CENG352 WA-1

Yusuf Eren Tunç

May 6, 2021

1 XML and JSON

For validation check, there will be straight answers final page.

1.1 XML

1.1.1 Part A

$< X >$
$< A >$
$< A > one < /A >$
$< B >$
$< B > two < /B >$
$< B > three < /B >$
$< /B >$
$< C > four < /C >$
$< /A >$

$< A >$
$< B >$
$< A > five < /A >$
$< A > six < /A >$
$< /B >$
$< C > seven < /C >$
$< /A >$
$< /X >$

1.1.2 Part B

i-four, seven
ii- one, four, seven
iii- seven
iv- two, three
v. two, three, five, six
vi. two, three

1.2 JSON


```
{
"suppliers" : [{
                "sid" : "101",
                "sname" : "Acme",
                "address" : "123Main",
                "parts" : [{
                        "pid" : "92",
                        "pname" : "handle",
                        "color" : "Green",
                        "price" : "5.21"
                }
        ]
},
{
        "sid" : "102",
        "sname" : "Ace",
        "address" : "456Lake",
        "parts" : [{
                "pid" : "92",
                "pname" : "handle",
                "color" : "Green",
                "price" : "6.5"
        },
        {
                "pid" : "93",
                "pname" : "gasket",
                "color" : "Red",
                "price" : "65.99"
        }
]
},
{
                "sid" : "103",
                "sname" : "Figaro",
                "address" : "678First"
},
{
        "parts" : [{
                "pid" : "90",
                "pname" : "bumper",
                "color" : "Red"
        },
        {
                "pid" : "91",
```

"*pname*" : "*caliper*",
"*color*" : "*Blue*"
}
]
}
]
}

2 Database Design

2.1 BCNF Decomposition

2.1.1 Part A

If we expand FDs

{PaperNo}$^+$ = {FirstAuthorNo, PaperTitle, PaperAbstract, PaperStatus }
{AuthorNo}$^+$ = {AuthorName, AuthorAddress, AuthorEmail }
{ReviewNo}$^+$ = {ReviewerName, ReviewerEmail, ReviewerAddress }
{PaperNo,ReviewNo}$^+$ = {Comments, ProgramComm, ReviewDate, Rating }

There will be 4 type of relation which are
1- R(PaperNo,FirstAuthorNo,PaperTitle,PaperAbstract,PaperStatus)
2- R(AuthorNo,AuthorName,AuthorAddress,AuthorEmail)
3- R(PaperNo,AuthorNo,Commments,ProgramComm,Rating,ReviewDate,ReviewerAddress,ReviewerEmai
4- R(ReviewerNo,ReviewerName,ReviewerEmail)

To simplfy we will compress third and forth relations into one table. We will name this relations as tables with first one as Paper, second one as Author and compressed one as Review.
Tables will be
Paper(PaperNo, FirstAuthorNo, PaperTitle, PaperAbstract, PaperStatus)
Where PaperNo is key and I think FirstAuthorNo is foreign key to Author table.

Author(AuthorNo, AuthorName, AuthorAddress, AuthorEmail)

Review(ReviewerName, ReviewerEmail, ReviewerAddress, PaperNo,ReviewNo, Comments, ProgramComm, ReviewDate, Rating}

2.1.2 Part B

Since all functional dependencies hold and join of three table is equal to initial schema, process is lossless.

2.2 3NF Decomposition

2.2.1 Part A

Minimal cover is
AC → H
D → E
G → B
E → F
E → K
AD → C
H → G

2.2.2 Part B

There will be 6 relation or 6 table which are

1- E,F,K
2- A,D,C
3- H,G
4- A,C,H
5- D,E
6- G,B

2.3 Finding Dependencies

2.3.1 Part A

FD's are
A → B
C → D
F → G

If queries returns nothing, it means functional dependency of distinctly counted attribute to groupped attribute is hold.

SELECT A
FROM temp
GROUP BY A
HAVING COUNT(DISTINCT B) ¿ 1;

SELECT C
FROM temp
GROUP BY C
HAVING COUNT(DISTINCT D) ¿ 1;

```
SELECT F
FROM temp
GROUP BY F
HAVING COUNT(DISTINCT G) ¿ 1;
```

2.3.2 Part B

```
CREATE TABLE a_b(
        A varchar PRIMARY KEY ,
        B varchar
);

CREATE TABLE c_d(
        C INT PRIMARY KEY ,
        D TEXT
);

CREATE TABLE f_g(
        F VARCHAR PRIMARY KEY ,
        G TEXT
);

CREATE TABLE a_c_e_f(
        A VARCHAR,
        C INT,
        E VARCHAR,
        F TEXT,
        FOREIGN KEY(A)
                REFERENCES a_b(A),
        FOREIGN KEY(C)
                REFERENCES c_d(C),
        FOREIGN KEY(F)
                REFERENCES f_g(F)
);
```

2.3.3 Part C

```
INSERT INTO a_b(A,B)
SELECT DISTINCT A,B
FROM temp;

INSERT INTO c_d(C,D)
SELECT DISTINCT C,D
FROM temp;

INSERT INTO f_g(F,G)
SELECT DISTINCT F,G
FROM temp;

INSERT INTO a_c_e_f(A,C,E,F)
SELECT DISTINCT A,C,E,F
FROM temp;
```

3 SQL DDL

3.1 CREATE TABLE Part

```
CREATE TABLE Customer(
        CustNo VARCHAR(55) PRIMARY KEY,
        CustFirstName VARCHAR(55) ,
        CustLastName VARCHAR(55),
        CustCity VARCHAR(55),
        CustState VARCHAR(55),
        CustZip VARCHAR(55),
        CustBal VARCHAR(55)
);

CREATE TABLE Employee(
        EmpNo VARCHAR(55) PRIMARY KEY,
        EmpFirstName VARCHAR(55),
        EmpLastName VARCHAR(55),
        EmpPhone VARCHAR(55),
        EmpEmail VARCHAR(55),
        EmpDeptName VARCHAR(55),
        EmpStatus VARCHAR(55),
        EmpSalary INT,
        supervisor VARCHAR(55) DEFAULT '007',
        FOREIGN KEY(supervisor) REFERENCES Employee(EmpNo) ON DELETE
SET DEFAULT
);

CREATE TABLE Product(
        ProdNo VARCHAR(55) PRIMARY KEY,
        ProdName VARCHAR(55),
        ProdPrice INT,
        ProdShipDate DATE
);
```

```
    CREATE TABLE Orders(
        OrdNo VARCHAR(55) PRIMARY KEY,
        CustNo VARCHAR(55) NOT NULL,
        EmpNo VARCHAR(55),
        OrdDate VARCHAR(55),
        OrdName VARCHAR(55),
        OrdCity VARCHAR(55),
        OrdZip VARCHAR(55),
        FOREIGN KEY(CustNo) REFERENCES Customer(CustNo) ON DELETE
CASCADE,
        FOREIGN KEY(EmpNo) REFERENCES Employee(EmpNo) ON DELETE
SET NULL
);

CREATE TABLE Contains(
        OrdNo VARCHAR(55) ,
        ProdNo VARCHAR(55) ,
        Qty VARCHAR(55) ,
        FOREIGN KEY(OrdNo) REFERENCES Order(OrdNo) ON DELETE
CASCADE,
        FOREIGN KEY(ProdNo) REFERENCES Product(ProdNo) ON DELETE
CASCADE
);
```

3.2 SQL Check Part

```
CREATE FUNCTION countproduct(ordn varchar(55))
        RETURNS BOOLEAN
        LANGUAGE plpgsql
AS $$
BEGIN
IF (2 < ANY(SELECT count(ProdNo)
                        FROM Contains co
                        WHERE co.ordno = ordn
                        GROUP BY ProdNo))
                THEN RETURN True;
        ELSE RETURN False;
        END IF;
END;
$$;

ALTER TABLE Orders
ADD CONSTRAINT constraint_1
CHECK ( countproduct(ordno) = True );

ALTER TABLE Orders
ADD CONSTRAINT constraint_2
CHECK ( OrdName LIKE CONCAT('%',OrdCity,'%') );

ALTER TABLE Employee
ADD CONSTRAINT constraint_3
CHECK ( EmpEmail NOT LIKE CONCAT('%',EmpFirstName,'%') AND Em-
pEmail NOT LIKE CONCAT('%',EmpLastName,'%') );
```

3.3 ASSERTION PART

```
CREATE ASSERTION assertion_1
CHECK (NOT EXISTS(
SELECT co.ProdNo
FROM Product pro, Contains co
WHERE pro.ProdNo = co.ProdNo
GROUP BY co.ProdNo
HAVING COUNT(qty) ¡= 30
));
```

3.4 TRIGGER PART

```
CREATE FUNCTION trigfunction()
        RETURNS TRIGGER
        LANGUAGE plpgsql
AS $$
BEGIN
        IF (OLD.EmpSalary + (OLD.EmpSalary*15/100) ¡= NEW.EmpSalary)
        THEN UPDATE Employee
        SET EmpStatus = 'Successful'
        WHERE NEW.EmpNo = EmpNO;
        END IF;
        RETURN NULL;
END;
$$;

CREATE TRIGGER trigger_1
        AFTER INSERT ON Employee
        FOR EACH ROW
                EXECUTE PROCEDURE trigfunction();
```