Tárolt eljárások MySQL-ben.

Az országok táblát nézegetjük. Írjunk egy tárolt eljárást, amely visszatér minden adattal az országok táblából.

```
DELIMITER //
CREATE PROCEDURE Mindenadat()
BEGIN
    SELECT * FROM orszagok;
END //
DELIMITER ;

CALL Mindenadat();
```

Itt a **DELIMITER** arra szolgál, hogy *megmondjuk vele a kliensnek*, hol ér véget egy teljes SQL parancs, amikor **több utasítást tartalmazó blokkot** (pl. tárolt eljárást, triggert, függvényt) használunk.

Alapesetben a lezárójel a pontosvessző, viszont egy tárolt eljárás belsejében is használsz `;`-t, ezért kell egy **ideiglenesen más lezáró karakter!**

- A ***`DELIMITER`*** `//` azt mondja: „Mostantól ne a **;** , hanem a // jelzi a parancs végét”
- Ezért a ***`END`*** `//`-nél ér véget a teljes eljárás
- Utána visszaállítjuk a ; -re

MySQL tárolt eljárásnak paramétert a `CREATE PROCEDURE` fejében tudunk átadni. Háromféle paraméter létezik:

- **IN** – bemeneti paraméter (csak be)
- **OUT** – kimeneti paraméter (csak ki)
- **INOUT** – be és ki is

Egyelőre csak az **IN-t** nézzük meg.

```
DELIMITER //
CREATE PROCEDURE Mindenadat(IN p_orsz VARCHAR(40))
BEGIN
    SELECT * FROM orszagok
        WHERE orszag = p_orsz;
END //
DELIMITER ;

CALL Mindenadat("Magyarország");
```

Az általános Függvény így néz ki:

```
DELIMITER //

CREATE FUNCTION fuggveny_neve()
RETURNS INT
DETERMINISTIC
BEGIN
    RETURN 123;
END //

DELIMITER ;
```

Itt a DELIMITER-t már az eljárásnál tárgyaltuk!
A fv-nek lehetnek paraméterei, de **NEM HASZNÁLHAT OUT** paramétert!!!

A RETURNS típusának a megadása kötelező!

A DETERMINISTIC opcionális (nem kötelező). Arra szolgál, hogy biztosítsa, ugyanarra a bemenetre mindig ugyanazt az eredményt adja a fv.

Nézzünk egy példát IF-el:

```
DELIMITER //

CREATE FUNCTION meret(terulet INT)
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
    IF terulet >= 10000000 THEN
        RETURN 'Oroszok';
    ELSEIF terulet >= 1000000 THEN
        RETURN 'Nagy ország';
    ELSEIF terulet > 93036 THEN
        RETURN 'Közepes';
    ELSEIF terulet = 93036 THEN
        RETURN 'Magyarok';
    ELSE
        RETURN 'Kicsi ország';
    END IF;
END //

DELIMITER ;
```

A fv hívása úgy történik, mint egy beépített fv. hívása:

```
SELECT orszag, meret(terulet) FROM orszagok;
```

Az eredmény:

orszag	meret(terulet)
SPANYOLORSZÁG	Közepes
PORTUGÁLIA	Kicsi ország
FRANCIAORSZÁG	Közepes

Nézzünk egy másik adatbázist, ami 3 táblás és barátságos, ez legyen az Euroskills2018 ami megtalálható a 2019. májusi közép szintű érettségi informatika ismeretek részében, onnan letölthető és telepíthető.
Először nézzünk pár eljárást:

Először kérdezzük le az adott szakmakódhoz tartozó versenyzőket az országuk nevével együtt:

```
DELIMITER $$

CREATE PROCEDURE versenyzok_szakma_szerint (IN p_szakma_kod CHAR(2))
BEGIN
    SELECT
        v.id,
        v.nev,
        s.szakmaNev AS szakma,
        o.orszagNev AS orszag,
        v.pont
    FROM versenyzo v
    JOIN szakma s ON v.szakmaId = s.id
    JOIN orszag o ON v.orszagId = o.id
    WHERE v.szakmaId = p_szakma_kod
    ORDER BY v.pont DESC;
END$$

DELIMITER ;
```

Ha értelmezem soronként, akkor a következőt látom:

A *DELIMITER $$* átállítja a sorvége jelet pontosvesszőről $$-ra

A *CREATE PROCEDURE versenyzok_szakma_szerint (IN p_szakma_kod CHAR(2))* azt jelenti, hogy eljárást készítek (*PROCEDURE*), az eljárás neve, amivel később meghívhatom: *versenyzok_szakma_szerint* és van egy csak bemenetre használható paramétere, ami neve a *p_szakma_kod* és ami 2 karakterből állhat maximum. Ezzel fogom megadni a szakma kódját, innen tudja majd, hogy a versenyzők közül melyiket kell kiválogatnia!

A *BEGIN* és az *END$$* között vannak a végrehajtandó utasítások! Az *END* után a *DELIMETER ;* utasítás visszaállítja az elválasztó karaktert pontosvesszőre (itt fontos, hogy legyen szóköz a ; előtt!)!

Az SQL lekérdezést elvileg mindenki érti, egyetlen megjegyzendő, hogy a táblák nevét egyetlen karakterre rövidítettem, hogy kiférjen!

Az eljárás meghívása (mindenki próbálja ki!):
CALL versenyzok_szakma_szerint('04');

Módosítsuk egy adott ID-val rendelkező versenyző pontszámát, ez már tartalmazza az eljárás hívását is. Itt arra kell figyelni, hogy két paramétere van az eljárásnak!

```
DELIMITER $$

CREATE PROCEDURE pontszam_modositas (IN p_id INT,IN p_uj_pont INT)
BEGIN
    UPDATE versenyzo
    SET pont = p_uj_pont
    WHERE id = p_id;
END$$

DELIMITER ;

CALL pontszam_modositas(1,600);
```

Végezetül nézzük meg a versenyzők számát szakmánként:

```
DELIMITER $$

CREATE PROCEDURE versenyzok_szama_szakmankent ()
BEGIN
    SELECT
        s.szakmaNev AS szakma,
        COUNT(v.id) AS letszam
    FROM szakma s
    LEFT JOIN versenyzo v ON s.id = v.szakmaId
    GROUP BY s.id, s.szakmaNev;
END$$

DELIMITER ;

CALL versenyzok_szama_szakmankent ();
```

Itt annyit kell megjegyezni, hogy a szakmák sorrendje az Id szerint lesz rendezve, hiszen először az van a csoportosításban.

Nézzünk függvényeket, mindenféle vezérlési szerkezettel (If, Case, While)

Először minősítsük a versenyzőket a pontszámuk szerint:

```
DELIMITER $$

CREATE FUNCTION pontszam_minosites(p_pont INT)
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
    DECLARE v_eredmeny VARCHAR(20);

    IF p_pont >= 700 THEN
        SET v_eredmeny = 'Kiváló';
    ELSEIF p_pont >= 650 THEN
        SET v_eredmeny = 'Jó';
    ELSEIF p_pont >= 600 THEN
        SET v_eredmeny = 'Közepes';
    ELSE
        SET v_eredmeny = 'Gyenge';
    END IF;

    RETURN v_eredmeny;
END$$

DELIMITER ;

SELECT nev, pont, pontszam_minosites(pont) FROM versenyzo;
```

Itt a fv-t egy paraméterrel hívjuk meg, ami INT típusú, ebben van a pontszám. Utána megadtuk a visszatérő érték típusát, ami VARCHAR(20). A DETERMINISTIC kulcsszó azt jelzi, hogy a függvény ugyanazon bemeneti paraméterek esetén mindig azonos eredményt ad vissza. (nem tartalmazhat időt, véletlenszámot stb.) A BEGIN után deklarálok egy változót, ezt adom majd vissza, ebben lesz a minősítés szövegesen. Az END$$ Előtt visszaadom a hívó résznek az eredményt!

Kapjon a versenyző 50 pontonként 1 + pontot (ciklussal oldjuk meg, addig vonogatunk ki 50-et a pontjaiból, amíg nem kapunk 50-nél kisebb számot)

```
CREATE FUNCTION bonusz_pont(p_pont INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE v_bonusz INT DEFAULT 0;
    DECLARE v_maradek INT;

    SET v_maradek = p_pont;

    WHILE v_maradek >= 50 DO
        SET v_bonusz = v_bonusz + 1;
        SET v_maradek = v_maradek - 50;
    END WHILE;

    RETURN v_bonusz;
END$$

DELIMITER ;
SELECT nev, pont, bonusz_pont(pont) FROM versenyzo;
```

Ha pedig létrehoztuk az előző két fv-t, akkor csináljunk egy végső értékelést:

```
DELIMITER $$

CREATE FUNCTION vegso_ertekeles(p_pont INT)
RETURNS VARCHAR(50)
DETERMINISTIC
BEGIN
    DECLARE v_bonusz INT;
    DECLARE v_minosites VARCHAR(20);

    SET v_bonusz = bonusz_pont(p_pont);
    SET v_minosites = pontszam_minosites(p_pont);

    CASE
        WHEN v_bonusz >= 14 THEN
            RETURN CONCAT(v_minosites, ' + kiemelkedő bónusz');
        WHEN v_bonusz >= 12 THEN
            RETURN CONCAT(v_minosites, ' + magas bónusz');
        ELSE
            RETURN CONCAT(v_minosites, ' + normál bónusz');
    END CASE;
END$$

DELIMITER ;

SELECT nev, pont, vegso_ertekeles(pont) FROM versenyzo;
```