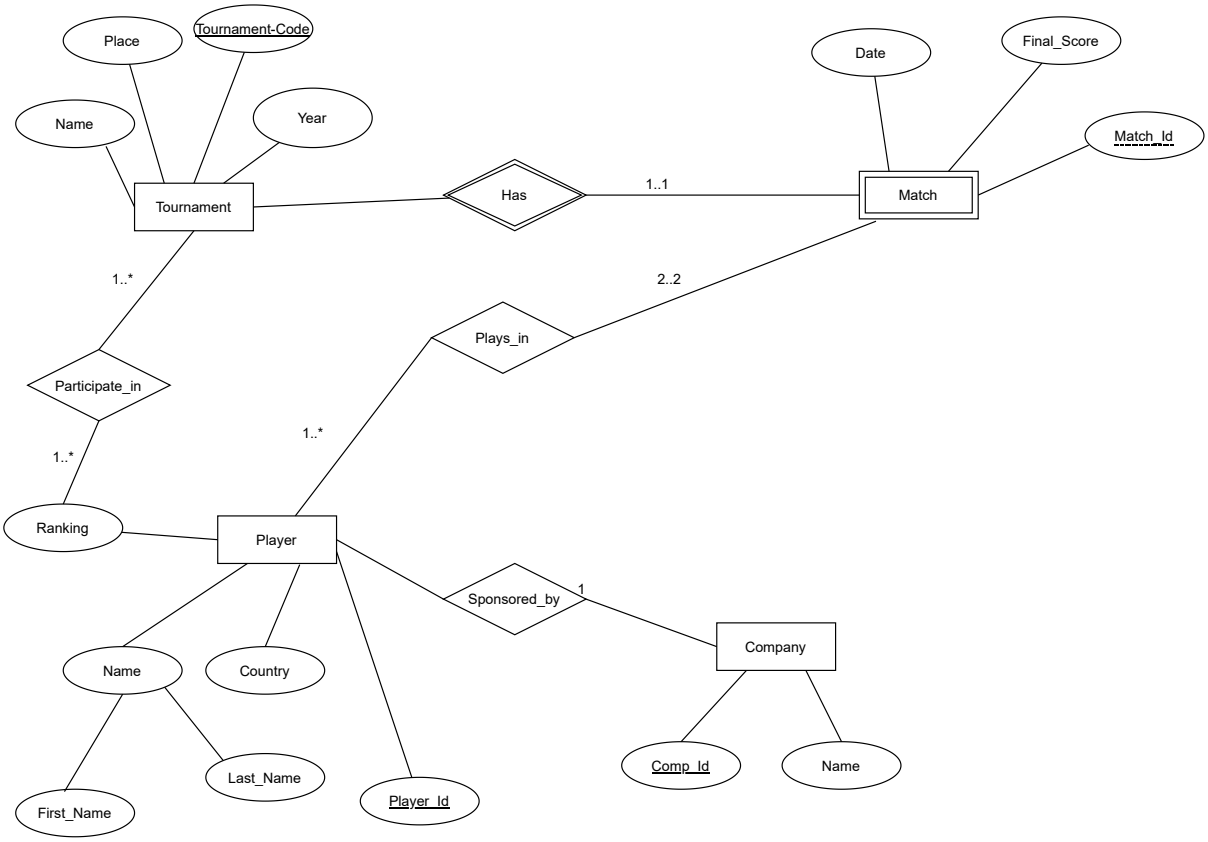
Βάσεις Δεδομένων
Σειρά Ασκήσεων

Ομάδα M

Νικήτας Τσίννας, el18187
Αναστάσιος Παπαζαφειρόπουλος, el18079
Νικόλαος Παγώνας, el18175

Exercise 1

A.



Στο παραπάνω διάγραμμα ER υπάρχουν:

- Το σύνολο οντοτήτων Tournament με Attributes: Name, Place, Year, Tournament-Code και primary key: Tournament-Code που θα κωδικοποιεί μοναδικά το τουρνουά. Αυτό είναι απαραίτητο γιατί καμία άλλη πληροφορία δε μπορεί να καθορίσει μοναδικά ένα τουρνουά.
- Το σύνολο οντοτήτων Player με Attributes: Name(First_Name, Last_Name), Country, Ranking και Player_Id, το οποίο θα μπορούσε να είναι για παράδειγμα ο αριθμός ταυτότητας εξαρτάται από τους διοργανωτές. Αυτό είναι και το primary key μιας και εδώ καμία από τις υπόλοιπες πληροφορίες δε μπορεί να καθορίσει μοναδικά έναν παίκτη.
- Το σύνολο οντοτήτων Company με Attributes: Name, Comp_Id και primary key: Comp_Id.

- Η N:M σχέση Participate_in που συνδέει το Tournament με το Player, καθώς πολλοί παίκτες μπορούν να συμμετέχουν σε ένα τουρνουά, αλλά και ένας παίκτης να συμμετέχει σε πολλά τουρνουά. Επίσης, η σχέση αυτή είναι total participation ως προς τον Player και ως προς το τουρνουά, καθώς δε γίνεται να υπάρξει τουρνουά χωρίς παίκτες από τη μία και από την άλλη θεωρήσαμε ότι δε γίνεται ένας παίκτης να μην έχει συμμετάσχει ποτέ σε τουρνουά, γιατί τότε δε θα θεωρούνταν παίκτης.
- Η 1:N σχέση Sponsored_by που συνδέει το Player με το Company.
- Το αδύναμο σύνολο οντοτήτων Match με Attributes: Date, Final_Score και Match_Id. Είναι αδύναμο, καθώς υλοποιήθηκε με τη λογική ότι κάθε match αναφέρεται σε ένα τουρνουά στο οποίοπραγματοποιήθηκε, άρα και δε μπορεί να καθοριστεί χωρίς το primary key του Tournament (Tournament_Code).
- Η 2:N σχέση Plays που συνδέει το Player με το Match, καθώς δύο παίκτες αγωνίζονται σε κάθε παιχνίδι. Επίσης, εδώ θεωρήσαμε ότι δε γίνεται να υπάρξει παίκτης που να μην έχει δώσει ποτέ παιχνίδι.

B.

Από το παραπάνω ER διάγραμμα προκύπτουν:

- Player(Player_Id, First_Name, Last_Name, Country, Ranking, Comp_Id)

  με foreign key: Comp_Id, καθώς η σχέση Sponsored_By είναι 1:N
- Company(Comp_Id, Name)
- Tournament(Tournament_Code, Name, Place, Year)
- Match(Match_Id, Tournament_Code, Date, Final_Score)

  με foreign key: Tournament Code ως πρωτεύον κλειδί της ισχυρής οντότητας.
- Participates_in(Player_Id, Tournament_Code), καθώς η σχέση αυτή στο ER είναι N:M, με primary

  και foreign keys, τα primary keys των οντοτήτων που συνδέονται.
- Plays_in(Player_Id, Match_Id), καθώς η σχέση αυτή στο ER είναι 2:N, με primary και foreign keys,

  τα primary keys των οντοτήτων που συνδέονται.

Exercise 2

Q_1 :

$$\Pi_{\text{pid}}(\sigma_{\text{companyname="Google"}}(\text{Person}\bowtie\text{Company}))\bigcap$$

$$\Pi_{\text{pid}}\Big(\sigma_{\text{sharenum}>500}(\sigma_{\text{companyname="Facebook"}}(\text{Shares}\bowtie\text{Company}))\Big)$$

Q_2 :

$$\Pi_{\text{Person.pid}}(\sigma_{\text{Person.cid = Shares.cid}\wedge\text{Person.managerid = Shares.pid}\wedge\text{Sharesnum}>0}(\text{Person}\times\text{Shares}))$$

Q_3 :

$$\Pi_{\text{pid}}\Big({}_{\text{pid}}\mathcal{G}_{(\text{count(cid)}>2)}(\sigma_{\text{sharesnum}>0}(\text{Shares}))\Big)$$

Q_4 :

$$\Pi_{\text{pid,cid}}(\sigma_{\text{sharesnum}>0}(Shares))\,/\,\Pi_{\text{cid}}(Company)$$

Exercise 3

Q1:

Βρες το id και το όνομα των καταστημάτων που στεγάζουν λιγότερο από 100 υπαλλήλους ή βρίσκονται στην Αθήνα.

```
select s.sname

from Store as s

where city='Athens' or employee_number <= 100
```

Q2:

Βρες το όνομα των καταστημάτων που προμηθεύονται μολύβια

```
select st.sname

from store as st
inner join (supply as su
   inner join Goods as g
   on g.gid=su.gid)
on st.storeid = su.storeid

where gname='μολύβι'
```

Q3:

Βρες το όνομα και την πόλη των καταστημάτων που προμηθεύονται τα προϊόντα που προμηθεύεται το κατάστημα με id=0808.

```
select st.sname, st.city

from supply as su

where not exists(
   (Select su1.gid
   from supply as su1
   where su1.storeid = 0808)

   except

   (select su2.gid
   from supply as su2
   where su2.storeid = st.storeid
)
```

Q4:

```
with aux as (
   select count(su.gid) as total_goods, su.storeid
   from Supply as su
   group by su.storeid
   order by total_goods desc
   limit 5)

select distinct st.sname

from Store as st natural join aux
```

Q5:

```
select st.city

from store as st
inner join (supply as su
   inner join goods as g
   on g.gid = su.gid)
on st.storeid = su.storeid

having max(price)>200
```

Q6:

```
select distinct su.gid

from (Supply as su inner join Store as st on su.storeid = st.storeid)
inner join Goods as g
on su.gid = g.gid

where st.city = 'Athens'
group by su.gid
having count (su.storeid) = (select count(storeid)
   from Store as st
   where st.city = 'Athens')
```

Q7:

```
(select gid

from store as st
natural join supply as su

where st.city = 'Athens')

except

(select gid

from Store as st
natural join Supply as su

Where st.city = 'Patras')
```

Exercise 4

A.

Find all attributes that have not appeared on the RHS of any FD: $S_1 = \{B, C\}$
Find all attributes that have not appeared on the LHS of any FD: $S_2 = \{A\}$

Compute the closure set of S_1: $S_1^+ = \{A, B, C, D, E\}$.

$S_1^+ = R$, so $\{B, C\}$ is the only candidate key.

B.

Computing a canonical cover for F:

Initially, $F_c = F$.

Iteration #1:

Can't use union rule.

Checking $B \to EA$:

Removing B:
$\varnothing^+ = \varnothing$, doesn't contain B, so B is needed.

Removing E:
$F_c' = \{B \to A, EBC \to D, BED \to A\}$
$(B)^+ = (AB)$, doesn't contain E, so E is needed.

Removing A:
$F_c' = \{B \to E, EBC \to D, BED \to A\}$
$(B)^+ = (BE)$, doesn't contain A, so A is needed.

Checking $EBC \to D$:

Removing E:
$(BC)^+ = (ABCDE)$, contains E, so E is extraneous in $EBC \to D$.

Now $F_c = \{B \to EA, BC \to D, BED \to A\}$

Iteration #2:

Can't use union rule.

Checking $B \to EA$:

Removing B:
$\varnothing^+ = \varnothing$, doesn't contain B, so B is needed.

Removing E:
$F_c' = \{B \to A, BC \to D, BED \to A\}$
$(B)^+ = (AB)$, doesn't contain E, so E is needed.

Removing A:
$F_c' = \{B \to E, BC \to D, BED \to A\}$
$(B)^+ = (BE)$, doesn't contain A, so A is needed.

Checking $BC \to D$:

Removing B:
$(C)^+ = (C)$, doesn't contain B, so B is needed.

Removing C:
$(B)^+ = (ABE)$, doesn't contain C, so C is needed.

Removing D:
$F_c' = \{B \to EA, BED \to A\}$
$(BC)^+ = (ABCE)$, doesn't contain D, so D is needed.

Checking $BED \to A$:

Removing B:
$(ED)^+ = (ED)$, doesn't contain B, so B is needed.

Removing E:
$(BD)^+ = (ABDE)$, contains E, so E is extraneous in $BED \to A$.

Now $F_c = \{B \to EA, BC \to D, BD \to A\}$

Iteration #3:

Can't use union rule.

Checking $B \to EA$:

Removing B:
$\varnothing^+ = \varnothing$, doesn't contain B, so B is needed.

Removing E:
$F_c' = \{B \to A, BC \to D, BD \to A\}$
$(B)^+ = (AB)$, doesn't contain E, so E is needed.

Removing A:
$F_c' = \{B \to E, BC \to D, BD \to A\}$
$(B)^+ = (BE)$, doesn't contain A, so A is needed.

Checking $BC \to D$:

Removing B:
$(C)^+ = (C)$, doesn't contain B, so B is needed.

Removing C:
$(B)^+ = (ABE)$, doesn't contain C, so C is needed.

Removing D:
$F_c' = \{B \to EA, BD \to A\}$
$(BC)^+ = (ABCE)$, doesn't contain D, so D is needed.

Checking $BD \to A$:

Removing B:
$(D)^+ = (D)$, doesn't contain B, so B is needed.

Removing D:
$(B)^+ = (ABE)$, doesn't contain D, so D is needed.

Removing A:
$F_c' = \{B \to EA, BC \to D\}$
$(BD)^+ = (ABDE)$, contains A, so A is extraneous in $BD \to A$.

Now $F_c = \{B \to EA, BC \to D\}$

Iteration #4:

Can't use union rule.

Checking $B \to EA$:

Removing B:
$\varnothing^+ = \varnothing$, doesn't contain B, so B is needed.

Removing E:
$F_c' = \{B \to A, BC \to D\}$
$(B)^+ = (AB)$, doesn't contain E, so E is needed.

Removing A:
$F_c' = \{B \to E, BC \to D\}$
$(B)^+ = (BE)$, doesn't contain A, so A is needed.

Checking $BC \to D$:

Removing B:
$(C)^+ = (C)$, doesn't contain B, so B is needed.

Removing C:
$(B)^+ = (ABE)$, doesn't contain C, so C is needed.

Removing D:
$F_c' = \{B \to EA\}$
$(BC)^+ = (ABCE)$, doesn't contain D, so D is needed.

F_c didn't change, so the canonical cover is $\{B \to EA, BC \to D\}$
Thus, the minimal cover is $\{B \to A, B \to E, BC \to D\}$

C.

R is in 1NF, assuming all attributes in R are atomic.

But R is not in 2NF, since it has a non-prime attribute (A) that is dependent on a proper subset (B) of a candidate key (BC). This follows from $B \to EA$.

Thus, it is also not in 3NF, BCNF, etc. and the best/strictest normal form is 2NF.

D.

$F_c = \{B \to EA, BC \to D\}$

For each functional dependency, add LHS and RHS attributes in a new schema:

$R_1 = (A, B, E)$
$R_2 = (B, C, D)$

Since R_2 contains a candidate key for R ($\{B, C\}$), there is no need to create a new schema.

Finally, since no schema is contained in another schema, we are done.

Thus, the 3NF decomposition produced by the algorithm is:

$R_1(A, B, E)$, with $FD_1 = \{B \to AE\}$
$R_2(B, C, D)$, with $FD_2 = \{BC \to D\}$

Exercise 5

A.

Find all attributes that have not appeared on the RHS of any FD: $S_1 = B$
Find all attributes that have not appeared on the LHS of any FD: $S_2 = D$

Compute the closure set of S_1: $S_1^+ = \{B, D\}$

$S_1^+ \neq R$, so for each attribute x in $R - B$, test whether $A \cup \{x\}$ is a candidate key:

$(S_1 \cup \{A\})^+ = \{A, B\}^+ = \{A, B, C, D\} = R$, so $\{A, B\}$ is a candidate key.
$(S_1 \cup \{B\})^+ = \{B\}^+ = \{B, D\}$
$(S_1 \cup \{C\})^+ = \{B, C\}^+ = \{A, B, C, D\} = R$, so $\{B, C\}$ is a candidate key.

Thus, the candidate keys are $\{A, B\}$ and $\{B, C\}$.

B.

$result = \{R\} = \{ABCD\}$

Check if $R_i = ABCD$ is in BCNF:

for each subset a of R_i, check if a^+ contains all attributes of R_i, or no attributes of $R_i - a$:

$a = A$:
$a^+ = A$
$R_i - a = BCD$
a^+ contains no attributes of $R_i - a$. OK

$a = B$:
$a^+ = BD$
$R_i - a = ACD$
a^+ doesn't contain all attributes of R_i, and it contains attribute D of $R_i - a$. NOT OK

Thus, $a \to (a^+ - a) \cup R_i \equiv B \to D$ breaks BCNF rules.

$result = (result - R_i) \cup (R_i - D) \cup (B, D) = \{ABC, BD\}$

BD is in BCNF by construction.

Check if $R_i = ABC$ is in BCNF:

for each subset a of R_i, check if a^+ contains all attributes of R_i, or no attributes of $R_i - a$:

$a = A$:
$a^+ = A$
$R_i - a = BC$
a^+ contains no attributes of R˙i - a. OK

$a = B$:
$a^+ = BD$

$R_i - a = AC$
a^+ contains no attributes of $R_i - a$. OK

$a = C$:
$a^+ = AC$
$R_i - a = AB$
a^+ doesn't contain all attributes of R_i, and it contains attribute A of $R_i - a$. NOT OK

Thus, $a \to (a^+ - a) \cup R_i \equiv C \to A$ breaks BCNF rules.

$result = (result - R_i) \cup (R_i - A) \cup (C, A) = \{BD, BC, CA\}$

CA is in BCNF by construction.

Check if $R_i = BC$ is in BCNF:

for each subset a of R_i, check if a^+ contains all attributes of R_i, or no attributes of $R_i - a$:

$a = B$:
$a^+ = BD$
$R_i - a = C$
a^+ contains no attributes of $R_i - a$. OK

$a = C$:
$a^+ = CA$
$R_i - a = B$
a^+ contains no attributes of $R_i - a$. OK

$a = BC$:
$a^+ = ABCD$
a^+ contains all attributes of R_i. OK

Thus, BC is in BCNF.

Note: Normally we need not check if relation schemas with 2 attributes are in BCNF (they always are). Nonetheless, we run the algorithm for completeness' sake.

We are done.

The BCNF decomposition produced by the algorithm is:

$R_1(BD)$, with $FD_1 = \{B \to D\}$
$R_2(BC)$, with $FD_2 = \varnothing$
$R_3(AC)$, with $FD_3 = \{C \to A\}$

We see that the functional dependency $AB \to C$ is not preserved. This is normal, since it is not always possible to get a BCNF decomposition that is dependency preserving.