**AAA Rentals**

**Final Report**

**Table of Contents**

1. Functional dependency analysis ……………………………………………………1
2. Final Logical data model……………………………………………………………9
3. Final Physical data model………………………………………………………….10
4. Prototype of RDD generated by ERwin……………………………………………11
5. DML statements……………………………………………………………………14
6. Queries and analysis ……………………………………………………………….16
7. APPENDIX…………………………………………………………………………18

**Functional dependency analysis:**

A screenshot of a cell phone

Description generated with very high confidence

A close up of a street

Description generated with high confidence

**FD’s from above analysis:**

{ApartmentID} → {Number}

{ApartmentID} → {NumberRents}

{ApartmentID} → {ComplexID}

{Number} → {NumberRents}

{RenterID} → {NumberRents}

{RenterID} → {Name}

{Name} → {NumberRents}

{Name} → {RenterID}

{StartDate} → {ComplexID}

{PaymentNumber} → {ApartmentID}

{PaymentNumber} → {Number}

{PaymentNumber} → {NumberRents}

{PaymentNumber} → {ComplexID}

{PaymentNumber} → {RenterID}

{PaymentNumber} → {Name}

{PaymentNumber} → {StartDate}

{PaymentNumber} → {Rent}

{PaymentNumber} → {DateDue}

{PaymentNumber} → {DatePaid}

{PaymentNumber} → {Amount}

{DateDue} → {StartDate}

{DateDue} → {ComplexID}

{ProspectNumber} → {ComplexID}

{ProspectNumber} → {Name}

{ProspectNumber} → {Address}

{ProspectNumber} → {Phone}

{ProspectNumber} → {StatusCode}

{ProspectNumber}->{ComplexID,Name,Address,Phone,StatusCode}

{Name} → {ComplexID}

{Name} → {ProspectNumber}

{Name} → {Address}

{Name} → {Phone}

{Name} → {StatusCode}

{Address} → {Phone}

{Phone} -> {Address}

**Functional dependencies from Enterprise statements:**

{ApartmentID, RentarID} -> {Rent}

{RenterID, ApartmentID} ->{StartDate}

{Number, ComplexID} -> {ApartmentID}

{ApartmentID} - > {Number, ComplexID}

{ApartmentID,RenterID,ProspectNumber} -> {Rent, StartDate}

{ApartmentID,ProspectNumber} -> {Rent, StartDate}

{RenterID} -> {ProspectNumber}

**Irreducible cover of FD’s:**

{ApartmentID} -> {Number}

{ApartmentID} -> {ComplexID}

{ProspectNumber} -> {Name}

{ProspectNumber} -> {Phone}

{ProspectNumber} -> {StatusCode}

{Phone} -> {Address}

{Address}->{Phone}

{RenterID} -> {Name}

{RenterID} -> {ProspectNumber}

{RenterID,ApartmentID} -> {Rent}

{RenterID,ApartmentID} -> {StartDate}

{RenterID,ApartmentID} -> {NumberRents}

{PaymentNumber} -> {RenterID}

{PaymentNumber} -> {ApartmentID}

{PaymentNumber} -> {Amount}

{PaymentNumber} -> {DateDue}

{PaymentNumber} -> {DatePaid}

**Applying union and eliminating transitive dependencies, we got below relations:**

{ApartmentID} -> {Number, ComplexID}

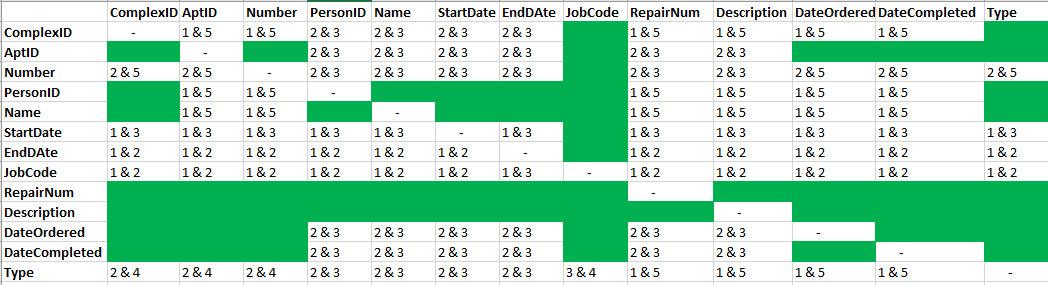
{ProspectNumber} -> {Name,Phone,StatusCode}

{Phone} -> {Address}

{RenterID} -> {Name, ProspectNumber}

{RenterID,ApartmentID} -> {Rent,StartDate, NumberRents}

{PaymentNumber} -> {RenterID,ApartmentID, Amount, DateDue, DatePaid}



{ComplexID} -> {Jobcode}

{ComplexID}->{Type}

{AptID}->{ComplexID}

{AptID}->{Number}

{AptID}->{Jobcode}

{AptID}->{DateOrdered}

{AptID}->{DateCompleted}

{AptID}->{Type}

{Number}->{Jobcode}

{PersonID}->{ComplexID}

{PersonID}->{Name}

{PersonID}->{StartDate}

{PersonID}->{EndDate}

{PersonID}->{Jobcode}

{PersonID}->{Type}

{Name}->{ComplexID}

{Name}->{PersonID}

{Name}->{StartDate}

{Name}->{EndDate}

{Name}->{Jobcode}

{Name}->{Type}

{StartDate}->{Jobcode}

{EndDate}->{Jobcode}

{RepairNum}->{ComplexID}

{RepairNum}->{AptID}

{RepairNum}->{Number}

{RepairNum}->{PersonID}

{RepairNum}->{Name}

{RepairNum}->{StartDate}

{RepairNum}->{EndDate}

{RepairNum}->{Jobcode}

{RepairNum}->{Description}

{RepairNum}->{DateOrdered}

{RepairNum}->{DateCompleted}

{RepairNum}->{Type}

{Description}->{ComplexID}

{Description}->{AptID}

{Description}->{Number}

{Description}->{PersonID}

{Description}->{Name}

{Description}->{StartDate}

{Description}->{EndDate}

{Description}->{Jobcode}

{Description}->{RepairNum}

{Description}->{DateOrdered}

{Description}->{DateCompleted}

{Description}->{Type}

{DateOrdered}->{ComplexID}

{DateOrdered}->{AptID}

{DateOrdered}->{Number}

{DateOrdered}->{DateCompleted}

{DateOrdered}->{Type}

{DateCompleted}->{ComplexID}

{DateCompleted}->{AptID}

{DateCompleted}->{Number}

{DateCompleted}->{DateOrdered}

{DateCompleted}->{Type}

**Functional dependencies from Enterprise statements:**

{PersonID} -> {NextVisit} (In case of CEO)

{PersonID}->{LastAccess} (In case of Manager)

{ComplexID}->{Description} (Description means complex name)

{Jobcode}->{Description} (Description means Job description)

{ComplexID}->{PersonID} (each complex has one manager)

{Type}->{Description} (Description means Insurance type)

**Irreducible cover of FD’s:**

{PersonID}->{ComplexID}

{PersonID}->{Name}

{PersonID}->{StartDate}

{PersonID}->{EndDate}

{PersonID}->{Jobcode}

{PersonID}->{Type}

{RepairNum}->{AptID}

{RepairNum}->{PersonID}

{RepairNum}->{Description}

{RepairNum}->{DateOrdered}

{RepairNum}->{DateCompleted}

{RepairNum}->{Type}

{PersonID} -> {NextVisit}

{PersonID}->{LastAccess}

**Applying union and eliminating transitive dependencies, we got below relations:**

{Jobcode}->{Description}

{PersonID}-> {Name, StartDate, EndDate, Jobcode}

{PersonID}->{NextVisit} (as CEO) (SubType entities)

{PersonID}->{LastAccess} (as Manager) (SubType entities)

{PersonID} (as Repairer) (SubType entities)

{Type}->{Description} (Description means Insurance type)

{RepairNum}-> {Description, DateOrdered, DateCompleted, ApartmentID, PersonID, Type} (PersonID from subtype Repairer)

{ComplexID}-> {Description, PersonID} (PersonID from subtype Manager)

**Final Logical data model:**

A screenshot of a cell phone

Description generated with very high confidence

**Procedure fallowed to get final data design:**

* I modified data model to satisfy the requirements of Bob by including lastAccess and NextVist columns to relations.

Physical Data Model

A screenshot of a computer

Description generated with very high confidence

**Prototype of RDD Generated by ERwin tool:**

use master;

create database AAA60;

use AAA60;

CREATE TABLE Jobcode

(

JobCode char(1) NOT NULL ,

Description char(10) NULL ,

PRIMARY KEY CLUSTERED (JobCode ASC)

)

go

CREATE TABLE staff

(

PersonID char(4) NOT NULL ,

Name char(20) NULL ,

StartDate date NULL ,

EndDate date NULL ,

JobCode char(1) NULL ,

PRIMARY KEY CLUSTERED (PersonID ASC),

FOREIGN KEY (JobCode) REFERENCES Jobcode(JobCode)

)

go

CREATE TABLE Managers

(

PersonID char(4) NOT NULL ,

LastAccess datetime NULL ,

PRIMARY KEY CLUSTERED (PersonID ASC),

FOREIGN KEY (PersonID) REFERENCES staff(PersonID)

)

go

CREATE TABLE Complex

(

ComplexID char(1) NOT NULL ,

Descripton char(10) NULL ,

PersonID char(4) NULL ,

PRIMARY KEY CLUSTERED (ComplexID ASC),

FOREIGN KEY (PersonID) REFERENCES Managers(PersonID)

)

go

CREATE TABLE Apartment

(

ApartmentID integer NOT NULL ,

Number char(4) NULL ,

ComplexID char(1) NULL ,

NumbeRents integer NULL ,

PRIMARY KEY CLUSTERED (ApartmentID ASC),

FOREIGN KEY (ComplexID) REFERENCES Complex(ComplexID)

)

go

CREATE TABLE Prospectors

(

ProspectNumber integer NOT NULL ,

Name char(20) NULL ,

Phone char(12) NULL ,

Address char(20) NULL ,

PRIMARY KEY CLUSTERED (ProspectNumber ASC)

)

go

CREATE TABLE Status

(

StatusCode integer NOT NULL ,

Description char(10) NULL ,

PRIMARY KEY CLUSTERED (StatusCode ASC)

)

go

CREATE TABLE Waitlist

(

ApartmentID integer NOT NULL ,

ProspectNumber integer NOT NULL ,

StatusCode integer NULL ,

PRIMARY KEY CLUSTERED (ApartmentID ASC,ProspectNumber ASC),

FOREIGN KEY (ApartmentID) REFERENCES Apartment(ApartmentID),

FOREIGN KEY (ProspectNumber) REFERENCES Prospectors(ProspectNumber),

FOREIGN KEY (StatusCode) REFERENCES Status(StatusCode)

)

go

CREATE TABLE Type

(

Type char(1) NOT NULL ,

Description char(10) NULL ,

PRIMARY KEY CLUSTERED (Type ASC)

)

go

CREATE TABLE manualLabour

(

PersonID char(4) NOT NULL ,

PRIMARY KEY CLUSTERED (PersonID ASC),

FOREIGN KEY (PersonID) REFERENCES staff(PersonID)

)

go

CREATE TABLE repair

(

RepairNum integer NOT NULL ,

Description char(10) NULL ,

DateOrdered date NULL ,

DateCompleted date NULL ,

ApartmentID integer NULL ,

Type char(1) NULL ,

PersonID char(4) NULL ,

PRIMARY KEY CLUSTERED (RepairNum ASC),

FOREIGN KEY (ApartmentID) REFERENCES Apartment(ApartmentID),

FOREIGN KEY (Type) REFERENCES Type(Type),

FOREIGN KEY (PersonID) REFERENCES manualLabour(PersonID)

)

go

CREATE TABLE Renter

(

RenterID char(5) NOT NULL ,

Name char(20) NULL ,

PRIMARY KEY CLUSTERED (RenterID ASC)

)

go

CREATE TABLE Lease

(

RenterID char(5) NOT NULL ,

ApartmentID integer NOT NULL ,

Rent money NULL ,

StartDate date NULL ,

PRIMARY KEY CLUSTERED (RenterID ASC,ApartmentID ASC),

FOREIGN KEY (RenterID) REFERENCES Renter(RenterID),

FOREIGN KEY (ApartmentID) REFERENCES Apartment(ApartmentID)

)

go

CREATE TABLE Payment

(

PaymentNumber integer NOT NULL ,

Amount money NULL ,

DateDue date NULL ,

DatePaid date NULL ,

RenterID char(5) NULL ,

ApartmentID integer NULL ,

PRIMARY KEY CLUSTERED (PaymentNumber ASC),

FOREIGN KEY (RenterID,ApartmentID) REFERENCES Lease(RenterID,ApartmentID)

)

go

CREATE TABLE CEO

(

PersonID char(4) NOT NULL ,

NextVisit date NULL ,

PRIMARY KEY CLUSTERED (PersonID ASC),

FOREIGN KEY (PersonID) REFERENCES staff(PersonID)

)

go

**DML statements for loading data into database:**

INSERT INTO Jobcode VALUES('C','CEO');

INSERT INTO Jobcode VALUES('M','Manager');

INSERT INTO Jobcode VALUES('R','Repair');

INSERT INTO staff VALUES('p00','Bob Bureaucrat',Null,Null,'C');

INSERT INTO staff VALUES('p01','Sam Supervisor',Null,Null,'M');

INSERT INTO staff VALUES('p02','Fred Foreman',Null,Null,'M');

INSERT INTO staff VALUES('p03','Mary Manager',Null,Null,'M');

INSERT INTO staff VALUES('p04','Alex Johnson','2015-10-01',Null,'R');

INSERT INTO staff VALUES('p05','Gail Steward','2015-10-01','2017-12-21','R');

INSERT INTO staff VALUES('p06','Ben Jackson','2017-12-22',Null,'R');

INSERT INTO staff VALUES('p07','Beth Redding','2017-12-22',Null,'R');

INSERT INTO Managers VALUES('P01','');

INSERT INTO Managers VALUES('P02','');

INSERT INTO Managers VALUES('P03','');

INSERT INTO Complex VALUES('L','Lakeview','P01');

INSERT INTO Complex VALUES('N','Northside','P02');

INSERT INTO Complex VALUES('P','Princeton','P03');

INSERT INTO Apartment VALUES(1,'101G','L',2);

INSERT INTO Apartment VALUES(2,'201','L',1);

INSERT INTO Apartment VALUES(5,'201','N',1);

INSERT INTO Apartment VALUES(7,'209','L',1);

INSERT INTO Apartment VALUES(8,'333','P',1);

INSERT INTO Apartment VALUES(9,'431P','P',2);

INSERT INTO Apartment VALUES(12,'310','L',Null);

INSERT INTO Apartment VALUES(14,'201','P',Null);

INSERT INTO Prospectors VALUES(55,'Jack Black','555-81-3232','73 Maple Ave');

insert into Prospectors values(60, 'Prem Chand','317-360-3645','402 w West St');

insert into Prospectors values(11, 'Kevin White','555-222-1234','234 Main St');

insert into Prospectors values(31, 'Gail Green','555-234-2525','P.O. Box 22');

insert into Prospectors values(45, 'Ed Brown','555-234-8888','12 N 1st St');

insert into Prospectors values(46, 'Ann Black','555-881-3233','73 Maple Ave');

insert into Status values(0, 'Incomplete');

insert into Status values(1, 'References');

insert into Status values(2, 'Waiting');

insert into Status values(3, 'Complete');

insert into Waitlist values(1,55,3);

insert into Waitlist values(1,60,2);

insert into Waitlist values(2,60,2);

insert into Waitlist values(5,11,1);

insert into Waitlist values(5,31,2);

insert into Waitlist values(12,45,1);

insert into Waitlist values(14,46,0);

insert into Type values('I', 'Insured');

insert into Type values('U', 'Uninsured');

insert into manualLabour VALUES('P04');

insert into manualLabour VALUES('P05');

insert into manualLabour VALUES('P07');

insert into manualLabour VALUES('P06');

insert into repair VALUES(23,'Faucet','2017-12-22','2017-12-23',1,'U','P04');

insert into repair VALUES(28,'Window','2017-12-28','2017-12-31',5,'I','P06');

insert into repair VALUES(31,'Carpet','2017-12-28','2017-12-31',5,'I','P05');

insert into repair VALUES(33,'Roof','2018-01-05','2018-02-23',8,'I',NULL);

insert into repair VALUES(35,'Lock','2018-01-10','2018-01-11',2,'U','P04');

insert into Renter values('A021','Jack Black');

insert into Renter values('C222','Fred Jones');

insert into Renter values('A025','Mike Allen');

insert into Renter values('A023','Jane Black');

insert into Renter values('B444','John Dough');

insert into Renter values('B456','Bill Smith');

insert into CEO values('P00',NULL);

insert into Lease values('A021',1,1100,'2016-12-01');

insert into Lease values('C222',1,1200,'2017-11-15');

insert into Lease values('A025',2,1100,'2016-12-01');

insert into Lease values('A023',5,1200,'2017-11-01');

insert into Lease values('A023',7,1250,'2017-11-15');

insert into Lease values('B444',8,700,'2017-12-01');

insert into Lease values('B456',9,900,'2017-12-01');

insert into Payment values(211,1100,'2017-01-01','2016-12-30','A021',1);

insert into Payment values(397,1100,'2017-02-01','2017-01-29','A021',1);

insert into Payment values(402,1200,'2017-12-15','2017-12-30','C222',1);

insert into Payment values(399,1100,'2017-01-01','2017-01-01','A025',2);

insert into Payment values(400,1200,'2017-12-01','2017-12-01','A023',5);

insert into Payment values(401,1200,'2017-12-15','2017-12-15','A023',7);

insert into Payment values(488,700,'2018-01-01','2016-12-30','B444',8);

insert into Payment values(511,500,'2018-01-01','2017-12-30','B456',9);

insert into Payment values(512,400,'2018-01-01','2017-12-31','B456',9);

select \* from Apartment;

select \* from CEO;

select \* from Complex;

select \* from Jobcode;

select \* from Lease;

select \* from Managers;

select \* from manualLabour;

select \* from Payment;

select \* from Prospectors;

select \* from Renter;

select \* from repair;

select \* from staff;

select \* from Status;

select \* from Type;

select \* from Waitlist;

**Queries in Memo:#3 and Memo:#4**

1) List information for rent payments (and when and for what apartment) that are less than the lease specifies.

select Payment.PaymentNumber,Payment.Amount,Payment.DateDue,Payment.DatePaid,Lease.RenterID,Lease.ApartmentID,Lease.Rent from Payment JOIN Lease on Lease.RenterID=Payment.RenterID and Lease.ApartmentID=Payment.ApartmentID where Lease.Rent > Payment.Amount;

2) For each apartment, list the rent payments in chronological order.

select \* from Payment JOIN Lease on Lease.RenterID=Payment.RenterID and Lease.ApartmentID=Payment.ApartmentID order by

Payment.ApartmentID,Payment.DatePaid;

3) List the name and number for prospects who are waiting for an apartment.

select Waitlist.ApartmentID,Prospectors.Name,Prospectors.ProspectNumber,status.Description from Waitlist JOIN Prospectors on Waitlist.ProspectNumber = Prospectors.ProspectNumber JOIN status on Waitlist.StatusCode=status.StatusCode where status.Description='Waiting';

4) List the top two leases based on rent.

select TOP 2 \* from Lease order by Lease.Rent DESC;

5) List Information about leases signed/started on November 1,2017 and those signed/started on December 1,2017

select \* from Lease where Lease.StartDate='2017-11-01' or Lease.StartDate = '2017-12-01';

6) List Information about the late rent payments.

select \* from Payment where Payment.DatePaid > Payment.DateDue;

7) List Apartment Rental Information by complex

select Complex.ComplexID,Complex.Descripton,Apartment.ApartmentID,Lease.RenterID,Lease.StartDate, Lease.Rent from Lease JOIN Apartment on Lease.ApartmentID=Apartment.ApartmentID JOIN Complex on Apartment.ComplexID = Complex.ComplexID order by Apartment.ComplexID;

8) who (Name and ID) manages the Princeton complex?

select staff.Name,staff.PersonID from Complex JOIN staff on Complex.PersonID=staff.PersonID where Complex.Descripton='Princeton';

9) who (Name and ID) has access to repair data for the northside complex?

select staff.Name,staff.PersonID from Complex JOIN staff on Complex.PersonID=staff.PersonID where Complex.Descripton='northside';

10) How many times has each apartment has been leased ?

select Apartment.ApartmentID,Apartment.NumbeRents from Apartment;

11) List all Insured repairs, Include the name of maintenance person if known.

select repair.RepairNum,staff.Name,repair.Description,repair.DateOrdered,repair.DateCompleted,Type.Description from repair JOIN Type on repair.Type =Type.Type LEFT OUTER JOIN staff on repair.PersonID=staff.PersonID where Type.Description='Insured';

**Appendix**

**OLD Logical data model based on Memo #1:**

**A screenshot of a social media post

Description generated with very high confidence**

**OLD Logical data model:**

**A screenshot of a social media post

Description generated with very high confidence**

**Procedure followed to get above design:**

* According to new sample data given by Design Team leader, I done FD analysis on sample data to get functional dependencies.
* I added necessary code tables to make data base more informative.
* By analyzing those functional dependencies, I came through problems with “Null” in {personID,RepairNum,AppartmentID}, where this FD describes about repair done by person in particular apartment.
* Where if we don’t know the personID, we will get Null’s in above relation which leads to violation of foreign key constraints.
* So, I came with better implementation by keeping apartmentID in Repair table, and sub categorizing the RepairNum and PersonID.
* This will solve the problem of unknow repair done at apartment.
* The same problem occurred when dealing with personID->ComplexID, to keep ComplexID in Staff table, we will get some constraint problems like, Bob is owner. he will get all complexID’s. which is violation of 1st normal form. So, I subcategorized PersonID and ComplexID. Which will solve this problem and satisfies the condition of managers and repairers.
* I choose proper and appropriate field sizes to save memory.

**OLD Physical Data model**

A screenshot of a social media post

Description generated with very high confidence