26.Példa

Készítsünk olyan listát, amely a Dolgozó táblából kilistázza azon dolgozók adatait, akik minden Egerben vagy Szegeden született dolgozónál többet keresnek!

Az alábbi parancsot használhatjuk:

```
SELECT * FROM Dolgozó WHERE 'A dolgozó fizetése' >
ALL (SELECT 'A dolgozó fizetése' FROM Dolgozó WHERE
'A dolgozó születési helye' LIKE "Eger" OR 'A dolgozó
születési helye' LIKE "Szeged")
```

A dolgozó törzsszáma	A dolgozó neve	A dolgozó születési helye	A dolgozó születési ideje	A dolgozó fizetése
T456734	Nagy József	Budapest	1972. 01. 30.	150000
T768545	Vári Ödön	Budapest	1958. 07. 12.	210000

Az adatbázis módosítása SQL segítségével

Relációsémák definiálása

Definiálhatjuk a tábla nevét, megadhatjuk az attribútumait, azok típusát és méretét. A legfontosabb adattípusok, amelyeket az SQL szabvány definiál:

Egész számok. Ezek megadásánál a **SHORTINT**, **INT** vagy **INTEGER** kulcsszavakat használhatjuk.

Valós számok, lebegőpontos tárolással. Ezeknél is különböző méretben tárolt számokat adhatunk meg.

A **FLOAT** és a **REAL** a hagyományos programozás nyelvekből is ismert normál lebegőpontos számot jelenti. A **DOUBLE PRECISION** duplapontos számot jelent, míg speciálisan az SQL-ben használhatjuk a **DECIMAL(<számjegy>,<tizedesjegy>)** formát is, ahol explicit módon megadhatjuk, hogy a szám hány számjegyből állhat, illetve hány tizedesjegyet tartalmazhat.

Fix vagy **változó hosszúságú karaktersorozatok.**

Megadásuk a **CHAR(<hossz>)**, illetve a **VARCHAR(<hossz>)** paranccsal történik. A CHAR segítségével olyan attribútumot definiálhatunk, amely pontosan a megadott hosszúságú karaktersorozatként fogja tárolni az adatokat, míg a VARCHAR-ral megadott attribútumoknál csak az aktuális számú karakter kerül tárolásra. Látszólag tehát a VARCHAR szolgáltatja a jobb megoldást, azonban az SQL elég rugalmas ahhoz, hogy a CHAR kulcsszóval definiált attribútumoknál is jól kezeli a rövidebb karaktersorozatokat is, úgy hogy azokat szóközzel kitölti, amit az összehasonlításnál nem vesz figyelembe.
Dátum és idő. Ezeket a **DATE** és **TIME** kulcsszavakkal lehet megadni.

A táblák létrehozására irányuló parancs szintaxisát.

CREATE TABLE <táblanév> { <attribútumdefiníció> [,<attribútumdefiníció>]... }

Az **<attribútumdefiníció>** paraméterben adjuk meg az egyes attribútumok nevét és típusát, a következő módon:

<név> <típus> [DEFAULT <érték>]

A <név> jelenti az attribútum nevét, míg a <típus> adja meg a típust és a méretet, a fentiekben ismertetett módon. A DEFAULT paranccsal alapértelmezett értékeket adhatunk meg az egyes attribútumoknak, ennek használata nem kötelező.

Lehetőség van arra is, hogy már meglévő táblához **új oszlopot adjunk.** Ez a következőképpen történhet:

ALTER TABLE <táblanév> ADD <attribútumdefiníció>

Hasonlóan történhet egy oszlop eltávolítása a táblából. Ebben az esetben az attribútum típusát nem szükséges megadni, elegendő a neve.

ALTER TABLE <táblanév> DROP <attribútumnév>

Lehetőségünk van tábla törlésére is a következő módon:

DROP <táblanév>

Tudjuk, hogy egy tábla definiálásakor a tábla nevén és az attribútumokon túlmenően még egyéb információkat is meg kell adnunk.
Ilyenek **a kulcsok, az attribútum értékekre vonatkozó korlátozások.**
Az SQL szabvány erre is szolgál megoldással. Először a kulcsok, illetve az egyedi értékekkel bíró attribútumok megadásának módját ismertetjük. Az SQL-

ben alapvetően az elsődleges kulcs megadására van lehetőségünk, ahogy azt a legtöbb, a gyakorlatban használatos ABKR megköveteli tőlünk.
Ha elsődleges kulcsot szeretnénk az SQL nyelv segítségével definiálni, a tábla létrehozását kibővíthetjük speciális parancsokkal.
Ez a következőképpen néz ki:

```
CREATE TABLE <táblanév> { <attribútumdefiníció>
[UNIQUE] [,<attribútumdefiníció> [UNIQUE]]... [,PRIMARY
KEY (<kulcsattribútum> [,<kulcsattribútum>]...)|UNIQUE
(<kulcsattribútum>) ]}
```

A **<kulcsattribútum>** paraméterben kell megadni annak az attribútumnak a nevét, amely a kulcsot alkotja, vagy annak egy részét képezi.
Amennyiben csak egy attribútum tartozik a kulcshoz, akkor használhatjuk mind a **PRIMARY KEY,** mind a **UNIQUE** parancsokat. Több attribútumból álló kulcsot csak a PRIMARY KEY kulcsszóval definiálhatunk. A UNIQUE kulcsszó segítségével minden egyes attribútumnál megadhatjuk, hogy az adott attribútum csak egyedi értékeket vehet fel.
Hasonlóan az elsődleges kulcs megadásához, az SQL lehetőséget biztosít **idegen kulcsok definiálására** is.
Ennek módja az alábbi:

```
CREATE TABLE <táblanév> { <attribútumdefiníció>
[REFERENCES <táblanév> (<attribútumnév>)]
[,<attribútumdefiníció> [REFERENCES <táblanév>
(<attribútumnév>)]]... [,FOREIGN KEY (<kulcsattribútum>
[,<kulcsattribútum>]...) <táblanév>
(<kulcsattribútum>[,<kulcsattribútum>]...) ]}
```

Láthatjuk, hogy az idegen kulcs megadása teljesen hasonló az elsődleges kulcshoz.
A különbség mindössze annyi, hogy idegen kulcsnál mindig meg kell adni, hogy az attribútum melyik másik tábla melyik kulcsmezőjéhez kapcsolódik.
Amennyiben az idegen kulcs egy attribútumból áll, használhatjuk a **REFERENCES** kulcsszót,
ha azonban az idegen kulcs összetett, akkor a **FOREIGN KEY** kulcsszóval kell definiálnunk.

Említettük, hogy az ABKR-nek gondoskodnia kell az úgynevezett hivatkozási épség fenntartásáról. Ez azt jelenti, hogy ha egy idegen kulcsban hivatkozunk egy másik tábla egy kulcsértékére, akkor a megadott értékű előfordulásnak létezni kell.

Amennyiben olyan módosító, vagy törlő műveletet hajtunk végre, ami ezt a szabályt megsérti, akkor az ABKR-nek ezt valahogyan kezelni kell.
Egyik lehetőség, hogy **a műveletet nem engedi végrehajtani, a másik, hogy megengedi, de a hivatkozási épség fenntartása érdekében automatikusan korrigálja az adatbázist.** A korrigálás kétféleképpen történhet.
Az egyik az, hogy ha egy hivatkozott sort törlünk vagy módosítunk, akkor a rá hivatkozó előfordulások is törlődnek, vagy módosulnak a másik táblában. A másik lehetőség az, hogy a helytelen hivatkozásokat egy speciális értékkel korrigálja az ABKR, amelyet nevezhetünk NULL értéknek.
Amennyiben azt szeretnénk, hogy valamely korrigáló eljárás lépjen életbe a hivatkozási épség megsérülésekor, akkor ezt a tábla definiálásakor az SQL parancsban külön megadhatjuk.
Mivel a hivatkozási épség mindig idegen kulcsokra vonatkozik, ezért ezt csak ezeknél lehet használni. A szintaxis definíciójánál a REFERENCES kulcsszóval megadott részt egészítjük ki.

```
REFERENCES <táblanév> (<attribútumnév>) [[ON DELETE
SET NULL|CASCADE] [ON UPDATE SET NULL|CASCADE]]
```

Az **ON DELETE** részben azt adhatjuk meg, hogy a törlés során bekövetkezett hivatkozás épség sérülését hogyan kezelje a rendszer,
míg
az **ON UPDATE** részben a módosításkor bekövetkezőt. Mindkét esetben az ismertetett két lehetőség közül választhatunk, vagyis a NULL érték beállítása, (SET NULL), illetve korrigálás a hivatkozó táblában is (CASCADE).

A tábla definiálásakor megadható megszorítások következő nagy csoportját az attribútumok értékére vonatkozó korlátozások alkotják. Ennek legegyszerűbb fajtája az, amikor a megszorítás az **egyes attribútumok értékeire** vonatkozik.
Ezt szintén a tábla definiálásakor adhatjuk meg, az attribútum leírásakor. Ennek formája a következő:

```
<attribútumdefiníció> NOT NULL|CHECK (<feltétel>)
```

A NOT NULL opció azt jelenti, hogy az adott attribútum nem vehet fel NULL értéket.
A CHECK kulcsszó után tetszőleges feltételt adhatunk. Az erre vonatkozó szabályok megegyeznek a WHERE SQL parancs után használt feltétel megadásánál alkalmazottakkal.
A feltétel ellenőrzése sor beszúrásakor, vagy az attribútum módosításakor történik.

Ennél általánosabb megszorítások is megfogalmazhatók. Lehetnek olyanok, amelyek **sorokra vonatkoznak,** és lehetnek olyan globális önálló megszorítások, amelyek a **teljes adatbázisra** vonatkoznak. A sorokra vonatkozó megszorítások ellenőrzése a sorban történő bármilyen módosításkor megtörténik. A teljesen általános megszorítások ellenőrzése minden olyan módosításkor bekövetkezik ,aminek az adott feltételre hatása lehet. A sorra vonatkozó feltételek megadása szintén a CHECK paranccsal történik, ezt a tábla definiálásának a végén kell megadni. Formája az alábbi:

```
CREATE TABLE <táblanév> { <attribútumdefiníció>
[,<attribútumdefiníció>]... [CHECK <feltétel>]}
```

A globális megszorítások definiálása külön kulcsszóval történik, melynek formája a következő:

```
CREATE ASSERTION <név> CHECK <feltétel>
```

27.Példa

Adjuk meg azt az SQL parancsot , amely létrehozza a Dolgozó táblát!

```
CREATE TABLE Dolgozó {'A dolgozó törzsszáma' CHAR(7),
'A dolgozó neve' VARCHAR(50), 'A dolgozó születési
helye' VARCHAR(30) 'A dolgozó születési ideje' DATE,
'A dolgozó fizetése' DECIMAL(7,0) }
```

A keletkezett tábla az alábbi:

A dolgozó törzsszáma	A dolgozó neve	A dolgozó születési helye	A dolgozó születési ideje	A dolgozó fizetése

28.Példa

Adjuk meg azt az SQL parancsot, amely a Dolgozó táblát létrehozza, és elsődleges kulcsnak a A dolgozó törzsszáma mezőt definiálja!

```
CREATE TABLE Dolgozó {'A dolgozó törzsszáma' CHAR(7)
PRIMARY KEY, 'A dolgozó neve' VARCHAR(50), 'A dolgozó
születési helye' VARCHAR(30) 'A dolgozó születési
ideje' DATE, 'A dolgozó fizetése' DECIMAL(7,0) }
```

Egy másik lehetséges megoldás:

```
CREATE TABLE Dolgozó {'A dolgozó törzsszáma' CHAR(7),
'A dolgozó neve' VARCHAR(50), 'A dolgozó születési
helye' VARCHAR(30) 'A dolgozó születési ideje' DATE,
'A dolgozó fizetése' DECIMAL(7,0) PRIMARY KEY ('A
dolgozó törzsszáma')}
```

29.Példa

Tegyük fel, hogy a Kifizetés táblában található 'A dolgozó törzsszáma' nevű mező idegen kulcs, amely a Dolgozó táblával való kapcsolatot valósítja meg. Adjuk meg azt az SQL parancsot, amely a fenti Kifizetés táblát létrehozza, úgy hogy a hivatkozási épség sérülésekor az ABKR azt automatikusan frissítéssel korrigálja!

```
CREATE TABLE Kifizetés {'A kifizetés dátuma' DATE, 'A
kifizetett bér' DECIMAL(7,0), 'A levont adóelőleg'
DECIMAL(7,0) 'A dolgozó törzsszáma' CHAR(7)
REFERENCES Dolgozó ('A dolgozó törzsszáma') ON UPDATE
CASCADE ON DELETE CASCADE}
```

Egy másik lehetséges megoldás:

```
CREATE TABLE Kifizetés {'A kifizetés dátuma' DATE, 'A
kifizetett bér' DECIMAL(7,0), 'A levont adóelőleg'
DECIMAL(7,0) 'A dolgozó törzsszáma' CHAR(7), FOREIGN
KEY ('A dolgozó törzsszáma') Dolgozó ('A dolgozó
törzsszáma') ON UPDATE CASCADE ON DELETE CASCADE}
```

A keletkezett tábla a következő:

A kifizetés dátuma	A kifizetett bér	A levont adóelőleg	A dolgozó törzsszáma

30.Példa

Adjuk meg azt az SQL parancsot, amely a Dolgozó táblát úgy definiálja, hogy a 'A dolgozó fizetése' mező esetén mindig ellenőrzésre kerüljön, hogy az éppen megadott érték eléri-e egy minimális bér összegét, mondjuk 22500 Ft-ot!

```
CREATE TABLE Dolgozó {'A dolgozó törzsszáma' CHAR(7),
'A dolgozó neve' VARCHAR(50), 'A dolgozó születési
helye' VARCHAR(30) 'A dolgozó születési ideje' DATE,
'A dolgozó fizetése' DECIMAL(7,0) CHECK 'A dolgozó
fizetése' >22500}
```

31.Példa

Adjuk meg azt az SQL parancsot, amely egy olyan globális megszorítást definiál, amely ellenőrzi, hogy a Dolgozó táblában az összes fizetés együttesen ne haladja meg a 10.000.000 Ft-ot!

```
CREATE ASSERTION Összefizetés CHECK (10000000 >=
(SELECT SUM('A dolgozó fizetése') FROM Dolgozó))
```

32.Példa

Adjuk meg azt az SQL parancsot , amely hozzáadja a Dolgozó táblához a dolgozó lakcímét!

```
ALTER TABLE Dolgozó ADD 'A dolgozó lakcíme'
VARCHAR(50)
```

A keletkezett tábla az alábbi:

A dolgozó törzsszáma	A dolgozó neve	A dolgozó születési helye	A dolgozó születési ideje	A dolgozó fizetése	A dolgozó lakcíme

33.Példa

Adjuk meg azt az SQL parancsot, amely a Dolgozó táblából törli A dolgozó születési helye attribútumot!

```
ALTER TABLE Dolgozó DROP 'A dolgozó születési helye'
```

A keletkezett tábla az alábbi:

A dolgozó törzsszáma	A dolgozó neve	A dolgozó születési ideje	A dolgozó fizetése	A dolgozó lakcíme

34.Példa

Adjuk meg azt az SQL parancsot, amely a Dolgozó táblát törli!

```
DROP Dolgozó
```

Változtatások az adatbázisban

Az SQL nyelv lehetőséget biztosít arra is, hogy meglévő adattáblák adatait karbantartsuk. Ennek keretében lehetőségünk van új sorok beszúrására, törlésére, létező sorok egyes adatainak megváltoztatására. Most áttekintjük az erre vonatkozó parancsokat.

Új sorok beszúrása a következő paranccsal történhet:

```
INSERT INTO <relációséma> VALUES (<értéklista>)
```

A **<relációséma>** paraméterben meg kell adnunk a reláció nevét. majd zárójelben azokat az attribútum neveket, amelyekhez az <értéklista> paraméterben megadott egymástól vesszővel elválasztott értékeket rendeljük. **Amennyiben csak a reláció nevét adjuk meg, azt alapértelmezésben úgy tekinti a rendszer, mintha az összes attribútumot felsoroltuk volna.**
A két lista elemeinek számban, sorrendben és típusban meg kell egyezniük. A végrehajtás során a megfelelő értékek hozzárendelődnek a megfelelő attribútumhoz, és az így keletkezett sor bekerül az adattáblába. Amennyiben egy létező attribútum név nem szerepel a listában, akkor ahhoz az alapértelmezés szerinti értéke rendelődik, amennyiben ilyen van, vagy pedig nem kap értéket, azaz például a NULL érték kerül bele.

A beszúrás művelet ellentettje a törlés. Ennél a specialitást az jelenti, hogy meg kell adnunk azt a feltételt, amelynek eleget tevő sorokat törölni szeretnénk. Ennek formája az alábbi:

```
DELETE FROM <reláció> WHERE <feltétel>
```

A <feltétel> paraméterben a már jól ismert módon adhatjuk meg azt a feltételt, amely alapján a törlés történik.

A módosítás szintaxisa a következő:

```
UPDATE <reláció> SET <értékadások> WHERE <feltétel>
```

Az értékadások vesszővel vannak egymástól elválasztva. Mindegyik értékadás egy attribútum névből, egyenlőségjelből és értékből áll. Az értékek módosítása azon sorokra történik meg, amelyek a megadott feltételnek eleget tesznek.

35.Példa

Tekintsük az alábbi Dolgozó táblát:

A dolgozó törzsszáma	A dolgozó neve	A dolgozó születési helye	A dolgozó születési ideje	A dolgozó fizetése
T234578	Kiss István	Eger	1968. 12. 11.	120000
T343234	Kiss Timót	Eger	1970. 02. 28.	105000
T456734	Nagy József	Budapest	1972. 01. 30.	150000

Adjuk meg azt az SQL parancsot, amely egy teljes sort szúr be a táblába!

```
INSERT INTO Dolgozó VALUES ("T768545","Vári Ödön",
"Budapest", 1958.07.12. 210000)
```

A módosult tábla a következő:

A dolgozó törzsszáma	A dolgozó neve	A dolgozó születési helye	A dolgozó születési ideje	A dolgozó fizetése
T234578	Kiss István	Eger	1968. 12. 11.	120000
T343234	Kiss Timót	Eger	1970. 02. 28.	105000
T456734	Nagy József	Budapest	1972. 01. 30.	150000
T768545	Vári Ödön	Budapest	1958. 07. 12.	210000

36.Példa

Az előző példában szereplő Dolgozó táblára vonatkozóan adjuk meg azt az SQL parancsot, amely egy olyan dolgozót szúr be a táblába, akinek egyelőre csak a neve és a törzsszáma áll rendelkezésre!

```
INSERT INTO Dolgozó('A dolgozó törzsszáma', 'A dolgozó neve' VALUES ("T429877","Kovács János")
```

Ezután az alábbi táblát kapjuk:

A dolgozó törzsszáma	A dolgozó neve	A dolgozó születési helye	A dolgozó születési ideje	A dolgozó fizetése
T234578	Kiss István	Eger	1968. 12. 11.	120000
T343234	Kiss Timót	Eger	1970. 02. 28.	105000
T456734	Nagy József	Budapest	1972. 01. 30.	150000
T768545	Vári Ödön	Budapest	1958. 07. 12.	210000
T429877	Kovács János			

37.Példa

Adjuk meg azt az SQL parancsot, amely a Dolgozó táblából törli az összes Budapesten született dolgozót!

```
DELETE FROM Dolgozó WHERE 'A dolgozó születési helye' = "Budapest"
```

Az így kapott tábla:

A dolgozó törzsszáma	A dolgozó neve	A dolgozó születési helye	A dolgozó születési ideje	A dolgozó fizetése
T234578	Kiss István	Eger	1968. 12. 11.	120000
T343234	Kiss Timót	Eger	1970. 02. 28.	105000
T429877	Kovács János			