DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

PES1201800046 RUCHIKA SHAHIDHARA

Zoological Park Management System is an example of a real-world application of a database management system. It facilitates the enquiry about the wildlife and details available on the basis of location, slot, etc. All these details including the details of a customer and the ticket details that the customer has bought. A customer after buying a ticket can then go to the Zoological Park which features animal guides containing different animal kinds. An animal kind consists of animals which has various growth details. An Employee also exists which manages the allotment of the tickets and also look after the animals with respect to feed time, its diet, growth, etc.

The following report on the Zoological Management System first consists of requirements and constraints of the database. The entities and relationships along with its attributes are identified. Then an ER Diagram was charted out. After identifying the functional dependencies, the tables were normalized up to the third normal form. A relational schema was then drafted out after on the normalized tables. Next, the tables were created including the check and referential integrity constraints. After inserting data into the tables, the normalized relation tables were tested for the lossless decomposition property using natural join. Next, the triggers were defined to track the deleted log when a user cancels a ticket by using a before delete trigger and to track the growth of a particular animal when its age, height, weight is updated by using an after update trigger. Finally, various complex SQL queries were written with nested select queries, aggregate functions such as sum, concat and joins such as natural join, left outer join, etc.

The report finally concludes with the pros and cons of the above defined database and its queries for the Zoological Management System. Each process was implemented keeping in mind that it was a real-life example that defines a simple procedure to implement a database and each step is defined in the following manner which also includes screenshots of various processes.

Index

Introduction

Requirements: Entity Relationship

Zoological Park Management System facilitates the storage each Animals detail, their kind, Zoo Region, etc. A Customer goes to a Zoo after buying a Ticket from an Employee who is assigned to the same Zoo that the Customer goes to. The Zoo features an Animal Guide which contains various Animal Kinds that consists of Animals that have some Animal Detail. Moreover, an Employee who manages a ticket can look after Animals.

The following Tables from the Relationships can be obtained:

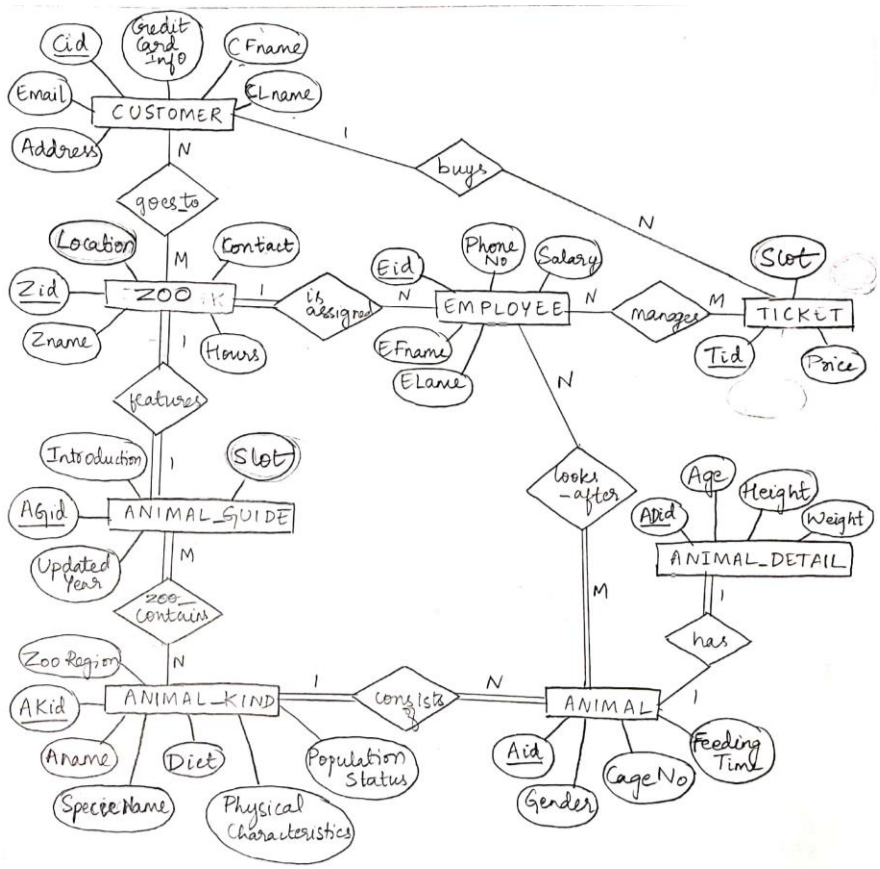
1) Goes_To (Customer, Zoo)
2) Buys (Customer, Ticket)
3) Is Assigned (Zoo, Employee)
4) Features (Zoo, Animal_Guide)
5) Zoo_Contains (Animal_Guide, Animal_Kind)
6) Consists (Animal_Kind, Animal)
7) Has (Animal, Animal_Detail)
8) Manages (Employee, Ticket)
9) Looks_After (Employee, Animal)

The following attributes are bounded by the constraints:

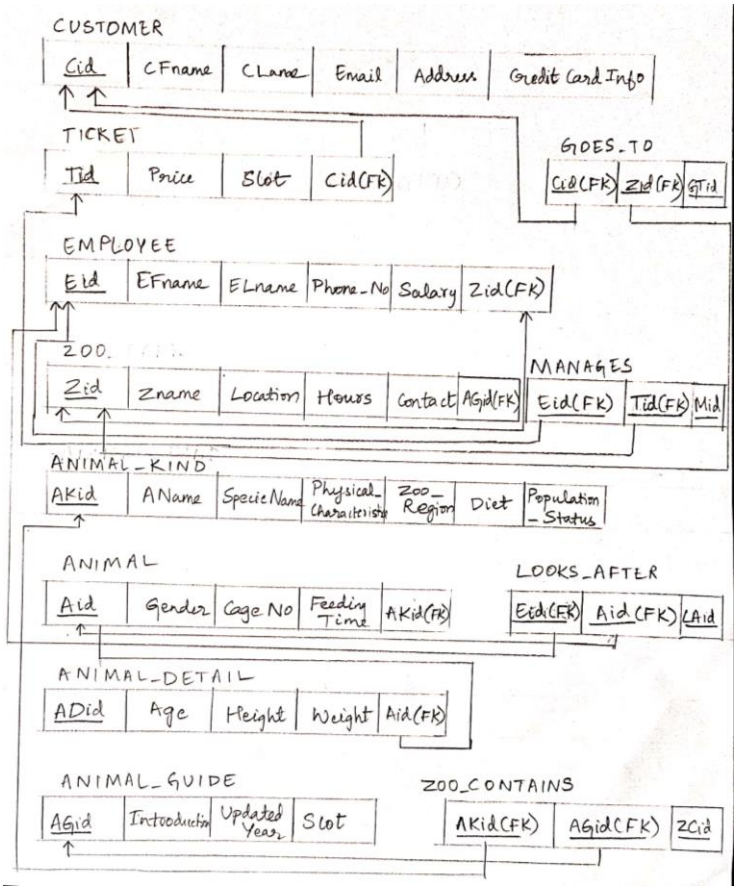
1) Diet of Animal_Kind – Herbivore, Omnivore or Carnivore
2) Price of Ticket – lesser than 2500, greater than 800
3) Salary of Employee – lesser than 27000, greater than 9000
4) Slot of Animal_Guide – Morning, Afternoon, Night

A few of the attributes such as Address stores the city name, Slot stores the time of the day, Introduction stores very minimal details, Hours stores the hours for requesting contact information, etc. for simplicity sake for the current Database created. Suppose if were to expand Address as an example to the complete Address, then it would be again Decomposed for Normalization into a Table containing House Number, Street Name, Locality, City, Pin code, etc. But the normalization has been applied to various N:M relationships by introducing a new Primary Key.

Data Model



ER Diagram



Relational Schema

From the above choice of Tables in the Relational Schema, most of the attributes store the information inside a var char data type, as most of the attributes are strings such as FName, Diet, Email, Species_Name, Population_Status, etc. But for the attributes such as Price, Cage_No, Updated_Year, etc. are stored inside int data type as they are whole number values.

FD and Normalization

The Zoological Park Database consists of the following Functional Dependencies:

1) {ADid} -> {Aid}, {ADid} -> {Age, Height, Weight}
2) {Aid} -> {AKid}, {Aid} -> {Gender, Cage_No, Feeding_Time}
3) {AKid} -> {AGid}, {AKid} -> {Aname, Species_Name, Physical_Characteristics, Zoo_Region, Diet, Population_Status}
4) {Cid} -> {Zid}, {Cid} -> {CFname, CLname, Email, Address, Credit_Card_Info}
5) {Eid} -> {Aid}, {Eid} -> {Tid}, {Eid} -> {Zid}, {Eid} -> {EFname, ELname, Phone_No, Salary}
6) {Tid} -> {Cid}, {Tid} -> {Price, Date}
7) {Zid} -> {AGid} -> {Zid} -> {Zname, Location, Hours, Contact}
8) {Cid, Zid} -> {GTid}
9) {Eid, Tid} -> {Mid}
10) {Eid, Aid} -> {LAid}
11) {AKid, AGid} -> {ZCid}

Information of the Relation Tables after Normalization:

Table_Name (PK – Primary Key, CK – Candidate Key, Normalized Form)

1) Customer (PK: {Cid}, CK: {Cid}, 3NF)
2) Ticket (PK: {Tid}, CK: {Tid}, 3NF)
3) Employee (PK: {Eid}, CK: {Eid}, 3NF)
4) Zoo (PK: {Zid}, CK: {Zid}, 3NF)
5) Animal_Kind (PK: {AKid}, CK: {AKid}, 3NF)
6) Animal (PK: {Aid}, CK: {Aid}, 3NF)
7) Animal_Detail (PK: {ADid}, CK: {ADid}, 3NF)
8) Animal_Guide (PK: {AGid}, CK: {AGid}, 3NF)
9) Goes_To (PK:{GTid}, CK: {Cid, Zid}, BCNF, 4NF)
10) Manages (PK: {Mid}, CK: {Eid, Tid}, BCNF, 4NF)
11) Looks_After (PK: {Laid}, CK: {Eid, Aid}, BCNF, 4NF)
12) Zoo_Contains (PK: {ZCid}, CK: {AKid, AGid}, BCNF, 4NF)

The following Entity Tables store the attributes, there has been no change as they all are atomic values and satisfy the 1st NF and so on:

1) Customer – Cid, CFname, CLname, Email, Address, Credit_Card_Info
2) Animal_Kind – Aname, Species_Name, Physical_Characteristics, Zoo_Region, Diet, Population_Status
3) Animal_Guide – Agid, Introduction, Updated_Year, Slot

After Normalization, the following N:M Relation Tables store the respective Primary Key IDs,

1) Goes_To – GTid, Cid, Zid
2) Zoo_Contains – ZCid, AKid, AGid
3) Manages –Mid, Eid, Tid
4) Looks_After – LAid, Eid, Aid

After Decomposition, the following tables are created to satisfy the decompositions for the relations (N:M), the respective common Prime Attributes as Foreign Keys and their respective attributes are:

1) Ticket – Tid, Price, Date, Cid(FK)
2) Employee – Eid, EFname, ELname, Phone_No, Salary, Zid(FK)
3) Zoo – Zid, Zname, Location, Hours, Contact, AGid(FK)
4) Animal – Aid, Gender, Cage_No, Feeding_Time, AKid(FK)
5) Animal_Detail – ADid, Age, Height, Weight, Aid(FK)

After Decomposition, The following relations satisfy the following relations (1:1 or 1:N) in their respective Tables mentioned below:

1) Buys – Cid(FK) in Ticket
2) Is_assigned – Zid(FK) in Employee
3) Features – Agid(FK) in Zoo
4) Consists – Akid(FK) in Animal
5) Has – Aid(FK) in Animal_Detail

Justification:

- The tables are in 1st Normal Form because there are no multi-valued attributes and multi-valued entities that had separate relations themselves. So, each tuple is a single value from the domain of that attribute.
- The tables are in 2nd Normal Form because all the non-prime attributes are not partially dependent on a proper subset of any of the candidate keys. This has been taken care of for the N:M relationship tables, where a new Primary Key is introduced instead of keeping a Composite Primary Key which may cause update anomaly. Since there are no weak relationships which cause the problem of violation of 2NF, each candidate key is fully functionally dependent on the primary key of that table.
- The tables are in 3rd Normal Form because no prime attribute has transitive functional dependency on a prime attribute. It also satisfies that each non-trivial functional dependency has a candidate key on the LHS of the relation for each the table.
- The 4 tables – Goes_To, Manages, Looks_After, Zoo_Contains are in Boyce-Codd Normal Form because the key on the LHS of the relation is both candidate key and super key. Moreover, the keys on the RHS are prime attributes as well when the key on the LHS is a single sttribute. These 4 tables are also in 4th Normal Form because each table consists of 2 independent multivalued attributes describing the relevant entity(Eg: Cid and Zid of Goes_To)

Hence, the conclusion that the Zoological Park database is Normalized to 3NF but a few of them are in 4NF as well.

DDL

Below is the script to create the Zoological Park Database.
It defines referential integrity constraints for each Foreign Key. For example, Zid constraint of the Employee table has been defined as:
CONSTRAINT fk_emp_zid FOREIGN (Zid) REFERENCES ZOO(Zid)
It also includes check constraints for: (Constraints present in Alter Table)

1) Diet of Animal_Kind – Herbivore, Omnivore or Carnivore
2) Price of Ticket – lesser than 2500, greater than 800
3) Salary of Employee – lesser than 27000, greater than 9000
4) Slot of Animal_Guide – Morning, Afternoon, Night

```
DROP DATABASE IF EXISTS ZooPark;
CREATE DATABASE ZooPark;

DROP TABLE IF EXISTS CUSTOMER;
CREATE TABLE CUSTOMER ( Cid VARCHAR(15) NOT NULL, CFname VARCHAR(30) NOT NULL, CLname
VARCHAR(30) NOT NULL, Email VARCHAR(30) DEFAULT NULL,  Address VARCHAR(100) DEFAULT
NULL, Credit_Card_Info VARCHAR(100) DEFAULT NULL, PRIMARY KEY(Cid) );

DROP TABLE IF EXISTS TICKET;
CREATE TABLE TICKET  ( Tid VARCHAR(15) NOT NULL, Price INT NOT NULL, Slot VARCHAR(15)
NOT NULL, Cid VARCHAR(15) NOT NULL, PRIMARY KEY(Tid), CONSTRAINT fk_ticket_cid FOREIGN
KEY (Cid) REFERENCES CUSTOMER(Cid) ON DELETE CASCADE ON UPDATE CASCADE );

DROP TABLE IF EXISTS ANIMAL_GUIDE;
CREATE TABLE ANIMAL_GUIDE ( AGid VARCHAR(15) NOT NULL, Zoo_Introduction TEXT NOT NULL,
Updated_Year INT DEFAULT NULL, Slot VARCHAR(15) NOT NULL, PRIMARY KEY(AGid) );

DROP TABLE IF EXISTS ZOO;
CREATE TABLE ZOO ( Zid VARCHAR(15) NOT NULL, ZName VARCHAR(50) NOT NULL, Location
VARCHAR(100) NOT NULL, Hours VARCHAR(100) NOT NULL, Contact VARCHAR(100) NOT NULL,
AGid VARCHAR(15) NOT NULL, PRIMARY KEY(Zid), CONSTRAINT fk_zoo_AGid FOREIGN KEY (AGid)
REFERENCES ANIMAL_GUIDE(AGid) ON DELETE CASCADE ON UPDATE CASCADE );

DROP TABLE IF EXISTS EMPLOYEE;
CREATE TABLE EMPLOYEE ( Eid VARCHAR(15) NOT NULL, EFname VARCHAR(30) NOT NULL, ELname
VARCHAR(30) NOT NULL, Phone_No VARCHAR(30) NOT NULL, Salary INT NOT NULL, Zid
VARCHAR(15) NOT NULL, PRIMARY KEY(Eid), CONSTRAINT fk_emp_zid FOREIGN KEY (Zid)
REFERENCES ZOO(Zid)  ON DELETE CASCADE ON UPDATE CASCADE );

DROP TABLE IF EXISTS ANIMAL_KIND;
CREATE TABLE ANIMAL_KIND ( AKid VARCHAR(15) NOT NULL, AName VARCHAR(30) NOT NULL,
Species_Name VARCHAR(30) NOT NULL, Physical_Characteristics TEXT NOT NULL,  Zoo_Region
VARCHAR(50) NOT NULL, Diet VARCHAR(30) NOT NULL,Population_Status VARCHAR(30) NOT
NULL, PRIMARY KEY(AKid) );
DROP TABLE IF EXISTS ANIMAL;
CREATE TABLE ANIMAL ( Aid VARCHAR(15) NOT NULL, Gender VARCHAR(10) DEFAULT NULL,
Cage_Number INT NOT NULL, Feed_Time VARCHAR(30) NOT NULL, AKid VARCHAR(15) NOT NULL,
PRIMARY KEY(Aid), CONSTRAINT fk_ani_akid FOREIGN KEY (AKid) REFERENCES
ANIMAL_KIND(AKid) ON DELETE CASCADE ON UPDATE CASCADE );

DROP TABLE IF EXISTS ANIMAL_DETAIL;
CREATE TABLE ANIMAL_DETAIL ( ADid VARCHAR(15) NOT NULL, Height VARCHAR(10) DEFAULT
NULL, Weight VARCHAR(10) DEFAULT NULL, Age INT DEFAULT NULL, Aid VARCHAR(15) NOT NULL,
PRIMARY KEY(ADid), CONSTRAINT fk_ad_aid FOREIGN KEY (Aid) REFERENCES ANIMAL(Aid) ON
DELETE CASCADE ON UPDATE CASCADE );
```

```
DROP TABLE IF EXISTS GOES_TO;
CREATE TABLE GOES_TO (GTid VARCHAR(15) NOT NULL, Cid VARCHAR(15) NOT NULL, Zid
VARCHAR(15) NOT NULL, PRIMARY KEY(GTid),CONSTRAINT fk1_gt_cid FOREIGN KEY (Cid)
REFERENCES CUSTOMER(Cid) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT fk2_gt_zid
FOREIGN KEY (Zid) REFERENCES ZOO(Zid) ON DELETE CASCADE ON UPDATE CASCADE );

DROP TABLE IF EXISTS MANAGES;
CREATE TABLE MANAGES ( Mid VARCHAR(15) NOT NULL, Eid VARCHAR(15) NOT NULL, Tid
VARCHAR(15) NOT NULL, PRIMARY KEY(Mid),Eid VARCHAR(15) NOT NULL, Tid VARCHAR(15) NOT
NULL, PRIMARY KEY(Eid, Tid), CONSTRAINT fk1_man_eid FOREIGN KEY (Eid) REFERENCES
EMPLOYEE(Eid) ON DELETE CASCADE ON UPDATE CASCADE,     CONSTRAINT fk2_manages_tid
FOREIGN KEY (Tid) REFERENCES TICKET(Tid) ON DELETE CASCADE ON UPDATE CASCADE );
DROP TABLE IF EXISTS LOOKS_AFTER;
CREATE TABLE LOOKS_AFTER (LAid VARCHAR(15) NOT NULL, Eid VARCHAR(15) NOT NULL, Aid
VARCHAR(15) NOT NULL, PRIMARY KEY(LAid),CONSTRAINT fk1_la_eid FOREIGN KEY (Eid)
REFERENCES EMPLOYEE(Eid) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT fk2_looka_aid
FOREIGN KEY (Aid) REFERENCES ANIMAL(Aid) ON DELETE CASCADE ON UPDATE CASCADE );

DROP TABLE IF EXISTS ZOO_CONTAINS;
CREATE TABLE ZOO_CONTAINS (ZCid VARCHAR(15), AKid VARCHAR(15) NOT NULL, AGid
VARCHAR(15) NOT NULL, PRIMARY KEY(ZCid),CONSTRAINT fk1_cont_akid FOREIGN KEY (AKid)
REFERENCES ANIMAL_KIND(AKid) ON DELETE CASCADE ON UPDATE CASCADE,    CONSTRAINT
fk2_contains_agid FOREIGN KEY (AGid)REFERENCES ANIMAL_GUIDE(AGid) ON DELETE CASCADE ON
UPDATE CASCADE );

ALTER TABLE ANIMAL_KIND
ADD CONSTRAINT chk_Animal_diet CHECK (Diet = 'Carnivore' OR Diet = 'Herbivore' OR Diet
= 'Omnivore');

ALTER TABLE TICKET
ADD CONSTRAINT chk_ticket_price CHECK (Price >=800 AND Price <=2500);

ALTER TABLE EMPLOYEE
ADD CONSTRAINT chk_employee_salary CHECK (Salary >=9000 AND Salary <=27000);

ALTER TABLE ANIMAL_GUIDE
ADD CONSTRAINT chk_Animal_slot CHECK (Slot = 'Morning' OR Slot = 'Afternoon' OR Slot =
'Night');
```

Testing for Lossless Join Property

The Lossless Join property is satisfied for relation R decomposed into R1, R2 if the following conditions are satisfied:

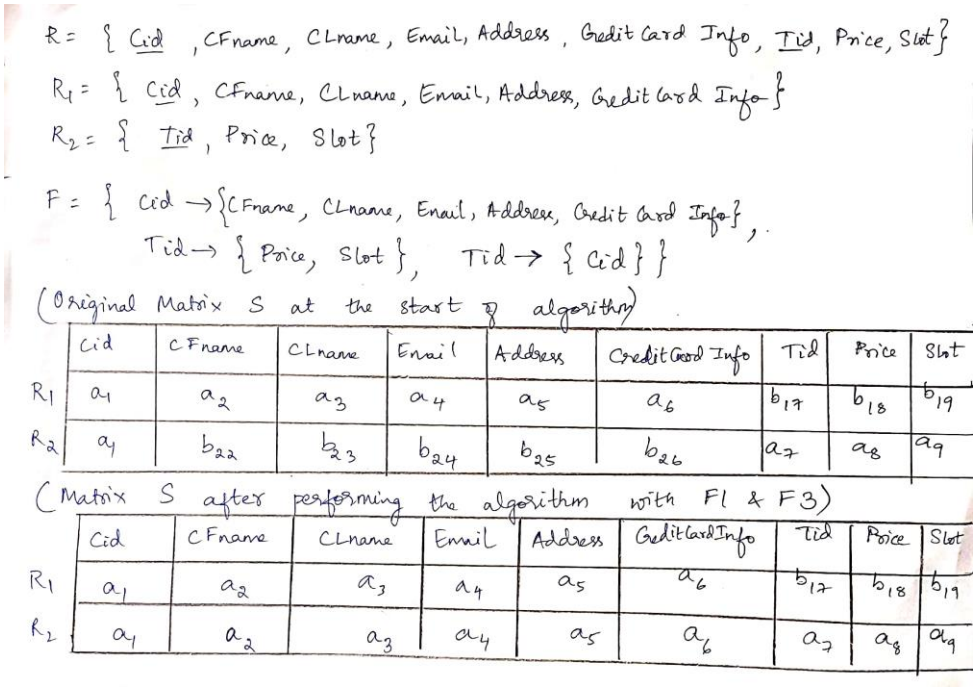
1) Union of Attributes of R1 and R2 must be equal to the attribute of R. Each attribute of R must be either in R1 or R2.
2) Intersection of Attributes of R1 and R2 must be null.
3) There must be one common attribute as the candidate key/primary key for at least one relation R1 or R2.

The two examples shown below, shows that there are no Spurious Tuples from the Lossless Join Decomposition while creating the tables:

Examples:

1) Natural Join between Customer & Ticket Decomposed Relation Tables:

```
SELECT * FROM customer; SELECT * FROM ticket;
SELECT * FROM customer NATURAL JOIN ticket;
```




```
SELECT * FROM customer;
```

Cid	CFname	CLname	Email	Address	Credit_Card_Info
C1	Amal	Sutone	amal@gmail.com	Nagpur	12233445533
C2	Shubham	Maratha	sm@gmail.com	Bangalore	12323456765434
C3	Babu	Rao	br@gmail.com	Mumbai	12345567987
C4	Shamu	Rao	sr@gmail.com	Bangalore	12345348987
C5	Nina	Patel	np@gmail.com	Bangalore	12345678269

```
SELECT * FROM ticket;
```

Tid	Price	Slot	Cid
T10	2500	Morning	C1
T11	2100	Afternoon	C2
T12	2300	Night	C2
T13	1400	Night	C2
T14	1200	Afternoon	C3
T15	1700	AFternoon	C3
T16	1700	Morning	C4
T17	1800	Morning	C4
T18	1900	Morning	C5

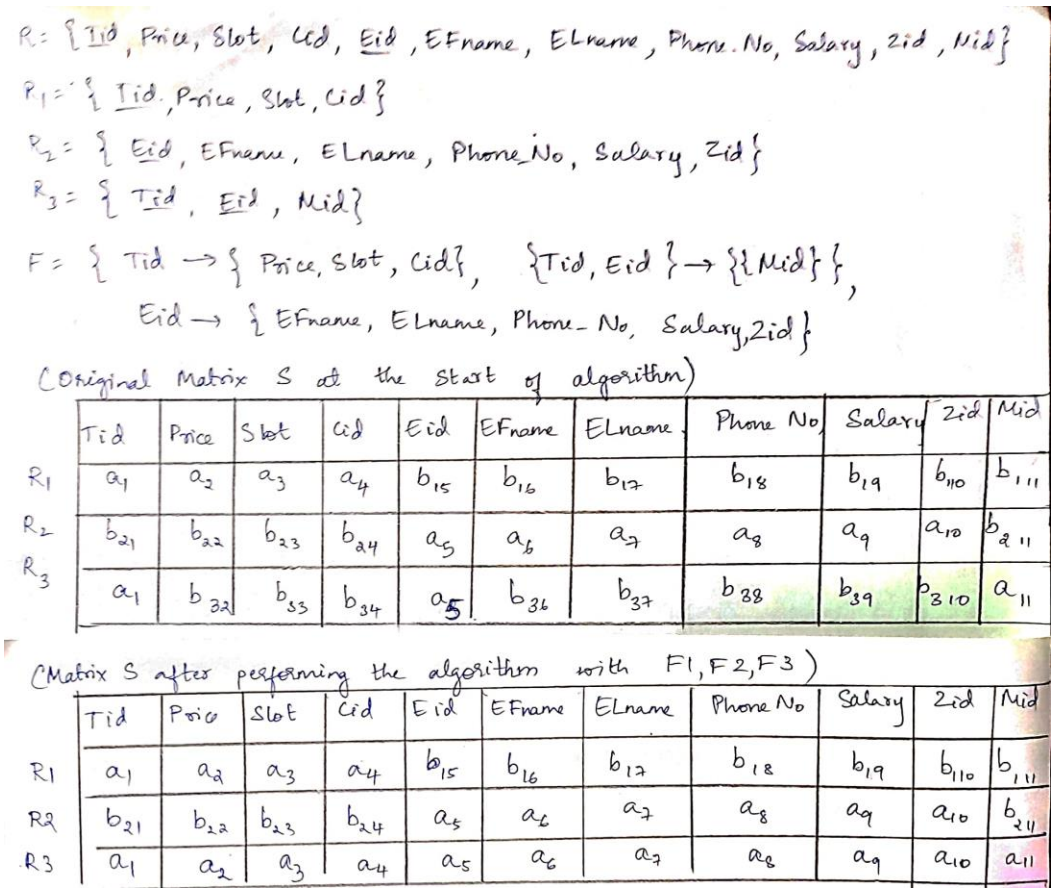
```
SELECT * FROM customer NATURAL JOIN ticket;
```

Cid	CFname	CLname	Email	Address	Credit_Card_Info	Tid	Price	Slot
C1	Amal	Sutone	amal@gmail.com	Nagpur	12233445533	T10	2500	Morning
C2	Shubham	Maratha	sm@gmail.com	Bangalore	12323456765434	T11	2100	Afternoon
C2	Shubham	Maratha	sm@gmail.com	Bangalore	12323456765434	T12	2300	Night
C2	Shubham	Maratha	sm@gmail.com	Bangalore	12323456765434	T13	1400	Night
C3	Babu	Rao	br@gmail.com	Mumbai	12345567987	T14	1200	Afternoon
C3	Babu	Rao	br@gmail.com	Mumbai	12345567987	T15	1700	AFternoon
C4	Shamu	Rao	sr@gmail.com	Bangalore	12345348987	T16	1700	Morning
C4	Shamu	Rao	sr@gmail.com	Bangalore	12345348987	T17	1800	Morning
C5	Nina	Patel	np@gmail.com	Bangalore	12345678269	T18	1900	Morning

There were 5 rows in customer and 8 rows in ticket. After performing a Natural Join on them, 8 rows were seen are there were no spurious tuples. This is not a lossy decomposition of the relation. Hence, the lossless join property of normalization is satisfied.

2) Natural Join between Manages, Employee & Ticket Decomposed Relation Tables:

```
SELECT * FROM manages; SELECT * FROM ticket;SELECT * FROM employee;
SELECT * FROM animal NATURAL JOIN animal_kind;
```



R = { Tid, Price, Slot, Cid, Eid, EFname, ELname, Phone_No, Salary, Zid, Mid }

R_1 = { Tid, Price, Slot, Cid }

R_2 = { Eid, EFname, ELname, Phone_No, Salary, Zid }

R_3 = { Tid, Eid, Mid }

F = { Tid → { Price, Slot, Cid }, {Tid, Eid} → {{Mid}}, Eid → { EFname, ELname, Phone_No, Salary, Zid } }

(Original Matrix S at the start of algorithm)

	Tid	Price	Slot	Cid	Eid	EFname	ELname	Phone No	Salary	Zid	Mid
R_1	a_1	a_2	a_3	a_4	b_{15}	b_{16}	b_{17}	b_{18}	b_{19}	b_{110}	b_{111}
R_2	b_{21}	b_{22}	b_{23}	b_{24}	a_5	a_6	a_7	a_8	a_9	a_{10}	b_{211}
R_3	a_1	b_{32}	b_{33}	b_{34}	a_5	b_{36}	b_{37}	b_{38}	b_{39}	b_{310}	a_{11}

(Matrix S after performing the algorithm with F1, F2, F3)

	Tid	Price	Slot	Cid	Eid	EFname	ELname	Phone No	Salary	Zid	Mid
R1	a_1	a_2	a_3	a_4	b_{15}	b_{16}	b_{17}	b_{18}	b_{19}	b_{110}	b_{111}
R2	b_{21}	b_{22}	b_{23}	b_{24}	a_5	a_6	a_7	a_8	a_9	a_{10}	b_{211}
R3	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}

```
SELECT * FROM manages;
```

Mid	Eid	Tid
M001	E102	T10
M002	E104	T11
M003	E106	T12
M004	E108	T13
M005	E110	T14
M006	E112	T15
M007	E113	T16
M008	E114	T17
M009	E114	T18

```
SELECT * FROM ticket;
```

Tid	Price	Slot	Cid
T10	2500	Morning	C1
T11	2100	Afternoon	C2
T12	2300	Night	C2
T13	1400	Night	C2
T14	1200	Afternoon	C3
T15	1700	AFternoon	C3
T16	1700	Morning	C4
T17	1800	Morning	C4
T18	1900	Morning	C5

```
SELECT * FROM employee;
```

Eid	EFname	ELname	Phone_No	Salary	Zid
E102	Prajoyat	Sharma	9876173344	20000	1001
E104	Adarsh	Patel	9876211243	14000	1002
E106	Akarsh	Singh	9123412344	12000	1003
E108	Rose	Paul	9876785435	13000	1004
E110	Parul	Patel	9167253045	10000	1005
E112	Kumar	Singh	9726385930	10000	1006
E113	Vaishnavi	Vellam	9871232344	10000	1007
E114	Rajesh	Lalit	9712609234	10000	1008

```
SELECT * FROM animal NATURAL JOIN animal_kind;
```

Tid	Price	Slot	Cid	Eid	EFname	EFname	ELname	Phone_No	Salary	Zid	Mid
T10	2500	Morning	C1	E102	Prajoyat	Prajoyat	Sharma	9876173344	20000	1001	M001
T11	2100	Afternoon	C2	E104	Adarsh	Adarsh	Patel	9876211243	14000	1002	M002
T12	2300	Night	C2	E106	Akarsh	Akarsh	Singh	9123412344	12000	1003	M003
T13	1400	Night	C2	E108	Rose	Rose	Paul	9876785435	13000	1004	M004
T14	1200	Afternoon	C3	E110	Parul	Parul	Patel	9167253045	10000	1005	M005
T15	1700	AFternoon	C3	E112	Kumar	Kumar	Singh	9726385930	10000	1006	M006
T16	1700	Morning	C4	E113	Vaishnavi	Vaishnavi	Vellam	9871232344	10000	1007	M007
T17	1800	Morning	C4	E114	Rajesh	Rajesh	Lalit	9712609234	10000	1008	M008
T18	1900	Morning	C5	E114	Rajesh	Rajesh	Lalit	9712609234	10000	1008	M009

There were 9 rows in manages, 9 rows in ticket, 8 rows in employee. After performing a Natural Join on them, 9 rows were seen are there were no spurious tuples. This is not a lossy decomposition of the relation. Hence, the lossless join property of normalization is satisfied.

Triggers

Example 1: Before Delete Trigger

Consider the situation where a customer buys a ticket and then wants to cancel it, then the ticket information in both the tables Ticket and Manages is deleted. But if we want to show the customer the information of the tickets he has cancelled, then an audit log of deleted ticket information needs to be stored in the database. This is information is stored in the Ticket_Archives Table. Hence, by implementing the trigger (before delete) – deleted_Tig_log, the cancelled ticket information from both tables Ticket and Employee id from Manages is stored in Ticket_Archives Table.

```
DROP TABLE IF EXISTS Ticket_Archives;
CREATE TABLE Ticket_Archives(
    TAid INT PRIMARY KEY AUTO_INCREMENT,
    Tid VARCHAR(15), Price INT, Slot VARCHAR(15),
       Cid VARCHAR(15), Eid VARCHAR(15)
);

DROP TRIGGER IF EXISTS deleted_Tid_log
DELIMITER $$
CREATE TRIGGER deleted_Tid_log
    BEFORE DELETE ON ticket FOR EACH ROW
    BEGIN
        INSERT INTO Ticket_Archives(Tid, Price, Slot, Cid, Eid)
        VALUES(OLD.Tid, OLD.Price, OLD.Slot, OLD.Cid,
               (SELECT Eid FROM MANAGES WHERE Tid = OLD.Tid));
    END$$

SELECT * FROM Ticket_Archives;
```

TAid	Tid	Price	Slot	Cid	Eid

```
DELETE FROM TICKET WHERE Tid = 'T13';
DELETE FROM TICKET WHERE Tid = 'T16';

SELECT * FROM Ticket_Archives;
```

TAid	Tid	Price	Slot	Cid	Eid
1	T13	1400	Night	C2	E108
2	T16	1700	Morning	C4	E113

When we perform deletion of Ticket information once with Tid = ‘T15’ and Tid = ‘T16’. We can see that the required information of – Tid, Price, Slot, Cid, Eid has been inserted into Ticket_Archives Table. This is a similar example of log of deleted items from various tables.

Example 2: After Update Trigger

Consider a situation where the care taker of the animal wants to update the age, height and weight of the animal every year. To track the growth of the animal per year we store the cumulative updates of the above-mentioned attributes in Growth_Archives Table whenever the change of the animal’s detail has been updated. To perform the above activity, an updated_Animal_growth_log trigger is implemented. In this trigger, only if the update takes place after a year, the ADid, new Age and difference between the old and new Heights and Weights are inserted into Growth_Archives Table whenever its updated.

```
DROP TABLE IF EXISTS Growth_Archives;
CREATE TABLE Growth_Archives(
    ADGid INT PRIMARY KEY AUTO_INCREMENT,
    ADid VARCHAR(15), Age_new INT, Height_diff VARCHAR(15), Weight_diff VARCHAR(15)
);
```

```
DROP TRIGGER IF EXISTS updated_Animal_growth_log
DELIMITER $$
CREATE TRIGGER updated_Animal_growth_log
    AFTER UPDATE
    ON Animal_Detail FOR EACH ROW
    BEGIN
        IF NEW.Age > OLD.Age THEN
              INSERT INTO Growth_Archives(ADid, Age_new, Height_diff, Weight_diff)
              VALUES(OLD.Adid, NEW.Age,
                     NEW.HEIGHT - OLD.Height, NEW.WEIGHT - OLD.Weight );
        END IF;
    END$$
```

```
SELECT * FROM Animal_Detail WHERE ADid = '60';
```

ADid	Height	Weight	Age	Aid
60	14	215	14	10

```
UPDATE ANIMAL_DETAIL
SET Height = 18, Weight = 220, Age = 15
WHERE ADid = '60';
SELECT * FROM Animal_Detail WHERE ADid = '60';
```

ADid	Height	Weight	Age	Aid
60	18	220	15	10

```
SELECT * FROM growth_archives;
```

ADGid	ADid	Age_new	Height_diff	Weight_diff
1	60	15	4	5

SQL Queries

Example 1: To find information of the Zoo Region from which the ticket with Ticket id = ‘T14’ was generated which was given to by an Employee working in that Zoo Region.

```
SELECT Zname, Location, Hours, Contact
FROM ZOO
WHERE Zid = ANY(SELECT Zid
                FROM EMPLOYEE
                WHERE Eid = ANY (SELECT Eid
                                 FROM MANAGES
                                 WHERE Tid='T14'));
```

Zname	Location	Hours	Contact
RGR	South	10	8787878789

Example 2: To find the full name of the Employee who works for any Zoo Region which is open for Contact with more than 15 hours

```
SELECT CONCAT(EFname, ' ', ELname) as Emp_Name
FROM EMPLOYEE
WHERE Zid = ANY(SELECT Zid
                FROM ZOO
                WHERE Hours > 15);
```

Emp_Name
Adarsh Bhatt
Akarsh Singh
Aswini Parabhat
Rose Paul
Vaishnavi Vellam
Rajesh Lalit

Example 3: To find the total number of tickets and the total cost of the tickets that for each customer with their Full Names

```
SELECT CONCAT(CFname, ' ', CLname) as Customer_Full_Name,
       SUM(Price) as Total_Tickets_Price,
       COUNT(Cid) Total_Tickets
FROM CUSTOMER NATURAL JOIN TICKET
GROUP BY Cid;
```

Customer_Full_Name	Total_Tickets_Price	Total_Tickets
Amal Sutone	2500	1
Shubham Maratha	4400	2
Babu Rao	2900	2
Shamu Rao	1800	1
Nina Patel	1900	1

Example 4: To find the total cost of the tickets that a particular person with Full Name = ‘Babu Rao’ has bought

```
SELECT SUM(Price) as Total_Tickets_Price
FROM CUSTOMER NATURAL JOIN TICKET
WHERE CONCAT(CFname, ' ', CLname) = 'Babu Rao';
```

Total_Tickets_Price
2900

Example 5: To find the details of each animal, its kind, its feed time and cage number that an employee with Full Name = ‘Phalak Jain’ looks after

```
SELECT animal_kind.AName, animal.Aid, animal.Feed_Time, animal.Cage_Number
FROM animal_kind
       LEFT OUTER JOIN animal ON animal_kind.AKid = animal.AKid
       LEFT OUTER JOIN looks_after ON ANIMAL.Aid = looks_after.Aid
       LEFT OUTER JOIN employee ON employee.Eid = looks_after.Eid
WHERE CONCAT(employee.EFname, ' ', employee.ELname) = 'Phalak Jain'
ORDER BY animal_kind.AName
```

AName ▲ 1	Aid	Feed_Time	Cage_Number
Elephant	1	1:00-1:30pm	1
Elephant	2	1:00-1:30pm	1
Fox	5	1:00-1:30pm	3
Fox	6	1:00-1:30pm	3
Peacock	3	2:00-2:30pm	2
Peacock	4	2:00-2:30pm	2
Tiger	7	2:00-2:30pm	4
Tiger	8	2:00-2:30pm	4

Example 6: To find the details of what each Animal Guide of the zoo contains which Animals, their Population status and type of guide provided in the introduction

```
SELECT ANIMAL_GUIDE.AGid, ANIMAL_GUIDE.Slot, ANIMAL_KIND.AName,
ANIMAL_KIND.Population_Status, ANIMAL_GUIDE.Zoo_Introduction
FROM ANIMAL_KIND
       LEFT OUTER JOIN ZOO_CONTAINS ON ANIMAL_KIND.AKid = ZOO_CONTAINS.AKid
       LEFT OUTER JOIN ANIMAL_GUIDE ON ANIMAL_GUIDE.AGid = ZOO_CONTAINS.AGid
ORDER BY ANIMAL_GUIDE.AGid;
```

(Table on next page)

AGid ▲ 1	Slot	AName	Population_Status	Zoo_Introduction
AG1	Morning	Elephant	Endangered	Educational
AG1	Morning	Peacock	Endangered	Educational
AG1	Morning	Fox	Not Endangered	Educational
AG1	Morning	Tiger	Endangered	Educational
AG1	Morning	Zebra	Not Endangered	Educational
AG1	Morning	Giraffe	Not Endangered	Educational
AG2	Afternoon	Elephant	Endangered	Educational
AG2	Afternoon	Peacock	Endangered	Educational
AG2	Afternoon	Fox	Not Endangered	Educational
AG2	Afternoon	Tiger	Endangered	Educational
AG2	Afternoon	Zebra	Not Endangered	Educational
AG2	Afternoon	Giraffe	Not Endangered	Educational
AG3	Night	Elephant	Endangered	Educational
AG3	Night	Peacock	Endangered	Educational
AG3	Night	Fox	Not Endangered	Educational
AG3	Night	Tiger	Endangered	Educational
AG4	Morning	Zebra	Not Endangered	Inspection
AG4	Morning	Giraffe	Not Endangered	Inspection
AG5	Afternoon	Elephant	Endangered	Inspection
AG5	Afternoon	Peacock	Endangered	Inspection
AG5	Afternoon	Fox	Not Endangered	Inspection
AG5	Afternoon	Tiger	Endangered	Inspection
AG6	Night	Zebra	Not Endangered	Inspection
AG6	Night	Giraffe	Not Endangered	Inspection
AG7	Morning	Fox	Not Endangered	Travel
AG7	Morning	Tiger	Endangered	Travel
AG8	Night	Zebra	Not Endangered	Travel
AG8	Night	Giraffe	Not Endangered	Travel

Conclusion

The Zoological Park System was implemented keeping in mind that it was a real-life example that defines a simple procedure to implement a database and each step is defined in the following manner which also includes screenshots of various processes. Each process was meticulously performed from mapping out an ER diagram, creating the model, defining check and integrity constraints along with triggers. This database consists of examples with relationships such as 1:1, 1:N and N:M, every statement defined/implemented takes care of the relationship constraints as well. The normalization of the database is up to the third normal form, but some attributes could have been further normalized (E.g.: Address), but for the sake of simplicity this attribute consists of the area from which the customer is from as there is no need to store the complete information of the customer's address. The other modifications to N:M relations was created with the fourth normal by creating a new primary key value. From the triggers logs also was stored in two new tables which logs the details of deleted ticket information when cancelled and growth parameters of an animal's details were updated. The complex SQL queries also provide accurate details of a converted English sentence as a query. Overall, the database was methodologically created with considering all parameters. This database can further be modified with more constraints and relationships such as more Zoo Parks in different locations, more animals, status of booked tickets, etc.