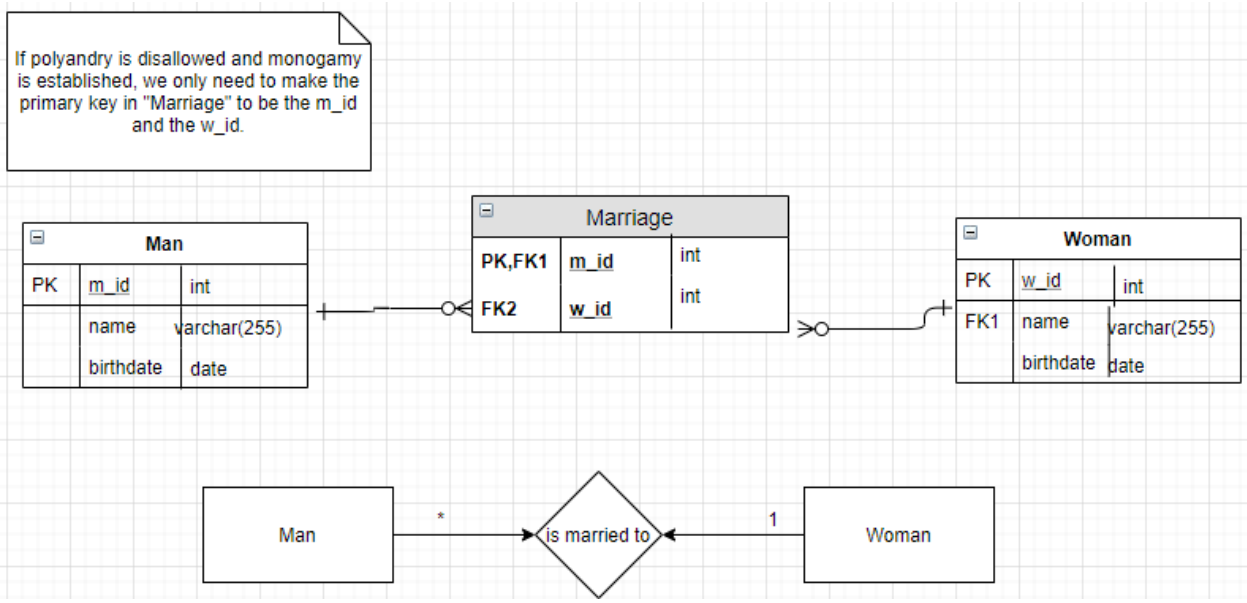
FINAL ASSIGNMENT

Database systems

Atanas Latinov
201811467

Exercise 1



DDL code

```
create table man (
        m_id int primary key,
   `name` varchar(255),
   birthdate date
);

create table woman (
        w_id int primary key,
   `name` varchar(255),
   birthdate date
);

create table marriage (
m_id int primary key,
w_id int,
```

```
foreign key (m_id) references man(m_id),
foreign key (w_id) references woman(w_id)
);
```

3. Check note in the image above for the modification if polyandry is disallowed.

Exercise 2

```
SET FOREIGN_KEY_CHECKS=OFF;
DROP TABLE IF EXISTS object;
DROP TABLE IF EXISTS property;
DROP TABLE IF EXISTS relationship;
SET FOREIGN_KEY_CHECKS=ON;

create table object (
        object varchar(100) primary key,
   description varchar(100) not null
);

create table property (
        object varchar(100),
   property varchar(100) not null,
   `value` int,
   primary key(object, property),
   foreign key (object) references object(object)
);

create table relationship (
```

```
        object1 varchar(100),
    relationship varchar(100),
    object2 varchar(100),
    primary key(object1, relationship, object2),
    foreign key (object1) references object(object),
    foreign key (object2) references object(object)
);

insert into object values("Earth", "the 3rd planet from the Sun in the Solar System, which is part of the Milky Way galaxy");
insert into object values("Universe", "endless space of space");
insert into object values("car", "used for transportation");
insert into object values("driver", "using the car");
insert into object values("chef", "person preparing food");
insert into object values("stove", "cooking appliance");
insert into object values("Piers", "husband");
insert into object values("Cynthia", "wife");
insert into object values("Maggie", "friend");
insert into object values("Johnson", "doctor");
insert into object values("Ella", "child");
insert into object values("computer", "electronic device for computation");
insert into object values("mobile phone", "electronic device for communicating");
insert into object values("iphone", "electronic device for communicating");
insert into object values("huawei", "electronic device for communicating");
insert into object values("pixel", "electronic device for communicating");
insert into object values("samsung", "electronic device for communicating");
insert into object values("sony", "electronic device for communicating");
insert into object values("motorola", "electronic device for communicating");
insert into object values("htc", "electronic device for communicating");
```

```
insert into object values("nokia", "electronic device for communicating");
insert into object values("oneplus", "electronic device for communicating");
insert into object values("elephone", "electronic device for communicating");
insert into object values("xiaomi", "electronic device for doing calculations");
insert into object values("ipad", "electronic device for various purposes");
insert into object values("smartwatch", "smart wearable electronic device");
insert into object values("microwave", "electronic device for heating food");
insert into object values("kettle", "electronic device for boiling water");
insert into object values("TV", "electronic device for watching content");
insert into object values("router", "electronic device for setting up LANs");
insert into object values("projector", "electronic device for projecting video content");
insert into object values("mind", "some people have it");
insert into object values("body", "some people don't have it");

insert into relationship values ("driver", "driving", "car");
insert into relationship values ("chef", "cooking with", "stove");
insert into relationship values ("Piers", " is married to ", "Cynthia");
insert into relationship values ("Maggie", " is the best friend of ", "Cynthia");
insert into relationship values ("Johnson", " is the personal doctor of ", "Cynthia");
insert into relationship values ("Johnson", " is the personal doctor of ", "Ella");
insert into relationship values ("Ella", " is the daughter of ", "Cynthia");
insert into relationship values ("Cynthia", " is the mother of ", "Ella");

insert into relationship values ("Earth", " is the home of ", "car");
insert into relationship values ("Earth", " is the home of ", "driver");
insert into relationship values ("Earth", " is the home of ", "chef");
insert into relationship values ("Earth", " is the home of ", "stove");
insert into relationship values ("Earth", " is the home of ", "Piers");
insert into relationship values ("Earth", " is the home of ", "Cynthia");
```

```
insert into relationship values ("Earth", " is the home of ", "Maggie");
insert into relationship values ("Earth", " is the home of ", "Johnson");
insert into relationship values ("Earth", " is the home of ", "Ella");
insert into relationship values ("Earth", " is the home of ", "computer");
insert into relationship values ("Earth", " is the home of ", "mobile phone");
insert into relationship values ("Earth", " is the home of ", "iphone");
insert into relationship values ("Earth", " is the home of ", "huawei");
insert into relationship values ("Earth", " is the home of ", "pixel");
insert into relationship values ("Earth", " is the home of ", "samsung");
insert into relationship values ("Earth", " is the home of ", "sony");
insert into relationship values ("Earth", " is the home of ", "motorola");
insert into relationship values ("Earth", " is the home of ", "htc");
insert into relationship values ("Earth", " is the home of ", "nokia");
insert into relationship values ("Earth", " is the home of ", "oneplus");
insert into relationship values ("Earth", " is the home of ", "elephone");
insert into relationship values ("Earth", " is the home of ", "xiaomi");
insert into relationship values ("Earth", " is the home of ", "ipad");
insert into relationship values ("Earth", " is the home of ", "smartwatch");
insert into relationship values ("Earth", " is the home of ", "microwave");
insert into relationship values ("Earth", " is the home of ", "kettle");
insert into relationship values ("Earth", " is the home of ", "TV");
insert into relationship values ("Earth", " is the home of ", "router");
insert into relationship values ("Earth", " is the home of ", "projector");
insert into relationship values ("Earth", " is the home of ", "Earth");
insert into relationship values ("Universe", " is the home of ", "Earth");
insert into relationship values ("Earth", " is the home of ", "Universe");
insert into relationship values ("mind", " is the home of the ", "body");
insert into relationship values ("body", " is the home of the ", "mind");
insert into relationship values ("body", " is the home z ", "mind");
```

```
-- insert into relationship values ("mind", " is the home z ", "body");


-- Q1

select o.description

from object o, relationship r

where o.object = r.object1 and r.object2 = "Cynthia";


-- Q2

select o1.object, o2.object

from object o1, object o2, relationship r where o1.object <= o2.object and o1.object = r.object1 and (r.object2 = "Cynthia")

group by o1.object, o2.object

having (o1.object <> o2.object) and o2.object in (

select o1.object

from object o1, object o2, relationship r where o1.object > o2.object and o1.object = r.object1 and (r.object2 = "Cynthia") and o2.object = r.object2

group by o2.object, o1.object);


-- Q3


select o.description, count(object) as cnt

from object o

where o.description like '%electronic%'

group by o.description

having (cnt > 10);


-- Q4


SELECT o.object
```

```
FROM object o
WHERE NOT EXISTS (
        SELECT o2.object
   FROM object o2
   WHERE NOT EXISTS (
         SELECT * FROM relationship r WHERE r.object1 = o.object AND r.object2 = o2.object
   )
);


-- Q5
-- This query is returning the set of all objects, with which no other object is in relationship (it is not a right-hand side object of any relationship)
select o.object
from object o
where o.object not in (select r.object2 from relationship r);


-- Q6


SELECT distinct r.object1, r.object2
FROM relationship r
WHERE EXISTS (
        select * from relationship r2
                where r2.object1 = r.object2 and r2.relationship = r.relationship and r2.object2 = r.object1
      and not exists (
                        select * from relationship r3 where r2.object1 > r3.object1 and r2.object2 <= r3.object2
      )
        )
        group by r.object1, r.relationship, r.object2
```

having (r.object1 <> r.object2)

Exercise 3

R = { A , B, C, D }

F = { A -> C

B, C -> D

A, B -> C}

Two approaches were executed:

First approach:

1. Find the attributes that are neither on the left and right side

> (none)

2. Find attributes that are only on the right side

> D

3. Find attributes that are only on the left side

> A, B

4. Combine the attributes on step 1 and 3

> A, B, D

5. Test if the closures of attributes on step 4 are all the attributes

> A+ = AC

B+ = B

D+ = D

> we continue with combinations of attributes

AB+ = ABCD because A->C and B, C -> D which means AB->D

Hence AB+ is our candidate key.

Second approach:

Compute the attribute closures:

A+ = AC

B+ = B

C+ = C

D+ = D

AB+ = ABCD

AC+ = AC

BC+ = BCD

Candidate key is: **AB**

2. Minimal cover

Rewrite FDs in canonical form

A -> C

BC -> D

AB -> C

We can minimize AB -> C, because A->C

We are left with:

A->C

BC->D

A->C

After we remove redundancies, we have

A->C

BC->D

We cannot eliminate any more functional dependencies.

3. 3NF

3.1 Find minimal cover and candidate keys (check above)

Minimal cover = {

A -> C

BC -> D

}

Candidate key: AB

Both functional dependencies in the min cover violate 3NF since the left sides are not super keys and the right sides are sets of non-key attributes.

- each FD should have its own relation. Also, the candidate key A, B is 3NF normalized and we get R3, shown below:

Relations:

R1(A, C) with F1 = A -> C; R2(B, C, D) with F2 = {B, C -> D}; R3(A, B)

Now all non-key attributes (C, D) are fully functional dependent only on the primary key (AB).

It is lossless because the Functional dependencies are preserved with this decomposition A->C in R1 and BC->D in R2 and every non-prime attribute of R is non-transitively dependent on every key attribute of R. Also, 3NF always ensures a lossless decomposition.

4. Why R with F is not in BCNF

A relation is in BCNF if and only if the arrow in every FD is an arrow out of a candidate key.

A in (A -> C) is not a candidate key, neither is BC in (BC -> D).

5. Prove that A, B -> D is in F+, the closure of F.

{A, B} -> {D} because {B, C} -> D; but A -> C, hence after substitution: {A, B} -> D

6. Show the first decomposition that is not dependency preserving

The decomposition which is not preserving is extracted if we start with the FD:

A -> C

It violates BCNF in R, so we decompose into a relation only with attributes of the violating FD and another relation with the attributes from: R – RHS(FD):

R1(A, C) and R2(A, B, D)

Now A -> C holds for R1. **BC -> D** does **not** hold because C is not in R2, so we have lost that dependency.

7. Show the second decomposition which is dependency preserving

If we start the algorithm with the FD {B, C} -> D we check on R. It violates BCNF so we decompose into:

R1(B, C, D) and R2(A, B, C). Now {B, C} -> D holds for R1, but A -> C violates BCNF for R2, so we need to decompose R2 into:

R3(A, C) and R4(A, B).

We end up with R1(B, C, D); R3(A, C); and R4(A, B) with all dependencies holding.

Exercise 4

1. 2400 Book tuples with uniformly distributed non-key values
   6 distinct category values meaning 2400/6=400 tuples per category value.

2. Book file stored in 1200 blocks
   1200 blocks / 6 distinct category values = 200 blocks for a category value

3. Purchasing has 21600 tuples. Dividing this by 12 months (condition if for month being equal to `March`) yields 1800 tuples satisfying that condition. We divide by 12 because we know values are uniformly distributed.
   21600/12 = 1800
4. 1800 blocks used for storing the Purchasing table. That means:

   21600 / 1800 = 12 tuples per block.

5. Using (3.) we can calculate that Purchasing tuples, qualifying the month condition in the following way:

   We have 1800 tuples in total, satisfying the month condition and there are 12 tuples per block

   1800/12 = 150 blocks

6. 1 + 1 for the 2 levels + 200 blocks for category tuples
7. 1+1 for the 2 levels + 1800 blocks for the tuples satisfying the conditions

Exercise 5

The schedule is executable in compliance with the 2PL method.

The following locks have been used:

S1(A); **R1(A)**; X1(B); **W1(B)**; S2(C); **R2(C)**; S1(C); **R1(C)**; U1(All); S2(B); **R2(B)**; S2(A); U2(B); X3(B); **W3(B)**; **R2(A)**; U2(All); X3(A); **W3(A)**; U3(All);

No conflicts occur between the locks. Also, transactions do not request additional locks once they release any locks. Locks are handled in compliance with the properties of the two distinct phases (expanding phase and shrinking phase).

If we want to introduce a change that would make it impossible to execute it by 2PL, we simply change the release and acquisition of locks between transactions 2 and 3.

Let us focus on the following fragment of the schedule:

S2(B); **R2(B)**; S2(A); U2(B); X3(B); **W3(B)**; **R2(A)**; U2(All);

By moving the acquisition of the S2(A) lock after the release of the U2(B) lock, we are violating the shrinking phase of transaction 2, which stipulates that no locks should be acquired after locks are released for a transaction. One such modification would be:

S2(B); **R2(B)**; U2(B); X3(B); **W3(B)**; S2(A); **R2(A)**; U2(All);

The S2(A) lock acquisition at this place would mean that after we have begun releasing locks with the U2(B) and thus entering the shrinking phase for transaction 2, we do a violation by acquiring an S2(A) lock after that.