Database Systems

CS205

Zina Abohaia, 20194621@ug.buid.ac.ae

Yousef Mamdouh, 20196448@ug.buid.ac.ae

**Car Rental Database project**

# The initial entities and attributes are:

# **Branch:** BranchNo, Location.

# We are assuming that every Branch has a unique BranchNo, and a unique location.

# **Staff:** StaID, FirstName, LastName, Salary, Position

# We are assuming that every staff member has a unique StaID, FirstName, LastName, and a Position in the Branch, as well as a Salary that is decided by the Position.

# **Car:** RentPrice, Make, Model, PlateNo

# We are assuming that Each Car has a RentPrice, which is the price for renting the car on a

# Daily basis, a Make, Model, and a unique PlateNo.

# **CarForRent:** CarNo, Since, Availability

# We assume that every CarForRent has a unique CarNo, with Since they were added noted, as well as their Availability at any given time.

# **Customer:** CusID, FirstName, LastName, Address,PhoneNo

# We assume that every Customer has a unique CusID, PhoneNo, and their Address, FirstName, and LastName is also taken.

# **RentalContract:** RentalNo, DateOut, DateReturned, TotalPrice

# We assume that every Deal has a unique RentalNo, and the DateOut and DateReturned are

# Used to calculate the TotalPrice.

# Next, we identify the relationships, cardinalities, and participation constraints:

# **Employ,** is a one-to-many relationship that connects the Branch to the Staff members working in them. Staff members must belong to exactly one Branch. A Branch might

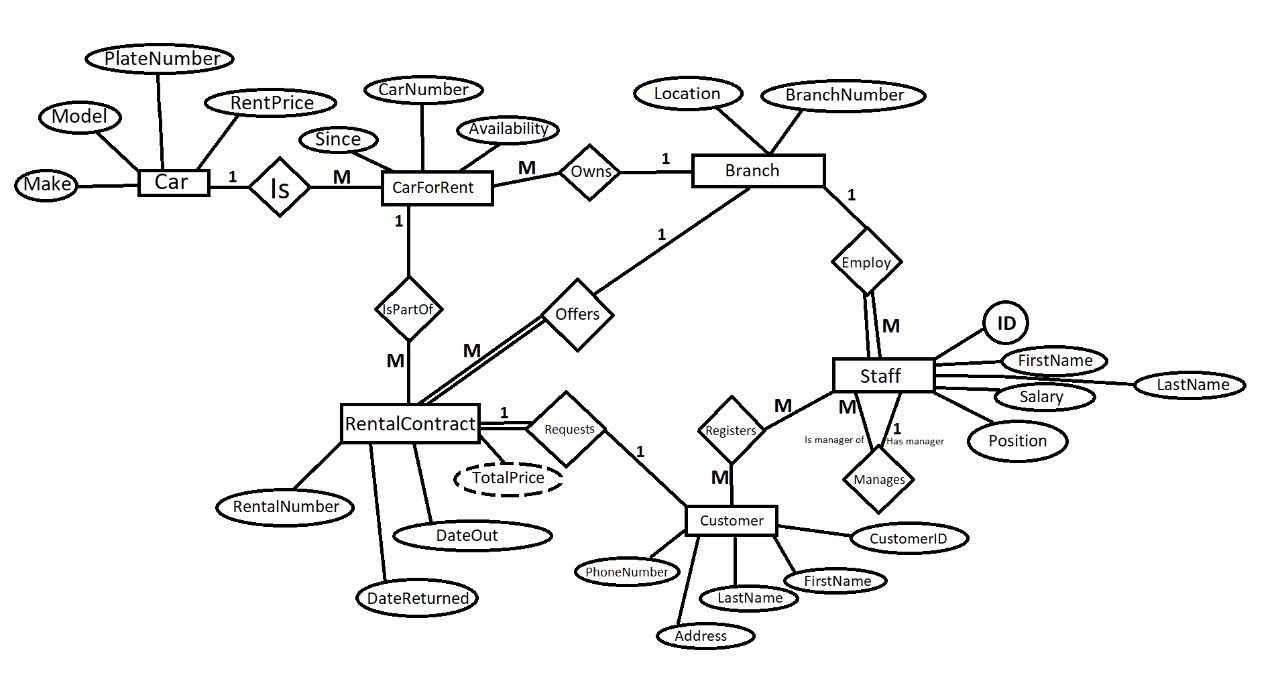
or might not have staff assigned to it, but it can also have many staff members. Because we are

representing Branch as an entity in the diagram, we do not need the branch attribute for

Staff members. The Employ relationship already describes their Branch.

1. **Registers,** which is a many-to-one relationship that connects staff members to the customers they register. We assume a staff member may register zero or many customers, and each customer is registered by one staff member, at a given time.
2. **Requests,** is a one-to-one relationship that connects the customer to the RentalContract they are requesting. At a given time, one customer can request a maximum of one RentalContract.
3. **Manages**, is a many-to-one recursive relationship in Staff. The Staff can have at most one manager, and one manager can manage many staff.
4. **Owns,** is a one-to-many relationship that connects the branch to CarForRent. A branch may not have any cars for rent, especially when it is just starting out. Every CarForRent needs to have a branch.
5. **Is,** is a one-to-many relationship that connects the Car entity to CarForRent. CarForRent must belong to a specific car. A car can have many CarForRent entities at different times. Every Car has at least one CarForRent Entity.
6. **IsPartOf,**  is a one-to-many relationship that connects CarForRent to RentalContract. A CarForRent may or may not have a RentalContract assigned to it but it can also have many RentalContracts with different dates.
7. We should note that the **RentalContract** is related to the **Branch** entity since each RentalContract is offered by the branch. We add the relationship **Offers,** a one-to-many relationship that connects the branch to the RentalContracts offered. A branch may not have any RentalContracts at any given period, but every RentalContract is offered by the branch.

Car Rental E-R Diagram:



## There are eight relationship sets shown in the ER diagram, represented using diamonds. To represent the 1:M Employ relationship set connecting Branch to Staff, we place the primary key of the "one" side, Branch, in the table for the "many" side, Staff. We don't place the primary key of Staff in Branch since then we would have a multi-valued attribute.

## **Owns** is a one-to-many relationhsip connecting Branch to CarForRent. We put the primary key of Branch, BranchNo, in the CarForRent table.

## **Employ** is a one-to-many relationship representing the connection between Branch and Staff. Using the foreign key mechanism, we add BranchNo to the Staff table.

## **Offers** is a one-to-many relationship representing the connection between Branch and RentalContract. Again, we use the foreign key mechanism, we add BranchNo to the RentalContract table.

## **Is** is a one-to-many relationship representing the connection between Car and CarForRent. Using the foreign key mechanism, we add PlateNo, Car's primary key, to the CarForRent table.

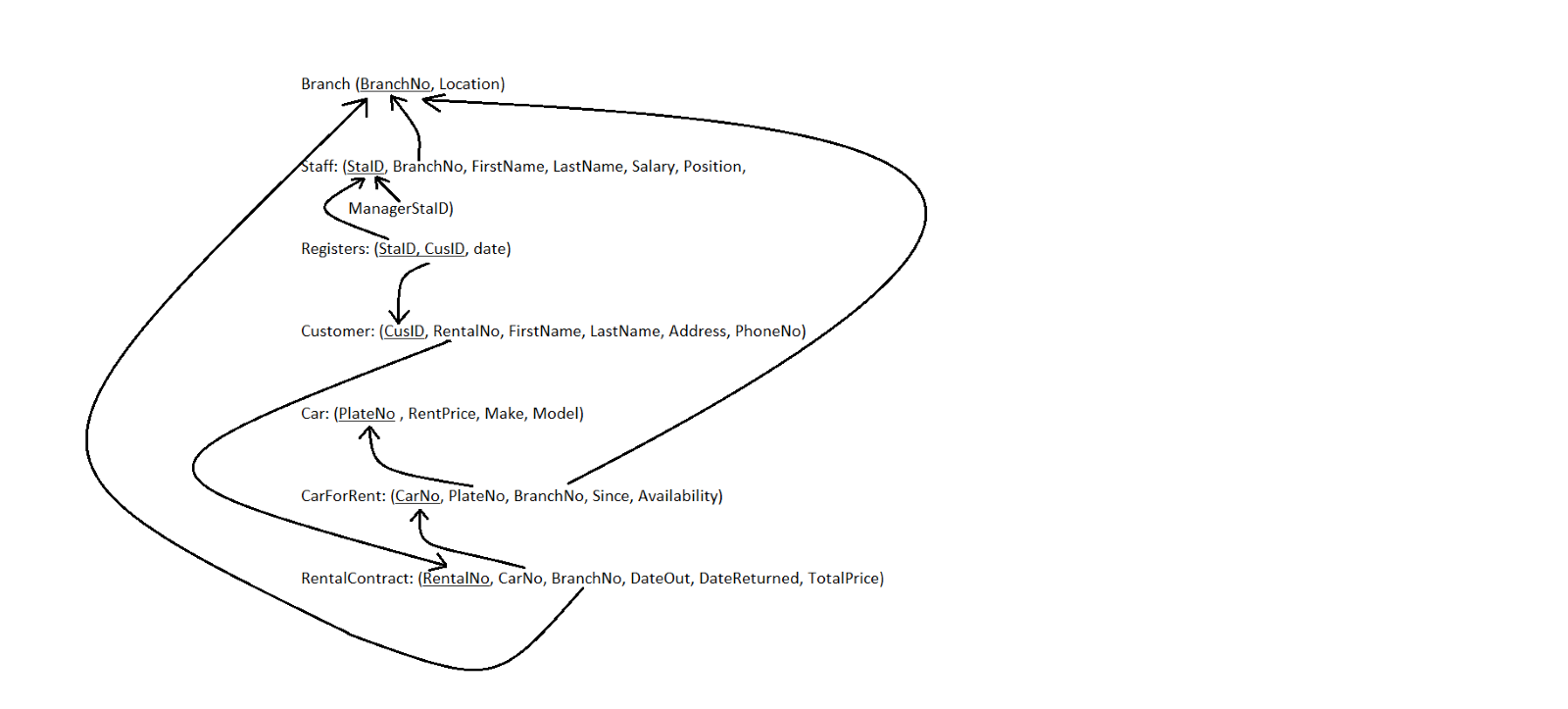
## **IsPartOf** is a one-to-many relationship representing the connection between CarForRent and RentalContract. Using the foreign key mechanism, we add PlateNo, CarForRent's primary key, to the RentalContract table, the "many" side.

## **Manages** is a one-to-many recursive relationship with Staff. Since the cardinality is one-to-many, we can use the foreign key mechanism. We make the key ManagerStaID, to reperesnt this relationship, adding to the Staff table.

## **Requests** is a one-to-one relationship that connects RentalContract to Customer. We chose to put the primary key of RentalContract into Customer as a foreign key, instead of doing the opposite.

## The last relationship in our diagram is the many-to-many relationship **Registers**, which connects Staff to Customer. We represent it the same way we represent a strong entity, and add the primary keys of both Staff, StaID, and Customer, CusID, to it.

## We can see this in the following relational schema:



## After Normalisation

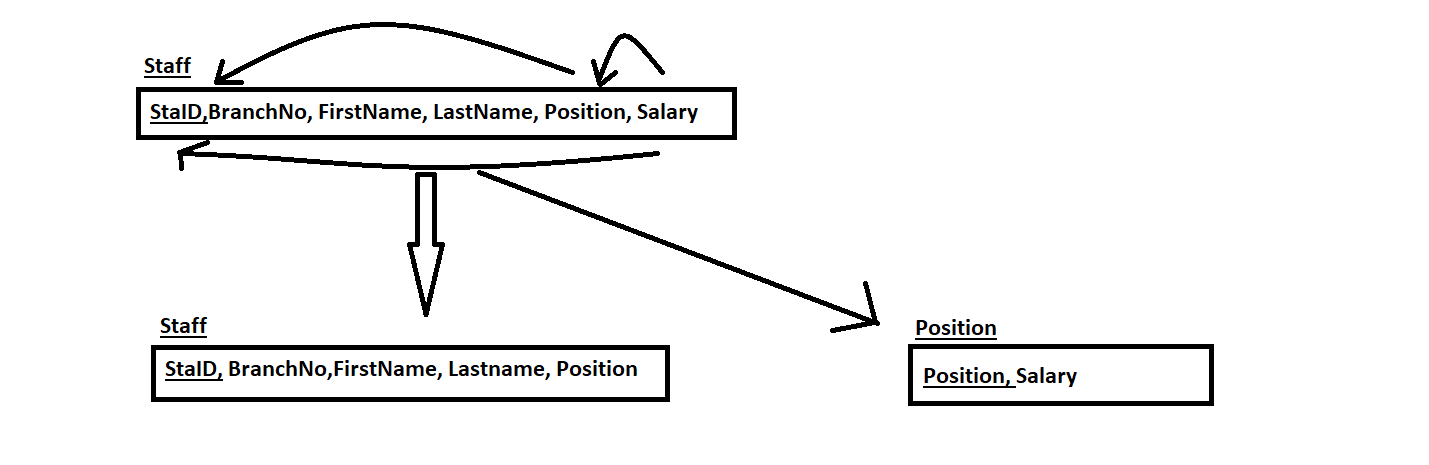
## We have checked every table for the different anomalies, and have found that our tables don’t include any insertion, deletion, nor modification anomalies. All our tables are in BCNF and those that got changed are listed:

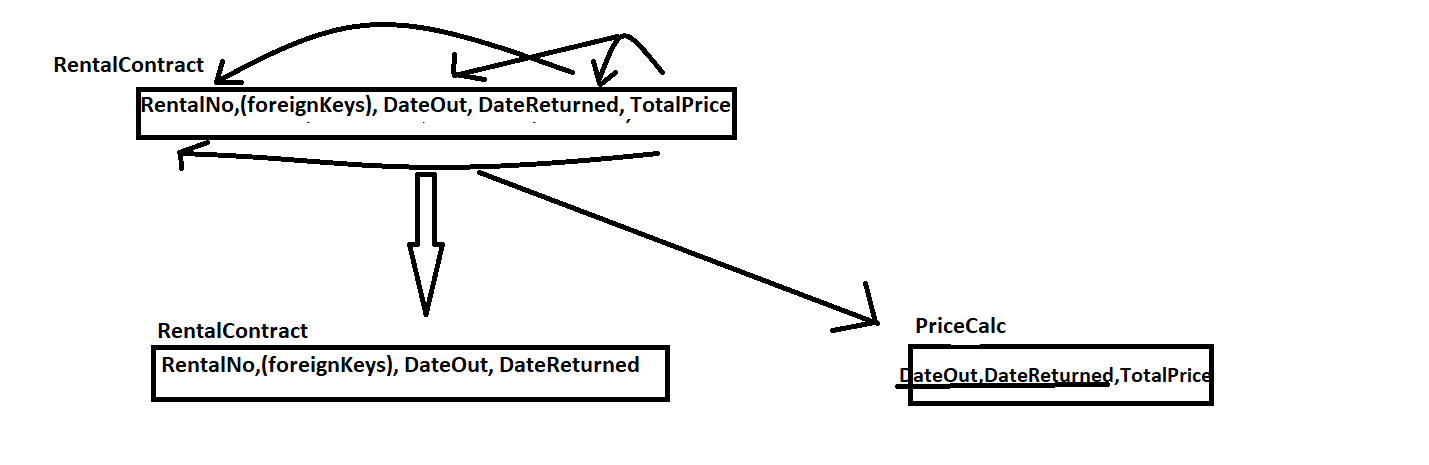
## We have not found any partial dependencies, and found only one table that is not explicitly in 1NF and that was the Customer Table. It included the attributes Address, and PhoneNo, which we only specified to be single-valued. We found Transitive dependencies in Staff (Salary was transitive on position), Car (RentPrice was transitive on the composite of make and model), and in RentalContract, where TotalPrice depended on DateOut, and DateReturned.

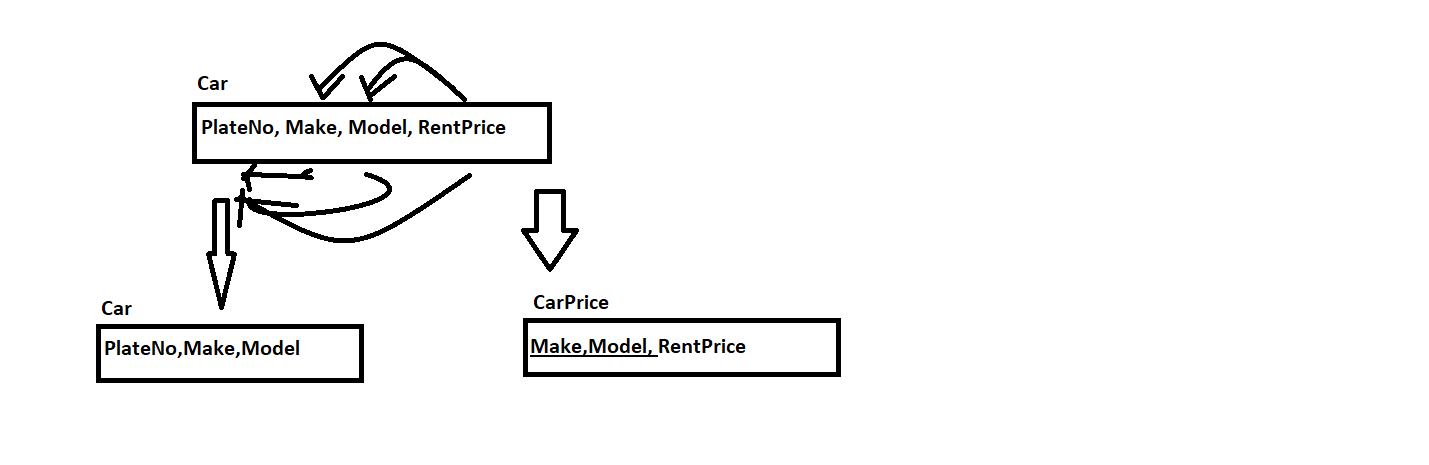
## This is our new schema after normalisation:



The transitive dependencies changes in detail:







SQL Code

CREATE TABLE Branch (

BranchNo VARCHAR2(4),

Location VARCHAR2(20) NOT NULL,

UNIQUE (Location),

CONSTRAINT Branch\_pk PRIMARY KEY (BranchNo));

CREATE TABLE Position (

Position VARCHAR2(20),

Salary INTEGER,

CONSTRAINT Position\_pk PRIMARY KEY (Position));

CREATE TABLE Staff (

StaId VARCHAR2(5),

FirstName VARCHAR2(20) NOT NULL,

LastName VARCHAR2(20) NOT NULL,

BranchNo VARCHAR2(4),

Position VARCHAR2(20),

CONSTRAINT Staff\_pk PRIMARY KEY (StaId),

CONSTRAINT Staff\_BranchNo\_fk FOREIGN KEY (BranchNo) REFERENCES Branch(BranchNo) ON DELETE CASCADE,

CONSTRAINT Staff\_Position\_fk FOREIGN KEY (Position) REFERENCES StaffPosition(Position) ON DELETE CASCADE);

--INSERT RENTALCONTRACT--

CREATE TABLE Customer (

CusId VARCHAR2(6),

FirstName VARCHAR2(20) NOT NULL,

LastName VARCHAR2(20) NOT NULL,

Address VARCHAR2(20) NOT NULL,

PhoneNo VARCHAR2(12) NOT NULL,

RentalNo VARCHAR(),

CONSTRAINT Cus\_pk PRIMARY KEY (CusId),

CONSTRAINT Customer\_RentalNo\_fk FOREIGN KEY (RentalNo) REFERENCES RentalContract(RentalNo) ON DELETE CASCADE);

CREATE TABLE Registers (

StaId VARCHAR2(6),

CusId VARCHAR2(6),

date DATE,

CONSTRAINT Registers\_StaId\_CusId\_pk PRIMARY KEY (StaId, CusId),

CONSTRAINT Registers\_StaId\_fk FOREIGN KEY (StaId) REFERENCES Staff(StaId) ON DELETE CASCADE,

CONSTRAINT Customer\_CusId\_fk FOREIGN KEY (CusId) REFERENCES Customer(CusId) ON DELETE CASCADE);

INSERT INTO Branch

(BranchNo, Location)

VALUES

(1000, Building 3 5th Street),

(1001, Building 7 2nd Street),

(1002, Building 11 7th Street),

(1003, Building 1 9th Street),

(1004, Building 5 6th Street);

INSERT INTO Position

(Position, Salary)

VALUES

(Manager, 15000),

(Accountant, 5000),

(Secretary, 5000),

(CustomerService, 5000),

(Driver, 5000),

(Developer, 5000);

UPDATE Position SET (Salary = 4000) WHERE (Position = Developer)

INSERT INTO Staff

(StaId, FirstName, LastName, BranchNo, Position)

VALUES

(10101, Yousef, Mamdouh, 1000, Developer),

(10202, Zina, Abohaia, 1000, Developer),

(10303, Ana, Lana, 1000, Manager),

(10404, Bana, Sana, 1001, Manager),

(10505, Mathew, Chris, 1000, CustomerService),

(10606, Chris, Mathew, 1001, CustomerService),

(10707, Mike, Christopher, 1002, CustomerService),

(10808, Micheal, Chris, 1003, CustomerService),

(10909, Ahmed, Ahmed, 1004, CustomerService),

(10111, Mohammed, Mohammed, 1005, CustomerService),

(10121, Dan, Pahn, 1002, Accountant),

(10131, Pahn, Dan, 1004, Accountant);

INSERT INTO Customer

(CusId, FirstName, LastName, Address, PhoneNo, RentalNo)

VALUES

(112000, Ghada, Ezzat, Silicon Oasis, 059837592, ),

(112001, Faten, Canaan, Silicon Oasis, 059836893, ),

(110302, Azza, Mahmoud, Jumeirah, 059627594, ),

(110203, Hamza, Hammoud, Mamzar, 059837592, ),

(110504, Fatma, Ali, Al Qusais, 0568345927, );

CREATE VIEW CustomerBasic AS

SELECT CusId, LastName, PhoneNo

FROM Customer

INSERT INTO Registers

(StaId, CusId, date)

VALUES

(10606, 112000, 05-06-2005),

(10808, 112001, 09-08-2005),

(10606, 110302, 11-04-2007),

(10909, 110203, 23-10-2008),

(10111, 110504, 17-01-2010);