Ceng352 Written Assignment 1

Tolunay Durmuş

2237303

1. Xml Json

   1.1. XML

   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>
```

   b.

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

   1.2. JSON

This representation does not avoid redundancy because we have to store Tuple of product with 92 id twice. So for every Supplier we repeat Parts.
Also, for my representation cost data is stored with pid. So pid is repeated also.

```
{
        "Suppliers": [{
                        "sid": "101",
                        "sname": "Acme",
                        "adress": "123 Main",
                        "Parts": [{
                                "pid": "92",
                                "pname": "handle",
                                "color": "Green"
                        }],
                        "Catalog": [{
                                "pid": "92",
                                "cost": "5.21"
                        }]
                },
                {
                        "sid": "102",
                        "sname": "Ace",
                        "adress": "456 Lake",
                        "Parts": [{
                                        "pid": "92",
                                        "pname": "handle",
                                        "color": "Green"
                                },
                                {
                                        "pid": "93",
                                        "pname": "gasket",
                                        "color": "Red"
                                }
                        ],
                        "Catalog": [{
                                        "pid": "92",
                                        "cost": "6.5"
                                },
                                {
                                        "pid": "93",
                                        "cost": "65.99"
                                }
                        ]
                },
                {
                        "sid": "103",
                        "sname": "Figaro",
                        "adress": "678 First",
                        "Parts": [],
                        "Catalog": []
                },
                {
                        "sid": null,
                        "sname": null,
                        "adress": null,
                        "Parts": [{
                                        "pid": "90",
                                        "pname": "bumper",
                                        "color": "Red"
                                },
                                {
                                        "pid": "91",
                                        "pname": "caliper",
                                        "color": "Blue"
                                }
                        ],
                        "Catalog": []
                }
        ]
}
```

2.

2.1. a.

{AuthorNo}+ ={AuthorNo, AuthorName, AuthorAdress, AuthorEmail} **AuthorNo is not a key**

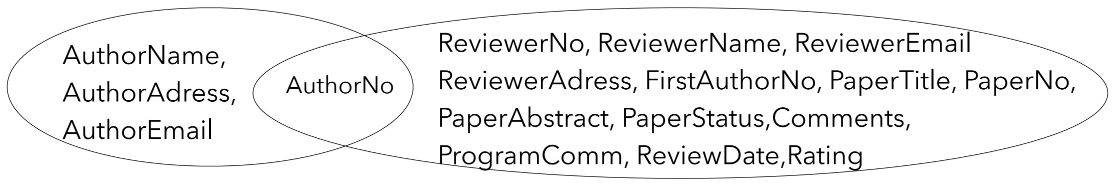
{AuthorEmail}+ ={ AuthorEmail, AuthorNo, AuthorName, AuthorAdress } **AuthorEmail is not a key**

{PaperNo}+ ={PaperNo, FirstAuthorNo, PaperTitle, PaperAbstract, PaperStatus} **PaperNo is not a key**

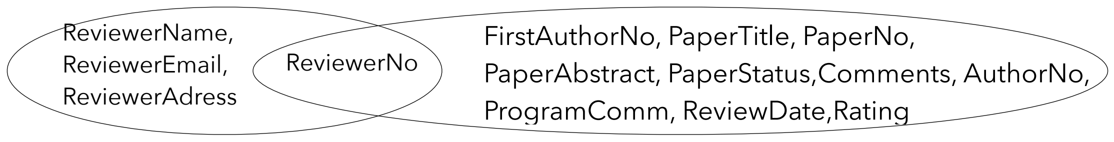
{ReviewerNo}+ = {ReviewerNo, ReviewerName, ReviewerEmail, ReviewerAdress} **ReviewerNo is not a key**

{ReviewerEmail}+={ReviewerNo, ReviewerName, ReviewerEmail,ReviewerAdress}**ReviewerEmail is not a key**

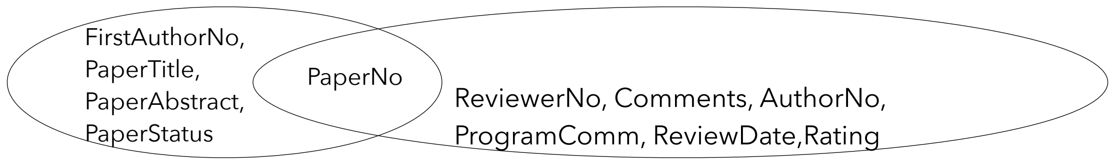
{ReviewerNo,PaperNo}+ = {ReviewerNo, ReviewerName, ReviewerEmail, ReviewerAdress, PaperNo, Rating, Comments,FirstAuthorNo, PaperTitle, PaperAbstract, PaperStatus,ProgramComm,ReviewDate} **not a key**
So, all of the given FDs are bad.
Lets start with **AuthorNo -> AuthorName, AuthorAdress, AuthorEmail**


AuthorName,
AuthorAdress,
AuthorEmail
AuthorNo
ReviewerNo, ReviewerName, ReviewerEmail
ReviewerAdress, FirstAuthorNo, PaperTitle, PaperNo,
PaperAbstract, PaperStatus,Comments,
ProgramComm, ReviewDate,Rating

R1(AuthorNo,AuthorName,AuthorAdress,AuthorEmail) AuthorNo is the Key of R1 and AuthorEmail is unique
Secondly **ReviewerNo -> ReviewerNo, ReviewerName, ReviewerEmail, ReviewerAdress**

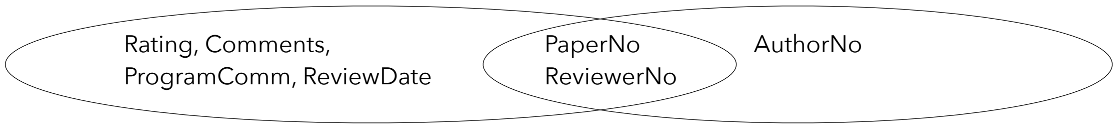

ReviewerName,
ReviewerEmail,
ReviewerAdress
ReviewerNo
FirstAuthorNo, PaperTitle, PaperNo,
PaperAbstract, PaperStatus,Comments, AuthorNo,
ProgramComm, ReviewDate,Rating

R2(ReviewerNo, ReviewerName, ReviewerEmail, ReviewerAdress) ReviewerNo is the Key of R2 and ReviewerEmail is unique
Another FD is **PaperNo->FirstAuthorNo, PaperTitle, PaperAbstract, PaperStatus**


FirstAuthorNo,
PaperTitle,
PaperAbstract,
PaperStatus
PaperNo
ReviewerNo, Comments, AuthorNo,
ProgramComm, ReviewDate,Rating

R3(PaperNo, FirstAuthorNo, PaperTitle, PaperAbstract, PaperStatus) PaperNo is the Key of R3
However, this is not BCNF because ReviewerNo,PaperNo does not cover AuthorNo (not a key). So, we should decompose again.

Thus, finally **ReviewerNo,PaperNo-> Rating, Comments, ProgramComm, ReviewDate**



R4(ReviewerNo,PaperNo, Comments, ProgramComm, ReviewDate,Rating)
(ReviewerNo,PaperNo) is the Key of R4
PaperNo foreign key referencing R3, ReviewerNo foreign key referencing R2

R5(ReviewerNo,PaperNo, AuthorNo)
(ReviewerNo,PaperNo, AuthorNo) is the Key of R5
(AuthorNo) foreign key referencing R1
(ReviewerNo) foreign key referencing R2
(PaperNo) foreign key referencing R3
(ReviewerNo,PaperNo) foreign key referencing R4

b.

Since we used BCNF decomposition it is lossless.

R1 has **AuthorNo -> AuthorName, AuthorAdress, AuthorEmail** and
AuthorEmail -> AuthorNo

R2 has **ReviewerNo->ReviewerNo,ReviewerName,ReviewerEmail, ReviewerAdress** and
ReviewerEmail -> ReviewerNo

R3 has **PaperNo->FirstAuthorNo, PaperTitle, PaperAbstract, PaperStatus**

R4 has **ReviewerNo,PaperNo -> Comments,ProgramComm,ReviewDate,Rating**

R5 has no FD

Thus, since no FD has lost . It is dependency preserving.

2.2.

a.

F= {AC ->BGH,   D->E,   G->B, E->FK,   FD->K, ADF->C, H->BGH}

- AC->B, AC ->G, AC ->H, D->E, G->B, E->F, E->K, FD->K, ADF->C, H->B, H->G, H->H
- A+={A} , C+={C} so AC->B stays,AC->G, AC->H stays
  F+={F} , D+={D,E,F,K} so FD->K deleted
  A+={A} , D+={D,E,F,K} , F+={F} so ADF->C is AD->C
  AC->B, AC ->G, AC ->H, D->E, G->B, E->F, E->K, AD->C, H->B, H->G, H->H
- H+={H,G,B} No need for H->H
  AC->H and H->B No need for AC->B
  AC->H and H->G No need for AC->G
  H->G   and G->B No need for H->B

U= {AC->H , D->E, G->B, E->F, E->K, AD->C, H->B, H->G} b.

U1{AC->H}, U2{D>E} U3{G->B} U4 {E->F, E->K} U5{AD->C} U6{H->G}

R1=(ACH : AC->H)
R2=(DE: D->E)
R3=(GB: G->B)
R4=(EFK: E->F, E->K)
R5=(ADC: AD->C)
R6=(HG: H->G)

ACH+={A,B,C,H,G} not a key
DE+={D,E,F,K} not a key
GB+={G,B} not a key
EFK+={E,F,K} not a key
However;
ADC+={A,B,D,C,E,F,K,H,G} ADC is a key. Thus, no additional relation needed.

2.3.

a. FD that I found are A->B , B->A , C->D , D->C , F->G , G->F , BDF->E BCF->E , ACF->E, ADF->E , BDG->E , BCG->E , ACG->E , ADG->E

A found FD by checking giving queries. If the result's every value is 1 this means that a FD is hold.

```
SELECT COUNT(DISTINCT B)
FROM Sample
GROUP BY A
HAVING COUNT(DISTINCT B) >1;

SELECT COUNT(DISTINCT A)
FROM Sample
GROUP BY B
HAVING COUNT(DISTINCT A) >1;

SELECT COUNT(DISTINCT D)
FROM Sample
GROUP BY C
HAVING COUNT(DISTINCT D) >1;

SELECT COUNT(DISTINCT C)
FROM Sample
GROUP BY D
HAVING COUNT(DISTINCT C) >1;

SELECT COUNT(DISTINCT G)
FROM Sample
GROUP BY F
HAVING COUNT(DISTINCT G) >1;

SELECT COUNT(DISTINCT F)
FROM Sample
GROUP BY G
HAVING COUNT(DISTINCT F) >1;

SELECT COUNT(DISTINCT E)
FROM Sample
GROUP BY B,D,F
HAVING COUNT(DISTINCT E) >1;

SELECT COUNT(DISTINCT E)
FROM Sample
GROUP BY B,C,F
HAVING COUNT(DISTINCT E) >1;

SELECT COUNT(DISTINCT E)
FROM Sample
GROUP BY A,C,F
HAVING COUNT(DISTINCT E) >1;

SELECT COUNT(DISTINCT E)
FROM Sample
GROUP BY A,D,F
HAVING COUNT(DISTINCT E) >1;

SELECT COUNT(DISTINCT E)
FROM Sample
GROUP BY B,C,G
HAVING COUNT(DISTINCT E) >1;

SELECT COUNT(DISTINCT E)
FROM Sample
GROUP BY B,D,G
HAVING COUNT(DISTINCT E) >1;

SELECT COUNT(DISTINCT E)
FROM Sample
GROUP BY A,C,G
HAVING COUNT(DISTINCT E) >1;

SELECT COUNT(DISTINCT E)
FROM Sample
GROUP BY A,D,G
HAVING COUNT(DISTINCT E) >1;
```

b.

```
CREATE TABLE IF NOT EXISTS AB (
  A CHAR(1) PRIMARY KEY NOT NULL,
  B CHAR(7) UNIQUE
);

CREATE TABLE IF NOT EXISTS CD (
  C INT PRIMARY KEY NOT NULL ,
  D CHAR(15) UNIQUE);

CREATE TABLE IF NOT EXISTS FG (
  F INT PRIMARY KEY NOT NULL ,
  G CHAR(15) UNIQUE
);

CREATE TABLE IF NOT EXISTS ACFE (
  A CHAR(1) REFERENCES AB(A),
  C INT REFERENCES CD(C),
  F INT REFERENCES FG(F),
  E INT,
  PRIMARY KEY (A, C, F)
);
```

c.

```
INSERT INTO AB (A, B)
SELECT DISTINCT A, B
FROM Sample;

INSERT INTO CD (C, D)
SELECT DISTINCT C, D
FROM Sample;

INSERT INTO FG (F, G)
SELECT DISTINCT F, G
FROM Sample;
INSERT INTO ACFE (A,C,F,E)

SELECT DISTINCT A,C,F,E
FROM Sample;
```

3.

```
CREATE TABLE IF NOT EXISTS Customer (
  CustNo VARCHAR PRIMARY KEY,
  CustFirstName VARCHAR,
  CustLasName VARCHAR,
  CustCity VARCHAR,
  CustState VARCHAR,
  CustZip VARCHAR,
  CustBal VARCHAR
);

CREATE TABLE IF NOT EXISTS Employee (
  EmpNo  VARCHAR DEFAULT "007",
  EmpFirstName VARCHAR,
  EmpLastName VARCHAR,
  EmpPhone VARCHAR,
  EmpEmail VARCHAR,
  EmpDeptName VARCHAR,
  EmpStatus VARCHAR,
  EmpSalary INT,
  supervisor VARCHAR REFERENCES Employee(EmpNo) ON DELETE SET
DEFAULT,
CHECK(EmpFirstName  NOT SIMILAR TO  '%'|| EmpEmail ||'%'),
CHECK(EmpLastName  NOT SIMILAR TO  '%'|| EmpEmail ||'%'),
PRIMARY KEY(EmpNo)
);

CREATE TABLE IF NOT EXISTS Product (
  ProdNo  VARCHAR PRIMARY KEY,
  ProdName VARCHAR,
  ProdPrice INT,
  ProdShipDate TIMESTAMP
);

CREATE TABLE IF NOT EXISTS Order (
  OrdNo VARCHAR PRIMARY KEY,
  CustNo VARCHAR REFERENCES Customer(CustNo) ON DELETE CASCADE,
  EmpNo VARCHAR REFERENCES Employee(EmpNo) ON DELETE SET NULL,
  OrdDate TIMESTAMP,
  OrdName VARCHAR,
  OrdCity VARCHAR ,
  OrdZip VARCHAR,
 CHECK( OrdCity  SIMILAR TO '%'|| OrdName '||%')
);

CREATE TABLE IF NOT EXISTS Contains (
  OrdNo VARCHAR REFERENCES Order(OrdNo) ON DELETE CASCADE ,
  ProdNo VARCHAR REFERENCES Product(ProdNo) ON DELETE CASCADE,
  Qty INT,
  PRIMARY KEY (OrdNo, ProdNo)
  CHECK(Qty >=3),
);
```

```
CREATE ASSERTION CHECK
(NOT EXISTS(
  SELECT OrdNo
  FROM O.OrdNo=C.OrdNo
  GROUP BY OrdNo
  HAVING COUNT(ProdNo)< 30
));
```

```
CREATE TRIGGER TheTrigger
AFTER UPDATE OF EmpSalary ON Employee
REFERENCING
  OLD ROW AS OldTuple
  NEW ROW AS NewTuple
FOR EACH ROW
  WHEN((NewTuple.EmpSalary-OldTuple.EmpSalary)/
OldTuple.EmpSalary >0.15)
    UPDATE Employee
    SET EmpStatus='Successful'
    WHERE EmpNo = OldTuple.EmpNo;
```