044982, 26562906, 24662407, 40016485
Signatures: Date: February 7th, 2018
[illegible], [illegible], [illegible]

[1] Rules for reference citation can be found in "Form and Style" by Patrich MacDonagh and Jack Bordan, fourth edition, May, 2000, available at http://www.encs.concordia.ca/scs/Forms/Form&Style.pdf.
Approved by the ENCS Faculty Council February 10, 2012

TABLE OF CONTENTS

1. E/R Diagram



*NOTE: dashed lines in red are relationships omitted from our creation schema but that still exist in the database itself. They will be represented using foreign keys (see referential constraints section).

1. Employee - Department (worksFor): many to one - Many employees can work for one department but one employee cannot work for more than one department.

2. Employee - Department (Manages) : one to one relationship- One employee who is a manager can manage one department at a time.

3. Department - Project (InCharge): one to many - One department is in charge of a number of projects. One department that is responsible for a specific project is stored in Location. The projects can be found through their unique LID (stored in Project).

4. Project - Location (Has): many to one - Each Project has a unique location, but different Projects may have the same location.

5. Employee- Project (Supervises): one to many - One employee may supervise many projects. They can be found through their Leader_SIN (stored in Project).

Relationships

- Department - Location (*Dep_Loc* relation): one to many - One department can have several locations, but each location belongs to only one Department.

- Employee - Phone (*Contact* weak relation): one to many - an employee can have multiple phone numbers (cell, home).
- Employee - Project (*WorksOn* relation): many to many - Many employees can work on the same project and one employee can work on multiple projects.
- Employee - Dependents (*DependsOn* weak relation): one to many - One employee may have multiple dependents (i.e two children).
- Employee - Employee (*SuperVising* role relation): one to many - Each Employee (that is not himself a Supervisor, Leader or Manager) has a Supervisor that is also an Employee, and a Supervisor may have several subordinates.

Assumptions

- *Relationship* (attribute in Dependent entity) - it indicates the type of relationship between the employee and the dependent (i.e. mother, father, child, husband, wife, etc.).
- *Deadline* (attribute in Project entity) - holds a date indicating the deadline of the project. If the date has already passed, the project is considered as completed.
- *Proj_StartDate* (attribute in Project entity) - holds the starting date of a project. It is used to calculate the stage of the project, which depends on the time remaining until the deadline.
- *Manager_SIN* (attribute in Department entity) - holds values of the Employee_SIN, indicating which employee manages the department.

  *Note: We assume that the employee who manages that specific department was also assigned to work in that department in general. DID in Employee has to match DID in Department .
- *Manager_StartDate* (attribute in Department entity) - holds values of Date, indicating when the manager started to manage that department.
- *Leader_SIN* (attribute in Project) - holds values of the Employee_SIN, indicating which employee is the leader of the project.
- *Phone* (weak entity set) - holds values for the type of the phone number (home or cell) and the number itself.

  *A separate entity was created in order to normalize the relation. After further analysis, it came to our attention that the relation was in 1NF. This norm can not have multivalued attributes (phone) and was consequently normalized. We assume that no different employees will have the same number.
- *Location* (weak entity set) - this entity holds the locations for departments and projects. Each location has a unique *LID* and an Address. The location LID is added to avoid redundancy and simplify the insertion of the address. We chose this entity to be weak, because it cannot exist on its own. We assume that a company will not buy or rent a location without assigning it to a department at the same time.
- *Dependent* (weak entity set) - We assume this entity set is weak because it cannot exist on its own. A dependent person has nothing to do in the company's database, if he/she is not assigned to be a dependent of an employee.

2. Relational Database Schema (See Appendix A for the detailed Schema Creation script)

Employee (Employee_SIN, Employee_Name, Address, DateOfBirth, Gender, Salary, DID)
Dependent (Employee_SIN, Dep_SIN, DEP_Name, DateOfBirth, Address, Gender, Relationship)

Department (DID, Dept_Name, Manager_SIN, Manager_StartDate)
Project (PID, Proj_Name, Proj_StartDate, DeadLine, LID, Leader_SIN)
WorksOn (Employee_SIN, PID, Hours)
SuperVising (Sup_SIN, Sub_SIN)
Location (LID, DID, Address)
Phone (Employee_SIN, Phone_Number, Phone_Type)
Notes:
1. We don't include Contact relationship, therefore the weak entity Phone will include:
    - All attributes of Phone
    - All attributes of the supporting relationship Contacts (if any)
    - All key attributes of Employee (many to one relationship)
2. We don't include DependsOf relationship , therefore the weak entity Dependent will include
    - All attributes of Dependent
    - All attributes of the supporting relationship DependsOf (if any)
    - All key attributes of Employee (many to one relationship)
3. We don't include Dep_Loc relationship, therefore the weak entity Location will include:
    - All attributes of Location
    - All attributes of the supporting relationship Dep_loc (if any)
    - All key attributes of Department (many to one relationship)

3. Constraints, Primary keys, Foreign Keys, Referential Constraints & FDs

Primary Keys - For an attribute to be a Primary key, it must be able to determine all other non-prime attributes in that same relation

- Employee_SIN in Employee: this attribute ensures that each employee is a different entity in the Employee relation. Hence, knowing that the SIN of each employee is unique, the concern of a redundant name or address (as for a husband and wife living together) will not be of a problem for the identification of the uniqueness of an employee. This attribute is also a Primary key because knowing the SIN of the employee , all other information can be found.
- {Employee_SIN, Dep_SIN} in Dependents:ensures that each dependent is a different entity and it allows dependents to have the same name, gender, address, date of birth and relationship type.
- DID in Department: each department needs to have an ID so they can be differentiated. The department name in this relation is UNIQUE.
- PID in Project: each project has a unique ID to differentiate the projects in the entity set. The project names are also UNIQUE.
- {Employee_SIN, PID} in WorksOn - Both attributes are needed as a key in order to determine the hours worked by an employee on a specific project. A key in a relation must be able to determine all other attributes. If only Employee_SIN is chosen as a key, PID won't be determined, because a single employee can work on multiple projects. Same case applies to PID, where Employee_SIN and hours are not functionally dependent on PID.
- Sub_SIN in SuperVising - Sub_SIN is a Primary key because each subordinate has a single supervisor.
- LID in Location - LID is a Primary key in Location because it determines all other attributes. Since one department can be located on several addresses (locations), the same location

entities will never have two different departments (DID), therefore, on inserting LID, we will always get the same values for address and department.

- Phone_Number in Phone - Phone_Number was picked to be a Primary key because it is the only attribute that can functionally determine all other attributes in the relation. It also has unique values, as stated in the Assumptions: two employees cannot have the same phone number. Employee_SIN does not determine Phone_number nor Phone_type because an employee can have many numbers (i.e home phone, cell phone) .

Foreign Keys and Referential Constraints

Note for Referential Integrity:

1. Using NO ACTION upon delete is applied for particular scenarios where the referenced attribute is originally Not Null in the schema creation and therefore using on delete SET NULL would be illegal. In addition, we do not use Cascade on the cases where we would not be interested on deleting the entire tuple (for example.after deleting an Employee that is the Manager of a Department, we would not want the Department tuple to be deleted).
2. Using on delete SET NULL will only update the referencing attribute to null in the reference table, hence leaving the remaining attributes of the tuple intact.
3. Cascading a delete is used to remove an entire tuple in the reference table.
4. Cascading an update is used to update the attribute in the reference table.

Department

- ➢ Foreign Key: Manager_SIN is a foreign key referencing to Employee_SIN in Employee.
- ➢ Referential Constraint ‘fk_Department_Employee1’: on delete NO ACTION, on update CASCADE. It ensures that the employee assigned to be a manager in that department also exists in the Employee entity set.

Employee

- ➢ Foreign Key: DID is foreign key referencing to DID in Department.
- ➢ Referential Constraint ‘fk_Employee_Department1’: on delete SET NULL, on update CASCADE. It ensures that the DID assigned to employee already exists in the Department entity set.

Location

- ➢ Foreign Key: DID is a foreign key referencing to DID in Department, it is inherited from the strong entity set Department because Location is a weak entity set related to it.
- ➢ Referential Constraint ‘fk_Location_Department_idx’: on delete SET NULL, on update CASCADE. It ensures that when assigning a Department to a newly created Location, the DID already exists in the Department entity set.

Project

- ➢ Foreign Key: LID is a foreign key referencing LID in Location.
- ➢ Referential Constraint ‘fk_Project_Location1’: on delete NO ACTION, on update CASCADE. It ensures that when inserting a new tuple in Project, the value of LID is already present in the Location table.

- ➢ Foreign Key: Leader_SIN is a foreign key referencing to Employee_SIN in Employee.

- ➢ Referential Constraint ‘fk_Project_Employee1’: on delete SET NULL , on update CASCADE. It ensures that when inserting a new tuple in Project, the Leader_SIN value is also present in the Employee table as an Employee_SIN.

WorksOn

- ➢ Foreign Key: Employee_SIN references the Primary Key Employee_SIN in Employee.
- ➢ Referential Constraint ‘fk_WorksOn_Employee’: on delete CASCADE, on update CASCADE. It ensures that on inserting a new tuple, the value of Employee_SIN is also present in the Employee table. An employee has to exist in the database first, in order to work on a Project.

- ➢ Foreign Key: PID references the Primary Key PID in Project.
- ➢ Referential Constraint ‘fk_WorksOn_Project1’: on delete CASCADE, on update CASCADE. It ensures that when inserting a new tuple, the value of PID is also present in Project table. For an employee to work on a project, the project has to exist in the database.

SuperVising

- ➢ Foreign Key: Sup_SIN and Sub_SIN reference and come from the Primary Key Employee_SIN in Employee.
- ➢ Referential Constraints ‘fk_SuperVising_Employee1’, ‘fk_SuperVising_Employee2’: on delete CASCADE, on update CASCADE. Both constraints ensure that the value of Employee_SIN used when inserting a new tuple is also present in the Employee table.

Phone

- ➢ Foreign Key: Employee_SIN is references Employee_SIN in Employee. It is inherited from the strong entity set Employee because Phone is a weak entity set related to it..
- ➢ Referential Constraint ‘fk_Phone_Employee1’’: on delete CASCADE, on update CASCADE. It ensures that when inserting a new tuple in Phone, the Employee_SIN is also present in the Employee table. The employee has to exist in our database in order to assign a phone number to him/her.

Dependent

- ➢ Foreign Key: Employee_SIN references the Primary key Employee_SIN in Employee. It is inherited from the strong entity set Employee, because Dependent is a weak entity set related to it.
- ➢ Referential Constraint ‘fk_Dependent_Employee1’: on delete CASCADE, on update CASCADE. It ensures that when inserting a new tuple in Dependent table, the Employee_SIN also exists in the Employee table.

<u>Other Constraints</u>

- Every employee should be assigned to work in a single department.
- Two or more employees with same phone number are not allowed (even if they are husband and wife, different phone numbers have to be provided).
- Annual salary expressed in canadian dollars and has a minimum of 15000 → DOUBLE CHECK (‘Salary’ >= 15000).

- Gender has to be a CHAR of size 1 (i.e “M”, “F”) → (‘Gender’ = ‘F’ OR ‘Gender’ = ‘M’)
- Phone_Type is fixed to four letter characters (home or cell) → CHAR (4)
- Employee_SIN, Manager_SIN, Leader_SIN, Sup_SIN, Sub_SIN are all fixed to 11 characters (xxx-xxx-xxx) → CHAR(11)
- Phone_Number is fixed to 12 characters → CHAR (12) of the form xxx-xxx-xxxx.

Functional Dependencies

Definitions we use in the explanations:
α - determinant
β - dependent

α → β

Functional dependency - on the same value of α, the returned value of β has to be the same everywhere in the relation.

- Candidate Key - when an attribute has unique values it is a Candidate Key. A Candidate Key can determine all other attributes in the relation.
- Prime Attributes: a prime attribute is the CK. If there are more than one CK, and an attribute is part of anyone of them, then that attribute is considered to be prime.
- Non-Prime Attribute: an attribute that is not part of the candidate keys
- Partial Dependency : a non prime attribute is functionally dependant on part of the Candidate Key. In other words, we can find a non prime attribute using only part of the Candidate Key
- Transitive Dependency - Non prime attribute is able to find another non prime attribute
- 3NF - A table in 3NF does not have Partial Dependencies and Transitive Dependencies

  α → β

  In all FD in a relation, one of these conditions has to be satisfied for a relation to be in 3NF:

  α is a Candidate Key (Super Key) **OR** β is a Prime Attribute

Employee (<u>Employee_SIN</u>, Employee_Name, Address, DateOfBirth, Gender, Salary, DID)
Candidate Keys:Employee_SIN
Prime Attributes: Employee_SIN
Non-prime Attributes: Employee_Name,Address, DateOfBirth, Gender, Salary, DID
FD: Employee_SIN → Employee_Name Address DateOfBirth Gender Salary DID (α is a Candidate Key)
No partial dependency - the CK consist of only one attribute
No transitive dependency - there is no non-prime attribute that determines another non-prime attribute
Employee is in 3NF because in the FD , α is a Candidate Key

Dependent (<u>Employee_SIN</u>, <u>Dep_SIN</u>, Dep_Name, DateOfBirth, Address, Gender, Relationship)
Candidate Keys: {Employee_SIN Dep_SIN} : NOT NULL and UNIQUE
Prime attributes: Employee_SIN, Dep_SIN
Non-prime attributes: Dep_Name, DateOfBirth, Address, Gender, Relationship
FD: Employee_SIN Dep_SIN → Dep_Name DateOfBirth Address Gender Relationship
Dependent is in 3NF because in the FD , α is a Candidate Key

No partial dependency
No transitive dependency

Department (<u>DID</u>, Dept_Name, Manager_SIN, Manager_StartDate)
Candidate Keys: DID, Dept_Name, Manager_SIN : NOT NULL and UNIQUE
Prime Attributes: DID, Dept_Name, Manager_SIN
Non-prime Attributes: Manager_StartDate

FD: DID → Dept_Name Manager_SIN Manager_StartDate
Dept_Name → DID Manager_SIN Manager_StartDate
Manager_SIN → DID Dept_Name Manager_StartDate
Department is in 3NF because in the three FDs , α is a Candidate Key
No partial dependency
No transitive dependency

Project (<u>PID</u>, Proj_Name, Proj_StartDate, DeadLine, LID, Leader_SIN)
Candidate Keys: PID, Proj_Name, LID : UNIQUE and NOT NULL
Prime attributes: PID, Proj_Name, LID
Non Prime Attributes : Proj_StartDate, DeadLine, Leader_SIN
FD: PID →Proj_name Proj_StartDate DeadLine LID Leader_SIN
Proj_Name → PID Proj_StartDate DeadLine LID Leader_SIN
LID → PID Proj_Name Proj_StartDate DeadLine Leader_SIN
Project is in 3NF because in the three FDs , α is a Candidate Key
No transitive dependency
No partial dependency

WorksOn (<u>Employee_SIN</u>, <u>Project_PID</u>, Hours)
Candidate Keys: {Employee_SIN Project_PID} : UNIQUE and NOT NULL
Prime Attributes: Employee_SIN, PID
Non Prime Attributes: Hours
FD: Employee_SIN Project_PID → Hours
WorksOn is in 3NF because in the FD, α is a Candidate Key
No transitive dependency
No partial dependency

Location (<u>LID</u>, DID, Address)
Candidate Keys: LID
Prime Attributes: LID
Non-prime attributes: DID
FD: LID → Address DID (α is a Candidate Key)
Location is in 3NF because in the FD, either α is a Candidate Key
No partial dependency
No transitive dependency

Phone (Employee_SIN, <u>Phone_Number</u>, Phone_Type)
Candidate Keys: Phone_Number

Prime Attributes: Phone_Number
Non-prime attributes: Employee_SIN, Phone_Type
FD: Phone_Number → Employee_SIN Phone_Type

Note: Employee_SIN Phone_Type → Phone_Number IS NOT an FD because an Employee can have multiple phones of the same type (for example 2 cell phones)
Phone is in 3NF because in the FD, α is a Candidate Key.
No partial dependency
No transitive dependency

SuperVising (Sup_SIN, Sub_SIN)
Candidate Keys: Sub_SIN : a subordinate has only one specific supervisor, therefore it determines Sup_SIN
Prime Attributes: Sub_SIN
Non-prime attributes: Sup_SIN
FD: Sub_SIN → Sup_SIN
SuperVising is in 3NF because in the FD, α is a Candidate Key.
No partial dependency
No transitive dependency

4. Conclusion

Functionalities that have been implemented to satisfy the project requirements

There are 4 HTML pages to the Database System, each offering specific functionality to the user. The Department, Projects and Employees pages are used to modify and update the database. Each one of these pages gives the user the option to either Add, Delete or Modify(Update) a tuple. The system uses php to create the mysql insert, update or delete statements, which are then returned and handled by the php sql functions. The modify option allows the user to change any aspect of a tuple without having to re-enter all the information, only the key is needed to identify the targeted tuple. There is also a branching option which allows the user to interact with any sub tables, that is related to the page. In the case of Employee, the user can branch to a WorksOn relation, which defines which project the employee is currently working on. There is also a Supervisor, Dependent and Phone branch option, giving the user the ability to modify this information. The Query page gives the user the option to either list all the contents of a selected table using the “Display Table” function. The user also has the option to perform a custom query. Hereafter all the queries are separated into 3 sections. The first is for employees, where there are 10 query options offered to the user, 4 require input and the other 6 do not. The second section is dedicated to Projects and gives the user the option to select from 7 queries, 3 which require content and 4 which can be selected. The final section is related to departments which gives the user the ability to select 12 queries, 4 which require content and 8 which can be selected. The majority of the queries are designed to return information on which employees are involved in certain aspects of the company, such as in which department they work and on which projects they participate. The queries also focus on the amount of people who work where and are managed/supervised by who.

Additional features (if any) that complement the project description

Due to the inability of MySQL to do subtraction of two dates, there needed to be a php section dedicated to calculating the difference between two given dates. The queries, such as the progress of a project or any of the average age related ones, take advantage of this php script.

Project Evolution

Our team designed the database from scratch starting with the E/R diagram. After numerous updates and review we came to the conclusion that the database has to be in its most efficient and simple form. There was elimination of multiple relationships, to avoid redundancy since a lot of data can be found using foreign keys from other tables. The E/R diagram was probably changed more than 20 times and after each time, the documentation and the insertions were updated as well. After the Project description was updated on such a short notice, we had to make multiple changes again, since it additionally stated that every Project has a Leader employee.

<u>5. Log of work done</u>

Name	**Work done**
Natalia, Eve	Documentation (multiple reviews and updates), html (front end), E/R diagram, database population (Data document).
Max, Javier	Database schema, review and update of the E/R, document and data, SQL queries and

	transactions, PHP functionalities implementation (back end).

APPENDIX A

```
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=&#39;TRADITIONAL,ALLOW_INVALID_DATES&#39;;

-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
-- Schema ezc353_4
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -

CREATE SCHEMA IF NOT EXISTS `ezc353_4` DEFAULT CHARACTER SET utf8 ;
USE `ezc353_4` ;

-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
-- Table `ezc353_4`.`Department`
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
CREATE TABLE IF NOT EXISTS `ezc353_4`.`Department` (
`DID` INT NOT NULL,
`Dept_Name` VARCHAR(45) NOT NULL,
`Manager_SIN` CHAR(11) NOT NULL,
`Manager_StartDate` DATE NULL,
PRIMARY KEY (`DID`),
UNIQUE INDEX `DID_UNIQUE` (`DID` ASC),
UNIQUE INDEX `Dept_Name_UNIQUE` (`Dept_Name` ASC),
UNIQUE INDEX `Manager_SIN_UNIQUE` (`Manager_SIN` ASC),
CONSTRAINT `fk_Deparment_Employee1`
FOREIGN KEY (`Manager_SIN`)
REFERENCES `ezc353_4`.`Employee` (`Employee_SIN`)
ON DELETE NO ACTION
ON UPDATE CASCADE)
ENGINE = InnoDB;

-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
-- Table `ezc353_4`.`Employee`
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
CREATE TABLE IF NOT EXISTS `ezc353_4`.`Employee` (
`Employee_SIN` CHAR(11) NOT NULL,
`Gender` CHAR(1) CHECK (`Gender`=`F` OR `Gender`=`M`),
`Salary` DOUBLE CHECK (`Salary` &gt;= 15000),
`Address` VARCHAR(45) NULL,
`Employee_Name` VARCHAR(45) NULL,
`DateOfBirth` DATE NULL,
`DID` INT NULL,
PRIMARY KEY (`Employee_SIN`),
UNIQUE INDEX `SIN_UNIQUE` (`Employee_SIN` ASC),
INDEX `fk_Employee_Deparment1_idx` (`DID` ASC),
CONSTRAINT `fk_Employee_Deparment1`
FOREIGN KEY (`DID`)
REFERENCES `ezc353_4`.`Department` (`DID`)
ON DELETE SET NULL
ON UPDATE CASCADE)
ENGINE = InnoDB;

-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
-- Table `ezc353_4`.`Location`
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
CREATE TABLE IF NOT EXISTS `ezc353_4`.`Location` (
`LID` INT NOT NULL,
`DID` INT NOT NULL,
`Address` VARCHAR(45) NULL,
PRIMARY KEY (`LID`),
UNIQUE INDEX `LID_UNIQUE` (`LID` ASC),

INDEX `fk_Location_Department_idx` (`DID` ASC),
CONSTRAINT `fk_Location_Department`
FOREIGN KEY (`DID`)
REFERENCES `ezc353_4`.`Department` (`DID`)
ON DELETE NO ACTION
ON UPDATE CASCADE)
ENGINE = InnoDB;

-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
-- Table `ezc353_4`.`Project`
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
CREATE TABLE IF NOT EXISTS `ezc353_4`.`Project` (
`PID` INT NOT NULL,
`Proj_Name` VARCHAR(45) NOT NULL,
`Proj_StartDate` DATE NULL,
`DeadLine` DATE NULL,
`Leader_SIN` CHAR(11) NULL,
`LID` INT NOT NULL,
PRIMARY KEY (`PID`),
UNIQUE INDEX `PID_UNIQUE` (`PID` ASC),
INDEX `fk_Dependent_Employee1_idx` (`Leader_SIN` ASC),
UNIQUE INDEX `Proj_Name_UNIQUE` (`Proj_Name` ASC),
CONSTRAINT `fk_Project_Employee1`
FOREIGN KEY (`Leader_SIN`)
REFERENCES `ezc353_4`.`Employee` (`Employee_SIN`)
ON DELETE SET NULL
ON UPDATE CASCADE,
CONSTRAINT `fk_Project_Location1`
FOREIGN KEY (`LID`)
REFERENCES `ezc353_4`.`Location` (`LID`)

ON DELETE NO ACTION
ON UPDATE CASCADE)
ENGINE = InnoDB;

-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
-- Table `ezc353_4`.`WorksOn`
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
CREATE TABLE IF NOT EXISTS `ezc353_4`.`WorksOn` (
`Hours` INT NULL,
`Employee_SIN` CHAR(11) NOT NULL,
`PID` INT NOT NULL,
PRIMARY KEY (`Employee_SIN`, `PID`),
INDEX `fk_WorksOn_Project1_idx` (`PID` ASC),
CONSTRAINT `fk_WorksOn_Employee`
FOREIGN KEY (`Employee_SIN`)
REFERENCES `ezc353_4`.`Employee` (`Employee_SIN`)
ON DELETE CASCADE
ON UPDATE CASCADE,
CONSTRAINT `fk_WorksOn_Project1`
FOREIGN KEY (`PID`)
REFERENCES `ezc353_4`.`Project` (`PID`)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;

-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
-- Table `ezc353_4`.`SuperVising`
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
CREATE TABLE IF NOT EXISTS `ezc353_4`.`SuperVising` (

`Sup_SIN` CHAR(11) NOT NULL,
`Sub_SIN` CHAR(11) NOT NULL,
PRIMARY KEY (`Sub_SIN`),
INDEX `fk_SuperVising_Employee2_idx` (`Sub_SIN` ASC),
UNIQUE INDEX `Sub_SIN_UNIQUE` (`Sub_SIN` ASC),
CONSTRAINT `fk_SuperVising_Employee1`
FOREIGN KEY (`Sup_SIN`)
REFERENCES `ezc353_4`.`Employee` (`Employee_SIN`)
ON DELETE CASCADE
ON UPDATE CASCADE,
CONSTRAINT `fk_SuperVising_Employee2`
FOREIGN KEY (`Sub_SIN`)
REFERENCES `ezc353_4`.`Employee` (`Employee_SIN`)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;

-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
-- Table `ezc353_4`.`Phone`
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
CREATE TABLE IF NOT EXISTS `ezc353_4`.`Phone` (
`Phone_Number` CHAR(12) NOT NULL,
`Employee_SIN` CHAR(11) NOT NULL,
`Phone_Type` CHAR(4) NULL,
PRIMARY KEY (`Phone_Number`),
CONSTRAINT `fk_Phone_Employee1`
FOREIGN KEY (`Employee_SIN`)
REFERENCES `ezc353_4`.`Employee` (`Employee_SIN`)
ON DELETE CASCADE
ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
-- Table `ezc353_4`.`Dependent`
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
CREATE TABLE IF NOT EXISTS `ezc353_4`.`Dependent` (
`Dep_SIN` CHAR(11) NOT NULL,
`DEP_Name` VARCHAR(45) NULL,
`DateOfBirth` DATE NULL,
`Address` VARCHAR(45) NULL,
`Gender` CHAR(1) CHECK (`Gender`=`F` OR `Gender`=`M`),
`Relationship` VARCHAR(45) NULL,
`Employee_SIN` CHAR(11) NOT NULL,
PRIMARY KEY (`Dep_SIN`, `Employee_SIN`),
INDEX `fk_Dependent_Employee1_idx` (`Employee_SIN` ASC),
CONSTRAINT `fk_Dependent_Employee1`
FOREIGN KEY (`Employee_SIN`)
REFERENCES `ezc353_4`.`Employee` (`Employee_SIN`)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```