Név: Barna Beáta

Neptun kód: PPOWE8

Kvíz

Adatbázis alapú rendszerek kötelező feladat

Feladat leírás

A program feladata első sorban a szórakoztatás, lehetőséget biztosít a felhasználóknak, hogy különböző nehézségű kérdésekhez kapcsolódó kvízeket tölthessenek ki. Egy gyors regisztrációt követően már használatba is vehetik az oldalt.

Funkciók:

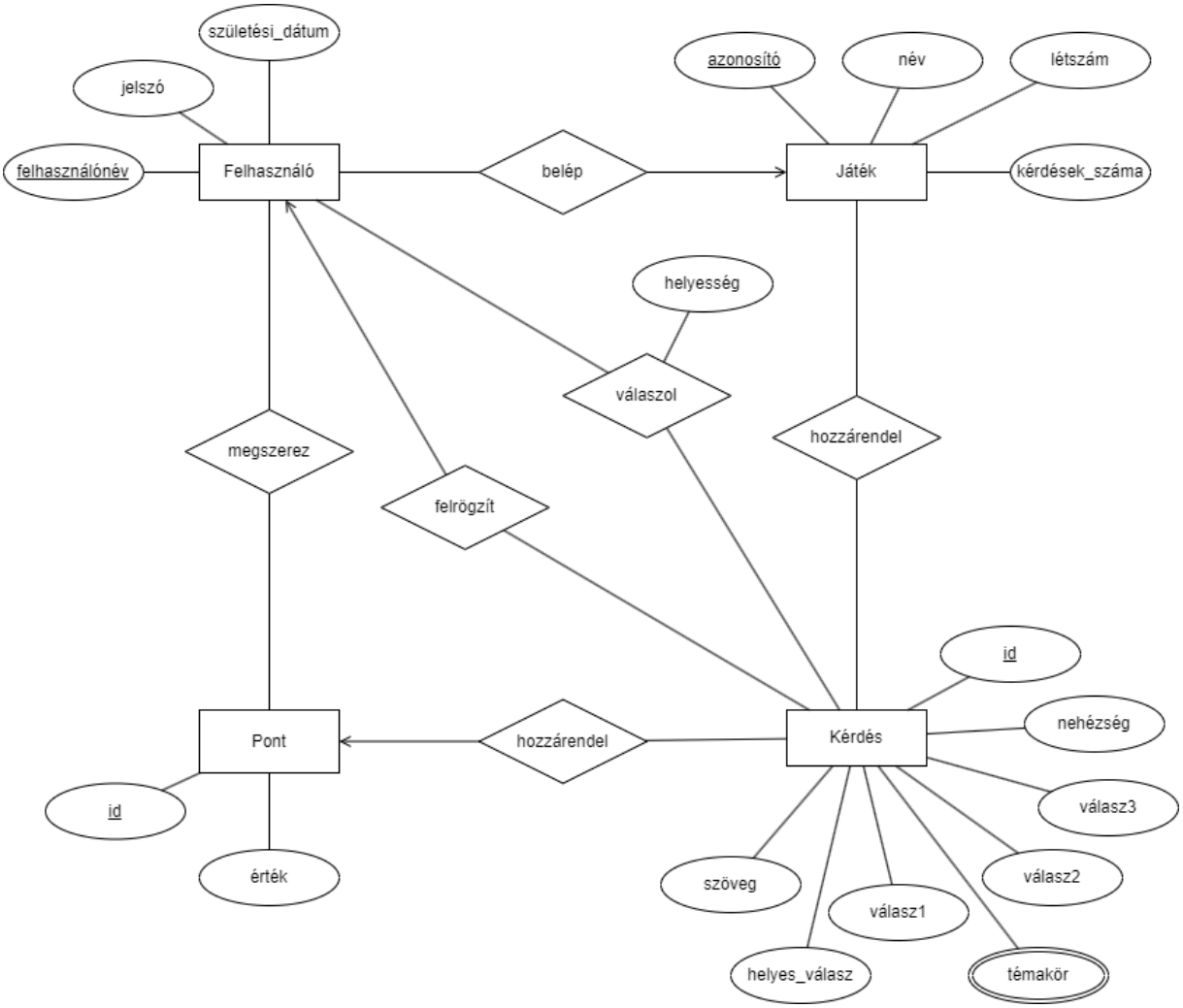
- Regisztráció és bejelentkezés
- Játékszoba készítési lehetőség
- Játékszobához való csatlakozás
- Különböző nehézségi szintű és témájú kérdések elérhetősége
- Különböző nehézségi szintű és témájú kérdések feltöltése
- Eredmények nyomon követése és rangsorolás a felhasználók között
- Megszerzett pontjaink nyomon követése
- Egyéni és összetett eredmények megtekintése és összehasonlítása a felhasználók között

Megvalósítás:

A kvíz oldal megvalósítása a webfejlesztés területére koncentrál, az alábbi technológiák használatával:

- Frontend: HTML, CSS, Bootstrap, JavaScript
- Backend: PHP
- Kódszerkesztő: Visual Studio Code
- Adatbázis-kezelő rendszer: Oracle Database
- A hozzá tartozó programozási nyelv: PL/SQL

Egyed-kapcsolat modell



Relációs adatbázisséma

FELHASZNÁLÓ (felhasználónév, jelszó, életkor, *JATÉK.azonosító*)

JÁTÉK (azonosító, név, létszám, kérdések_száma, nehézség, belépve)

HOZZÁRENDEL (*JÁTÉK.azonosító*, *KÉRDÉS.id*)

KÉRDÉS (id, szöveg, válasz1, válasz2, válasz3, helyes_válasz, témakör, nehézség, *PONT.id*, *FELHASZNÁLÓ.felhasználónév*)

PONT (id, érték)

MEGSZEREZ (*FELHASZNÁLÓ.felhasználónév*, *PONT.id*)

VÁLASZOL (id, *FELHASZNÁLÓ.felhasználónév*, *KÉRDÉS.id*, helyesség)

Normalizálás

A relációs adatbázisséma nincs 1NF-ben, mert van benne többértékű attribútum.
A témakör attribútumot új relációsémába veszem fel, amihez külső kulcsként hozzáveszem az őt tartalmazó Kérdés relációséma kulcsát és hozzáadok egy megnevezést.

FELHASZNÁLÓ (felhasználónév, jelszó, életkor, *JATÉK.azonosító*)
JÁTÉK (azonosító, név, létszám, kérdések_száma, nehézség, belépve)
HOZZÁRENDEL (*JÁTÉK.azonosító*, *KÉRDÉS.id*)
KÉRDÉS (id, szöveg, válasz1, válasz2, válasz3, helyes_válasz, nehézség, *PONT.id*, *FELHASZNÁLÓ.felhasználónév*)
TÉMAKÖR (*KÉRDÉS.id*, megnevezés)
PONT (id, érték)
MEGSZEREZ (*FELHASZNÁLÓ.felhasználónév*, *PONT.id*)
VÁLASZOL (id, *FELHASZNÁLÓ.felhasználónév*, *KÉRDÉS.id*, helyesség)

Így a relációs adatbázisséma 1NF-ben van, mivel már nem található benne sem összetett sem többértékű attribútum.

FELHASZNÁLÓ (felhasználónév, jelszó, életkor, *JATÉK.azonosító*)

2NF: A séma 2NF-ben van, mert a kulcsattribútumok halmaza egyetlen elemből áll, ezért minden másodlagos attribútum teljesen függ a kulcstól.

3NF: A séma 3NF-ben van, mert nem tudunk függést kijelölni.

JÁTÉK (azonosító, név, létszám, kérdések_száma, nehézség, belépve)

2NF: A séma 2NF-ben van, mert a kulcsattribútumok halmaza egyetlen elemből áll, ezért minden másodlagos attribútum teljesen függ a kulcstól.

3NF: A séma 3NF-ben van, mert nem tudunk függést kijelölni.

HOZZÁRENDEL (*JÁTÉK.azonosító*, *KÉRDÉS.id*)

2NF: A séma 2NF-ben van, mert a kulcsattribútumok halmaza egyetlen elemből áll, ezért minden másodlagos attribútum teljesen függ a kulcstól.

3NF: A séma 3NF-ben van, mert nem tudunk függést kijelölni.

KÉRDÉS (id, szöveg, válasz1, válasz2, válasz3, helyes_válasz, nehézség, *PONT.id*, *FELHASZNÁLÓ.felhasználónév*)

2NF: A séma 2NF-ben van, mert a kulcsattribútumok halmaza egyetlen elemből áll, ezért minden másodlagos attribútum teljesen függ a kulcstól.

3NF: A séma 3NF-ben van, mert nem tudunk függést kijelölni.

TÉMAKÖR (*KÉRDÉS.id*, megnevezés)

Redundancia elkerülése érdekében létrehozok egy KÉRDÉS_TÉMAKÖR sémát, melybe felveszem a KÉRDÉS.id és a TÉMAKÖR.id külső kulcsokat. A témaköröket pedig egyetlen séma fogja tartalmazni.

KÉRDÉS_TÉMAKÖR (*KÉRDÉS.id*, *TÉMAKÖR.id*)

TÉMAKÖR (id, témakör_megnevezése)

PONT (id, érték)

2NF: A séma 2NF-ben van, mert a kulcsattribútumok halmaza egyetlen elemből áll, ezért minden másodlagos attribútum teljesen függ a kulcstól.

3NF: A séma 3NF-ben van, mert nem tudunk függést kijelölni.

MEGSZEREZ (*FELHASZNÁLÓ.felhasználónév*, *PONT.id*)

2NF: A séma 2NF-ben van, mert a kulcsattribútumok halmaza egyetlen elemből áll, ezért minden másodlagos attribútum teljesen függ a kulcstól.

3NF: A séma 3NF-ben van, mert nem tudunk függést kijelölni.

VÁLASZOL (id, *FELHASZNÁLÓ.felhasználónév*, *KÉRDÉS.id*, helyesség)

2NF: A séma 2NF-ben van, mert a kulcsattribútumok halmaza egyetlen elemből áll, ezért minden másodlagos attribútum teljesen függ a kulcstól.

3NF: A séma 3NF-ben van, mert nem tudunk függést kijelölni.

FELHASZNÁLÓ (felhasználónév, jelszó, életkor, *JATÉK.azonosító*)

JÁTÉK (azonosító, név, létszám, kérdések_száma, nehézség, belépve)

HOZZÁRENDEL (*JÁTÉK.azonosító*, *KÉRDÉS.id*)

KÉRDÉS (id, szöveg, válasz1, válasz2, válasz3, helyes_válasz, nehézség, *PONT.id*, *FELHASZNÁLÓ.felhasználónév*)

KÉRDÉS_TÉMAKÖR (*KÉRDÉS.id*, *TÉMAKÖR.id*)

TÉMAKÖR (id, témakör_megnevezése)

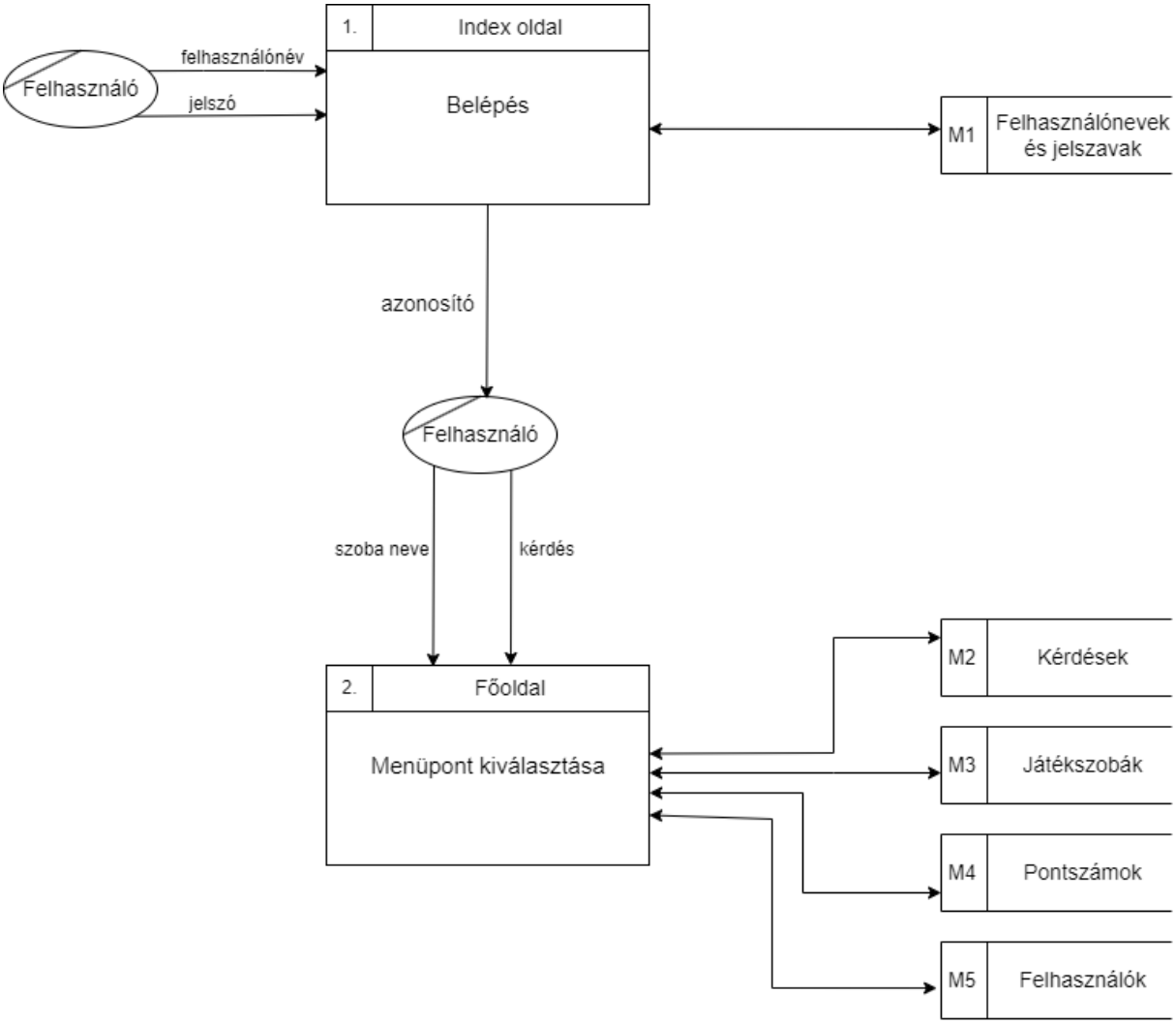
PONT (<u>id</u>, érték)

MEGSZEREZ (<u>*FELHASZNÁLÓ.felhasználónév*</u>, <u>*PONT.id*</u>)

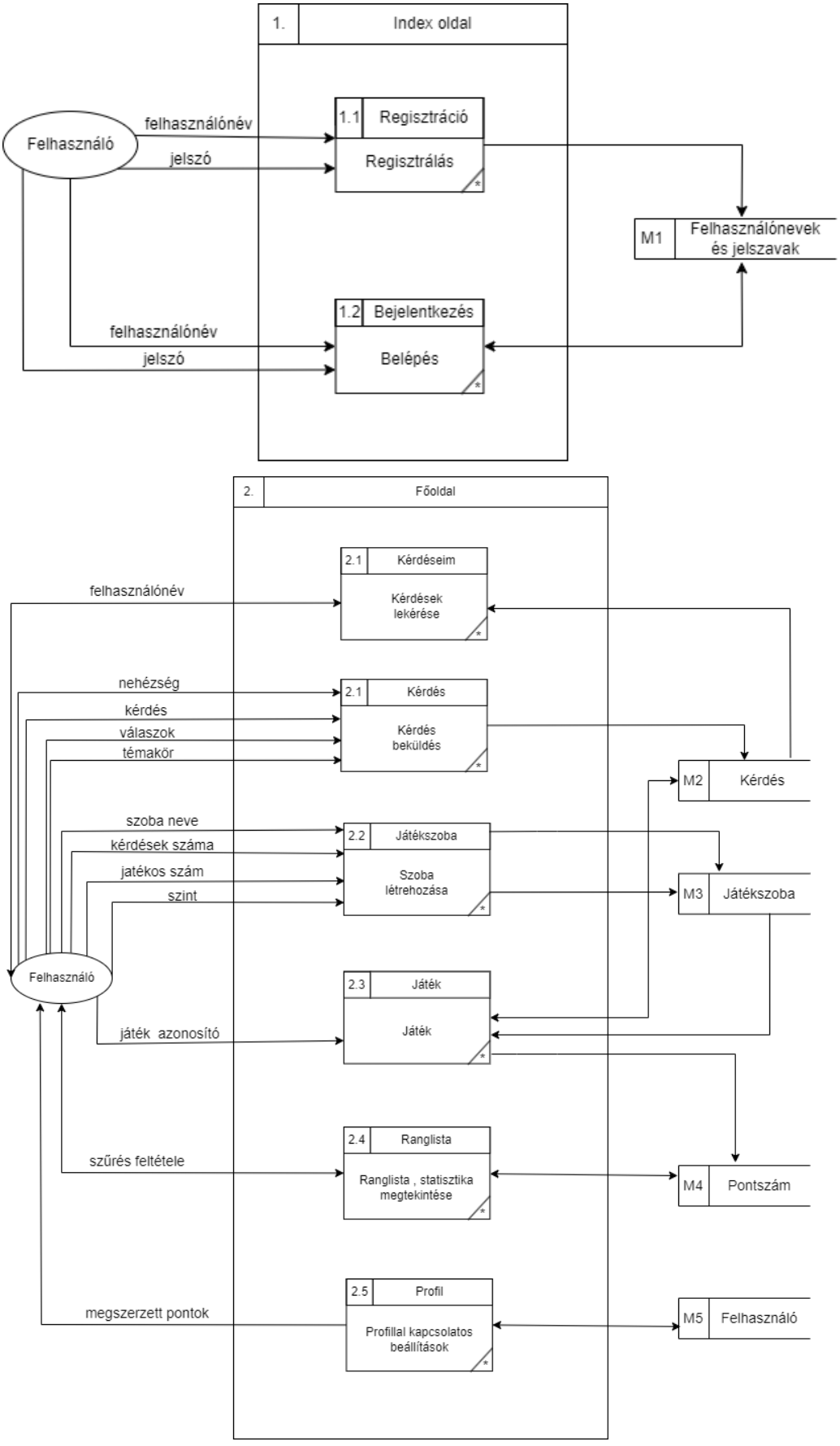
VÁLASZOL (<u>id</u>, <u>*FELHASZNÁLÓ.felhasználónév*</u>, <u>*KÉRDÉS.id*</u>, helyesség)

Adatfolyam-diagramok

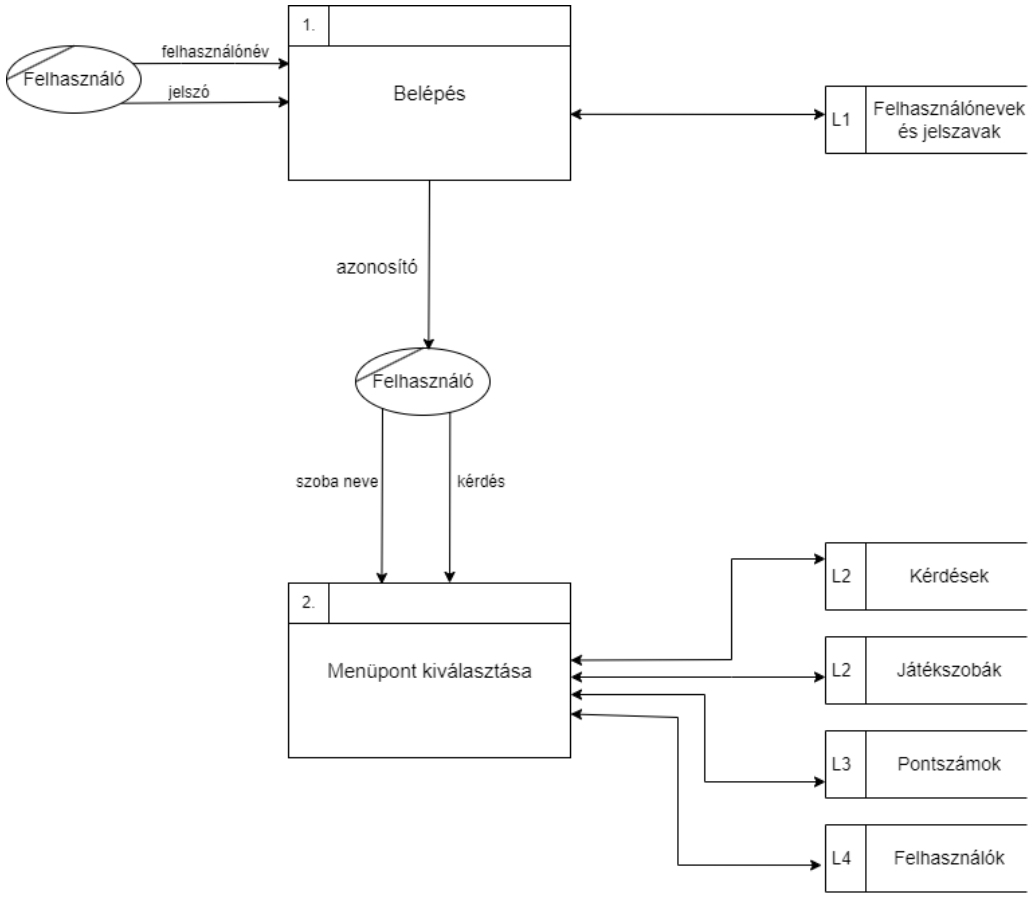
1. szintű fizikai AFD



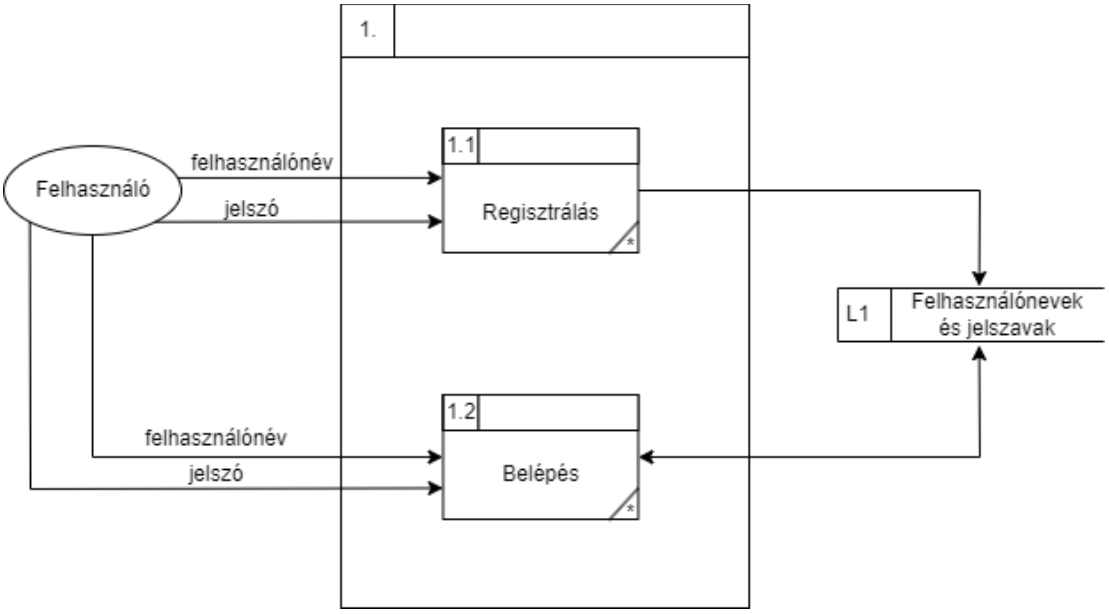
2. szintű fizikai AFD

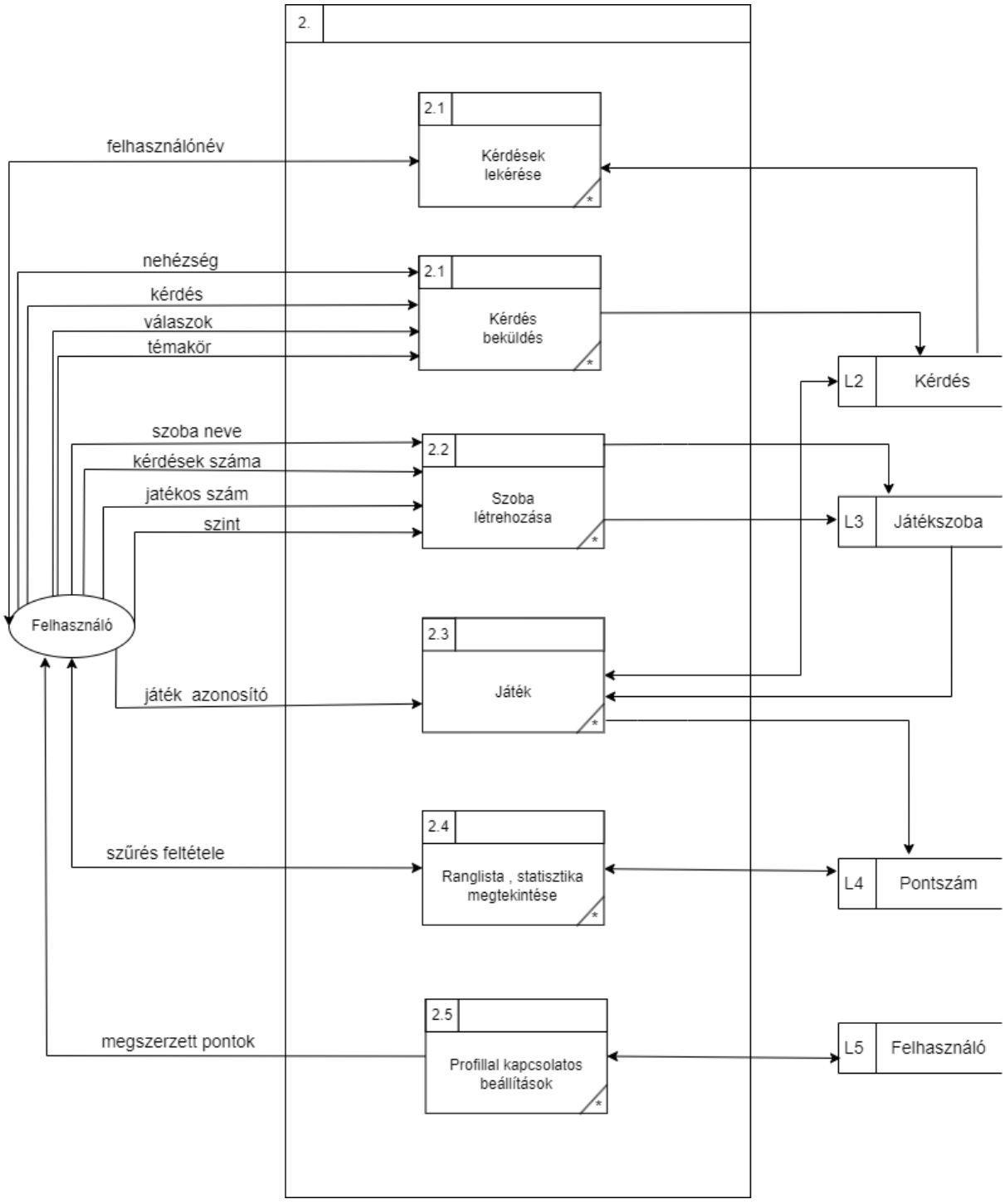


1. szintű logikai AFD

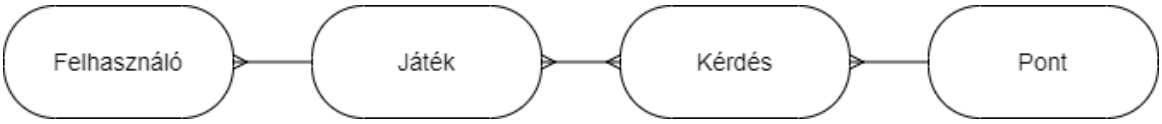


2. szintű logikai AFD





Egyedmodell



Egyed-esemény mátrix

L = létrehozás O = olvasás M = módosítás T = törlés		Események							
		Regisztráció	Bejelentkezés	Kérdés beküldés	Szoba létrehozás	Játék	Statisztika megtekintés	Profil beállítás	Kérdés lekérdezés
Egyed	Felhasználó	L	O	L	L	O	O	T	MT
	Játék					O			O
	Pont					M	O	O	
	Kérdés					O	O		

Szerep-funkció mátrix

		Funkciók						
		Regisztráció	Bejelentkezés	Kérdés beküldés	Szoba létrehozás	Játék	Statisztika megtekintés	Profil beállítás
Szerepek	Felhasználó		X	X	X	X	X	X
	Látogató	X						

A megvalósított összetett lekérdezések

```
$sql = "SELECT *
        FROM (
        SELECT v.felhasznalo_id, v.helyesseg, k.szoveg, k.id
        FROM VALASZOL v
        JOIN KERDES k ON v.kerdes_id = k.id
        WHERE v.felhasznalo_id = :felhasznalonev
        ORDER BY v.id DESC)
        WHERE ROWNUM <= (SELECT kerdesek_szama FROM JATEK WHERE azonosito =
:jatekazon)";
```

helye a programokódban: connection.php 345-352. sora

Ez a lekérdezés az adatbázisból információt kér le a felhasználó által játszott játék eredményeiről. A lekérdezés először egy JOIN műveletet végez a "VALASZOL" és a "KERDES" táblákon, az "id" oszlop alapján, majd a "felhasznalo_id", "helyesseg", "szoveg" és "id" oszlopokat kiválasztja a lekérdezés eredményeként. Az eredményeket a "VALASZOL" táblában tárolt "id" oszlop szerint csökkenő sorrendbe rendezve adja vissza. Illetve még egy WHERE feltételt alkalmaz, hogy csak az első kerdesek_szama számú elemet adja vissza. A

kerdesek_szama változó értéke a "JATEK" táblában tárolt "kerdesek_szama" oszlop értéke lesz, amelynek értéke a jelenlegi játékhoz tartozó kérdések számát jelzi.

```
$sql = "INSERT INTO MEGSZEREZ (felhasznalo_id, pont_id)
        SELECT F.felhasznalonev, P.id
        FROM FELHASZNALO F
        JOIN KERDES K ON F.felhasznalonev = :felhasznalonev
        JOIN PONT P ON P.id = K.nehezseg
        WHERE K.id = :kerdesid";
```

helye a programokódban: connection.php 373-378. sora

A lekérdezés először a "FELHASZNALO" és a "KERDES" táblákat JOIN művelettel kapcsolja össze a "felhasznalonev" és a "KERDES.id" mezők alapján, majd az így kapott halmazt a "PONT" táblával kapcsolja össze az "id" mezőjükön keresztül. A WHERE feltétel szűkíti az eredményhalmazt azon kérdésre, amelynek "id" mezője megegyezik az kerdesid változó értékével. Az eredményhalmaz a "F.felhasznalonev" és a "P.id" oszlopokat tartalmazza.
Az INSERT INTO utasítás a fentiek alapján a "MEGSZEREZ" táblába beszúrja a kapott eredményhalmaz összes sorát. Az első oszlop a "felhasznalo_id" oszlop, amely a "F.felhasznalonev" oszlopban tárolt felhasználó nevét kapja meg, a második oszlop pedig a "pont_id", amely a "P.id" oszlopban tárolt pont azonosítóját kapja meg.

```
CREATE OR REPLACE FORCE EDITIONABLE VIEW "BARNAB"."HELYES_VALASZOK_RANK" ("FELHASZNALO", "TEMAKOR", "HELYES_VALASZOK", "RANK_HELYES_VALASZOK") AS
  SELECT FELHASZNALO.felhasznalonev AS felhasznalo, TEMAKOR.temakor_megnevezese AS temakor,
  SUM(VALASZOL.helyesseg) AS helyes_valaszok,
  RANK() OVER (PARTITION BY TEMAKOR.temakor_megnevezese ORDER BY SUM(VALASZOL.helyesseg) DESC) AS rank_helyes_valaszok
FROM KERDES_TEMAKOR
INNER JOIN TEMAKOR ON KERDES_TEMAKOR.TEMAKOR_id = TEMAKOR.id
INNER JOIN KERDES ON KERDES_TEMAKOR.KERDES_id = KERDES.id
INNER JOIN VALASZOL ON KERDES.id = VALASZOL.KERDES_id
INNER JOIN FELHASZNALO ON VALASZOL.felhasznalo_id = FELHASZNALO.felhasznalonev
GROUP BY FELHASZNALO.felhasznalonev, TEMAKOR.temakor_megnevezese;
```

a fenti lekérdezés egy nézettáblában szerepel, a HELYES_VALASZOK_RANK nézettáblában

Ez a parancs egy nézetet hoz létre vagy cseréli le, ha már létezik, amely a felhasználók helyes válaszainak rangsorát jeleníti meg az egyes témakörökben. A nézet oszlopai a következők:

felhasznalo: a felhasználó neve, akinek a helyes válaszait rangsoroljuk

temakor: a témakör neve, amelyben a felhasználó válaszolt

helyes_valaszok: a felhasználó helyes válaszainak száma az adott témakörben

rank_helyes_valaszok: a felhasználó helyes válaszainak rangsora az adott témakörben

A nézet létrehozása során összekapcsoljuk a KERDES_TEMAKOR, TEMAKOR, KERDES, VALASZOL és FELHASZNALO táblákat, majd ezek alapján csoportosítjuk az eredményt a felhasználók és a témakörök szerint. A helyes válaszokat összeadjuk, majd a RANK() ablakfüggvénnyel rangsoroljuk az eredményt témakörönként a helyes válaszok száma szerinti csökkenő sorrendben.

```
create or replace FUNCTION eletkor_szerinti_ranglista(p_min_age IN NUMBER, p_max_age IN NUMBER) RETURN SYS_REFCURSOR IS
  c_result SYS_REFCURSOR;
BEGIN
  OPEN c_result FOR
    SELECT f.felhasznalonev, f.eletkor, SUM(p.ertek) AS pontszam
    FROM felhasznalo f
    INNER JOIN megszerez m ON f.felhasznalonev = m.felhasznalo_id
    INNER JOIN pont p ON m.pont_id = p.id
    WHERE f.eletkor >= p_min_age AND f.eletkor <= p_max_age
    GROUP BY f.felhasznalonev, f.eletkor;
  RETURN c_result;
END;
```

a fenti lekérdezés az ELETKOR_SZERINTI_RANGLISTA funkcióban szerepel

Ez a PL/SQL függvény egy életkor szerinti ranglistát készít felhasználók és a hozzájuk tartozó pontszámok alapján. A függvény két bemeneti paramétert vár: a minimális és a maximális életkort. Az eredmény egy SYS_REFCURSOR típusú változó, amely egy kurzort (eredményhalmazt) tartalmaz, amely a következő adatokat tartalmazza:

felhasználónév: a felhasználó azonosítója

életkor: a felhasználó életkora

pontszám: a felhasználó által összegyűjtött pontszám az adatbázisban

A függvény csatlakozik a felhasznalo, pont, és megszerez táblákhoz, és kiszámítja az egyes felhasználók összpontszámát a megadott életkorintervallumban. Az eredményt végül a c_result nevű kurzorba tölti be és azt adja vissza a függvény.

```
$sql = "SELECT tk.temakor_megnevezese, COUNT(*) AS kerdesek_szama
        FROM temakor tk
        JOIN kerdes_temakor kt ON tk.id = kt.temakor_id
        JOIN kerdes k ON kt.kerdes_id = k.id
        GROUP BY tk.temakor_megnevezese
        ORDER BY kerdesek_szama ASC";
```

helye a programokódban: connection.php 451-456. sora

A temakor táblát összekapcsolja a kerdes_temakor és a kerdes táblákkal.

A lekérdezés a témakörök nevét (temakor_megnevezese) és a hozzájuk tartozó kérdések számát (kerdesek_szama) számolja.

A GROUP BY segítségével azonos témakörök kérdéseinek száma kerül összegezésre.

Az ORDER BY a kérdések száma szerint növekvő sorrendbe rendezi a témaköröket.

Ez a lekérdezés segítséget nyújt a témakörök és azokhoz tartozó kérdések számának lekérdezésében az adatbázisban.

```
$sql = "SELECT k.szoveg,
        COUNT(*) AS valaszok_szama,
        SUM(CASE WHEN v.helyesseg = 0 THEN 1 ELSE 0 END) AS
helytelen_valaszok_szama
        FROM kerdes k
        JOIN valaszol v ON k.id = v.kerdes_id
        GROUP BY k.szoveg
        HAVING COUNT(*) > 0
        ORDER BY SUM(CASE WHEN v.helyesseg = 0 THEN 1 ELSE 0 END) / COUNT(*)
DESC
        FETCH FIRST 1 ROWS ONLY";
```

helye a programokódban: connection.php 471-479. sora

A lekérdezés visszaadja az összes kérdést a kérdés szövegével, valamint az összes válasz számával és a helytelen válaszok számával együtt. A lekérdezés csak azokat a kérdéseket jeleníti meg, amelyek megválaszolásra kerültek. Az eredmények azon kérdések szerint vannak rendezve, amelyekre a legtöbb helytelen válasz érkezett. A lekérdezés csak az első találatot adja vissza, amely a legtöbb helytelen választ tartalmazó kérdést tartalmazza.

Tehát az eredmény a legnehezebb kérdést adja vissza, azaz azt, amelyre a legtöbb játékos adott helytelen választ.

```
$sql = "SELECT j.azonosito, j.nev, COUNT(*) / j.kerdesek_szama as darabszam
        FROM jatek j
        JOIN hozzarendel h ON j.azonosito = h.jatek_azonosito
        GROUP BY j.azonosito, j.nev, j.kerdesek_szama
        ORDER BY darabszam DESC";
```

helye a programokódban: connection.php 494-498. sora

A fenti SQL lekérdezés egy játékokat és az azokhoz rendelt kérdésszámokat tartalmazó táblázatból számolja ki az összes kérdésre adott válaszok számát a játék kérdésszámához viszonyítva, majd rendezni az eredményeket csökkenő sorrendben a darabszám alapján. A lekérdezés a hozzarendel táblát használja, hogy meghatározza, melyik játékhoz melyik kérdés tartozik. A GROUP BY záradék segítségével csoportosítja a játékokat és azokhoz tartozó kérdésszámokat, majd kiszámítja az összes válaszok számát és a játék kérdésszámához viszonyított arányukat. Végül az eredményeket az arány szerint csökkenő sorrendben rendezve adja vissza, tehát megkapjuk a játékok listáját népszerűség szerint.

```
$sql = "SELECT CASE
        WHEN COUNT(*) = 0 THEN null
        ELSE COUNT(CASE WHEN helyesseg = 1 THEN 1 END) / COUNT(*)
        END AS eredmeny
        FROM VALASZOL
        WHERE felhasznalo_id = :felhasznalonev";
```

helye a programokódban: connection.php 526-531. sora

A kód egy lekérdezést ír le, amely egy felhasználó által adott válaszok helyességi arányát számolja ki. A :felhasznalonev helyére a lekérdezést használó program felhasználónév paramétert fog megadni.

A lekérdezés a VALASZOL táblát használja, és szűkít arra a felhasználóra, akinek a neve megegyezik a megadott felhasználónév paraméterrel. Ezután kiszámítja az összes válaszok számát és a helyes válaszok számát, majd az eredményül kapott helyes válaszok számát elosztja az összes válaszok számával. Ha az összes válaszok száma nulla, akkor az eredmény null lesz, különben az eredmény egy tizedesjegyre kerekített százalékos arány lesz, amely azt mutatja, hogy az adott felhasználó milyen arányban adott helyes válaszokat az összes válaszaihoz képest.

```
create or replace FUNCTION GET_FELHASZNALO_OSSZPONT(felhasznalonev IN VARCHAR2)
RETURN NUMBER IS
  osszpont NUMBER := 0;
BEGIN
  SELECT SUM(PONT.ertek) INTO osszpont FROM PONT
  INNER JOIN MEGSZEREZ ON pont_id = PONT.id
  WHERE MEGSZEREZ.felhasznalo_id = felhasznalonev;
  RETURN osszpont;
END;
```

helye a programokódban: Function GET_FELHASZNALO_OSSZPONT

A fenti program egy olyan PL/SQL függvényt definiál, amely a felhasználó által megszerzett pontok összegét számolja ki. A függvény egy bemeneti paramétert, a felhasználónév stringet várja, majd egy SQL lekérdezést hajt végre a PONT és a MEGSZEREZ táblák közötti kapcsolatok felhasználásával. A lekérdezés eredményeként megszerzi a felhasználó által megszerzett pontok összegét, majd ezt az értéket a függvény visszatérési értékeként adja vissza.