Várhelyi András

Adatbázis készítésének alapjai

NSZFI
NEMZETI SZAKKÉPZÉSI ÉS FELNŐTTKÉPZÉSI INTÉZET

A követelménymodul megnevezése:

Dokumentumkészítés és -kezelés az irodában

A követelménymodul száma: 1618-06 A tartalomelem azonosító száma és célcsoportja: SzT-006-50

Új Magyarország

ADATBÁZIS KÉSZÍTÉS ALAPJAI

ESETFELVETÉS - MUNKAHELYZET

Ön egy fuvarozó cégnél azt a feladatot kapja, hogy hozzon létre adatbázis a cég üzleti partnerinek és tehergépkocsijainak nyilvántartására.

SZAKMAI INFORMÁCIÓTARTALOM

ADATBÁZIS

Az adatbázis tágabb értelemben egy olyan adathalmaz, amelynek elemei egy meghatározott tulajdonságuk alapján összetartozónak tekinthetők. Az adatbázis-kezelőknek meg kell oldani ezen adatok rendezését, a köztük lévő kapcsolat nyilvántartását, az adatokhoz való hozzáférés szabályozását, az adatok védelmét, az integritás megőrzését, az adatok módosíthatóságát, lekérdezését, különféle szempontok szerinti kigyűjtését, válogatását és egyéb statisztikai funkciókat is.

Egy meghatározott témakörrel kapcsolatos információk lehetnek például a vevői megrendelések, számlázási vagy készlet-nyilvántartási adatok, stb.

1. Az adatbázishoz kapcsolódó fogalmak

Az **adatbázis** az adatok és a köztük lévő összefüggések rendszere, amelyet egymás mellett tárolunk. Nagyon fontos, hogy az adatbázisunk szerkezetét jól megtervezzük, mert a későbbiekben csak így tudunk hatékonyan dolgozni vele.

A **tábla** a logikailag összetartozó adatokat foglalja össze. A tábla oszlopokból és sorokból áll, melyeket mezőknek, illetve rekordoknak nevezünk.

A **rekord** az adatbázis egy sora. Egy rekordban tároljuk az egymással összefüggő adatokat.

A **mező** az adatbázis egy oszlopa, amelyben az egyedek tulajdonságértékeit tároljuk.

Az **elemi adatok** a táblázat celláiban szereplő értékek, amelyek az egyed konkrét tulajdonságai.

Az **egyed** az, amit le akarunk írni, amelynek az adatait tároljuk és gyűjtjük az adatbázisban. Az egyedet idegen szóval entitásnak nevezzük. Egyednek tekinthetünk például egy személyt.

Az **attribútum** vagyis tulajdonság az egyed valamely jellemzője. Az egyed az attribútumok összességével jellemezhető. Egy személy egy jellemzője lehet például a neve.

Az egyedre vonatkozóan megadott tulajdonságok összességét **egyedtípusnak** nevezzük. Egy személy leírható például a nevével, életkorával, testmagasságával, a szeme és haja színével együttesen.

Az egyedre vonatkozóan megadott konkrét tulajdonságokat **egyed-előfordulásnak** nevezzük. Egy egyed-előfordulás például Kovács Ödön, aki 29 éves, 183 cm magas, kék szemű, barna hajú.

Elsődleges kulcs: a táblázat rekordjainak egyértelmű azonosítója, értéke egyedi.

Idegen kulcs: olyan azonosító amelynek segítségével egy másik táblázat elsődleges kulcsára hivatkozhatunk.

ACCESS

A Microsoft ACCESS egy relációs adatbázis-kezelő rendszer, mely a Windows alatt fut. A különböző egyedekről táblákban tároljuk az adatokat.

1. Az Access indítása

A Start menü - Programok - Microsoft Access ikonra kattintással, vagy az Intézőben valamelyik, az Access-hez társított adatbázis-fájlra kattintva elindul a program.

Ha az Access-t a Start menüből vagy az Office Irányítópultról indítjuk, a kezdő párbeszédpanelen kiválaszthatjuk, hogy melyik adatbázist szeretnénk megnyitni. A program felkínálja azokat a fájlokat, amelyekkel legutóbb dolgoztunk, de kikereshetjük a kívánt fájlt a fájlrendszerben, vagy létrehozhatunk egy új, üres adatbázist is.

1. ábra. Access[1]

2. Az adatbázis tervezés főbb lépései

- Első lépés: az adatbázis céljának meghatározása.
- Második lépés: a szükséges táblák meghatározása.
- Harmadik lépés: a szükséges mezők meghatározása.
- Negyedik lépés: kapcsolatok meghatározása.
- Ötödik lépés: a terv finomítása. Elemezzük a terveket, hogy megtaláljuk az esetleges hibákat. (Tesztelés, normál formák, elsődleges kulcsok, **kapcsolattípusok**: egy a többhöz, több a többhöz, egy az egyhez.)

Az adatbázisban létrehozhatunk:

- táblákat (az adatok tárolásához),
- lekérdezéseket (az adatok logikai megjelenítésére, szűrésére),
- űrlapokat (a vizuális megjelenítésre, felhasználásra),
- jelentéseket (a nyomtatáshoz),
- makrókat, modulokat (az űrlapok eseményeihez).

Ezeket megtehetjük manuálisan és varázslóval is.

3. Tábla felépítése

A táblákban egy-egy információcsoport adatait tárolhatjuk. Ha minden információcsoporthoz külön táblát használunk, akkor elérhetjük, hogy minden adatot csak egyszer tároljunk. Ha így építjük fel adatbázisunkat, akkor hatékonyabb lesz, és az adatbeviteli hibák esélyét is lecsökkentjük.

[1] www.microsoft.com

2. ábra. Egy tábla az Accessben

a. Tábla nézetei

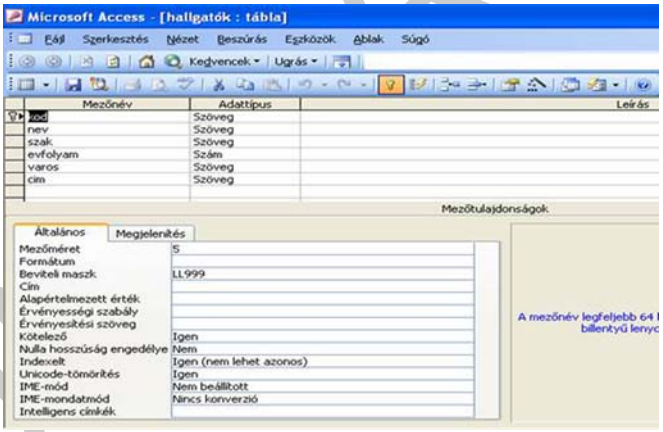
Egy táblát 2 féle nézetben lehet használni, ill. megtekinteni:

Tervező nézet

Ebben a nézetben van lehetőségünk kialakítani és módosítani a tábla szerkezetét, vagyis azt, hogy milyen mezőket tartalmazzon, és azok a mezők milyen tulajdonságokkal rendelkezzenek.

Adatlap nézet

Ebben a nézetben tudjuk az adatainkat felvinni, módosítani, ill. törölni a táblából.



3. ábra. Tábla tervező nézetben

b. Kulcsok és indexelés

Az indexeléssel meggyorsítjuk a táblákban a keresést és a rendezést, a lekérdezéseket és a csoportosításokat, a frissítés ugyanakkor az indexelés időigényével megnő.

Az indexeléssel egy kiválasztott mező szerint logikailag rendezzük az adatbázist, de valójában minden rekord a helyén marad.

A táblák elsődleges kulcsa automatikusan indexelt.

4. Tábla létrehozása

Mielőtt létrehoznánk új táblánkat meg kell nyitni az adatbázist, melyben a táblát létre kívánjuk hozni. Ha ez megtörtént, az Objektumok panelen kattintsunk a Táblák objektumcsoportra. Ekkor 3 lehetőség közül választhatunk. Lehetőségünk van már létrehozott tábla megnyitására (Megnyitás) ill. tervezésére (Tervezés), és létrehozhatunk új táblát (Új).

A táblákat különböző módon hozhatunk létre:

- Tábla Varázsló
- Adatlap nézet
- Tervező nézet
- Importálás

Ha az új táblázat létrehozásához a Tervező nézet pontot választottuk, Tervező nézetbe lépünk, melyben kialakíthatjuk táblánk szerkezetét, felsorolhatjuk, hogy milyen mezőkből álljon a tábla. A mezők neve mellett meg kell adnunk még azt, hogy milyen típusú értéket tárolunk benne, és adhatunk leírást is hozzájuk.

A mezők típusai a következők lehetnek:

Szöveg – Az ilyen típusú mezőben szöveget (vagyis tetszőleges karaktereket) tárolhatunk, de csak legfeljebb annyit, amennyit a Mezőméret tulajdonságban beállítottunk. A mező értékétől függetlenül minden új rekord esetén a Mezőméretben megadott számú byte-tal nő a tábla és így az adatbázis mérete is.

Feljegyzés – Ebben a mezőben szintén szöveget tárolhatunk. A különbség az előző (Szöveg) mezővel kapcsolatban, hogy ebben a mezőben tetszőleges hosszú szöveget tárolhatunk. Ezt a mezőtípust akkor használjuk, ha a mezőben tárolt szöveg hossza rekordonként nagyon eltérő. Akkor is ezt a mezőtípust használjuk, ha csak néhány rekordnál van szükségünk erre a mezőre, mert szemben az előző (Szöveg) mezőtípussal, itt mindig csak annyival nő a tábla, és így az adatbázis mérete is rekordonként, amennyi karaktert a mező tartalmaz.

Szám – Ezt a típust akkor használjuk, ha a mezőben szám típusú adatot akarunk tárolni, akár egész, akár tört számot. Hogy milyen típusú számot akarunk a mezőben tárolni, azt a mezőméret tulajdonságnál állíthatjuk be.

Dátum/Idő – Ha dátumot vagy időpontot akarunk tárolni egy mezőben, használjuk ezt a típust. Semmiképpen ne használjuk ilyen célból a Szám ill. a Szöveg mezőtípust

Pénznem – Pénznemek és matematikai számításokban használt, maximum négy tizedesjegy pontosságú numerikus adatok. A tizedesjeltől balra 15, a tizedesjeltől jobbra 4 számjegy állhat.

Számláló – Egyedi, egymást egyesével követő számok vagy a Microsoft Access által megadott véletlen szám, amelynek célja, hogy egyértelműen azonosítson minden rekordot. Az ilyen típusú mező értékét nem mi adjuk meg, hanem az Access adja meg automatikusan.

Igen/Nem – Az ilyen típusú mezőnek csak kétféle értéke lehet (pl. Igen/Nem vagy Ki/Be stb.).

OLE objektum – Az ilyen típusú mező tartalma egy a táblához csatolt, vagy abba beágyazott objektum (Excel adatlap, Word dokumentum, grafika, hang vagy más bináris adat).

Keresés Varázsló – Létrehoz egy olyan mezőt, amely lehetővé teszi, hogy egy másik táblából vagy listából válasszunk egy értéket, utóbbi esetben egy lista vagy kombinált lista segítségével. Ha erre a lehetőségre kattintunk, akkor elindul a Keresés Varázsló, amely létrehoz egy Keresőmezőt.

Hiperhivatkozás – URL (pl. http://cnn.com) vagy UNC (pl. \\GEPNEV\megosztas\) típusú hivatkozást tartalmazó mező

4. ábra. Mezők típusai

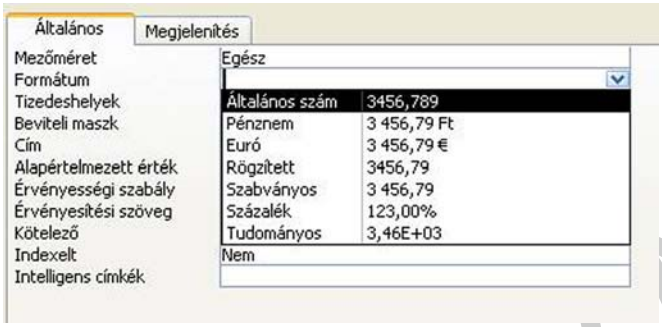
Mezők Formátuma:

Access két egymáshoz hasonló mezőtulajdonságot kínál: ezek a **Formátum (Format)** és a **Beviteli maszk (InputMask)**.

A Formátum (Format) tulajdonságot az adatok azonos alakban való megjelenítéséhez használjuk.. A megjelenítési formátum csak az adat beírását és mentését követően lép érvénybe, vagyis a mezőben nem látható semmi, ami megszabná vagy irányítaná, hogy az adatot milyen formátumban írjuk be. Ha az adatok beírási módját irányítani kell, akkor az adatmegjelenítési formátum helyett vagy mellett használjunk beviteli maszkot.

A beviteli maszk biztosítja, hogy az adatok a megadott formátumot vegyék fel, és segítségével megadhatjuk a beírható értékek jellegét is.

Ha egy mező esetén megadjuk mind a megjelenítési formátumot, mind a beviteli maszkot, akkor a Access a beviteli maszkot használja az adatokat beírásakor és szerkesztésekor, a Formátum beállítás pedig azt határozza meg, hogy az adatok a rekord mentésekor hogyan jelenjenek meg. Ha mind a Formátum (Format), mind a Beviteli maszk (InputMask) tulajdonságot megadjuk, ügyeljünk arra, hogy ne ütközzenek egymással.



5. ábra. Formátumok

a. Egyedi kulcs létrehozása

Kattintsunk a mezőre, melyet szeretnénk elsődleges kulcsnak beállítani. Kattintsunk a sárga kiskulcsot ábrázoló ikonra, vagy válasszuk a Szerkesztés menü Elsődleges kulcs menüpontját. Egy táblának csak egy elsődleges kulcsa lehet. Ha nem hozunk létre elsődleges kulcsot, a tábla mentésekor a program felajánlja annak beállítását.

Ha befejeztük a tábla szerkesztését

Miután befejeztük a tábla szerkezetének kialakítását, válasszuk a Fájl menü Bezárás menüpontját. Ekkor megadhatjuk, hogy mi legyen a táblának a neve, majd létrejön az üres tábla.

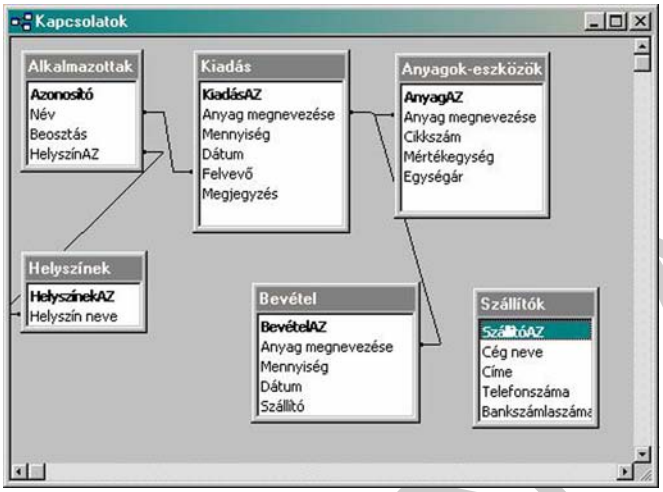
Tábla feltöltése adatokkal

Ha már létrehoztunk táblákat és szeretnénk ezeket adatokkal feltölteni, akkor először jelöljük ki azt a táblát, amelyet szeretnénk adatokkal feltölteni, majd nyomjuk meg az Megnyitás gombot. Ekkor megjelenik a tábla Adatlap nézetben, és így beírhatjuk új adatainkat a táblába. Ha befejeztük az adatok felvitelét, válasszuk a Fájl menü Bezárás menüpontját.

5. Kapcsolat kialakítása táblák között

Táblák közötti kapcsolatokat úgy hozhatunk létre, hogy meghatározzuk, melyik mezőkön keresztül kapcsolódnak egymáshoz, vagyis megadjuk a kapcsoló mezőket és a kapcsolat típusát.

Kapcsolatok kialakításához válasszuk az Eszközök menü (vagy a jobb egérgombbal kattintva, a helyi menü) Kapcsolatok menüpontját. A kapcsolatot úgy határozzuk meg, hogy kijelöljük a két mezőt, melyeken keresztül a két tábla kapcsolódik egymáshoz. A két mező közül az egyiknek elsődleges indexnek kell lennie. A két mezőt úgy tudjuk kijelölni, hogy az egyik mezőt megfogjuk, és áthúzzuk az egér segítségével a másik mezőre.



6. ábra. Táblák kapcsolatai

a. A kapcsolat típusai

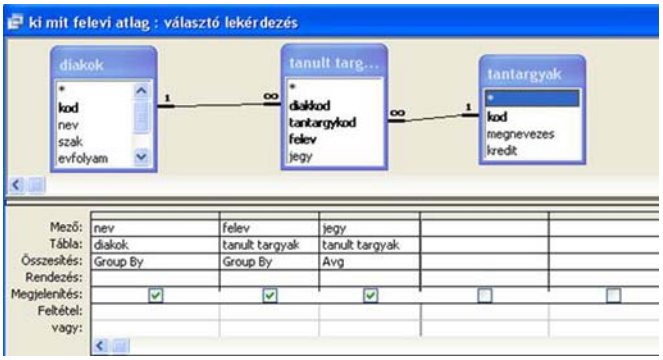
A kapcsolatok típusai lehetnek:

- Egy az egyhez kapcsolat
- Egy a többhöz kapcsolat
- Több a többhöz kapcsolat

6. Lekérdezések:

Többfajta lekérdezést létrehozhatunk attól függően, hogy az adatokat milyen logikai elrendezésben szeretnénk látni, illetve törölhetünk, módosíthatunk és hozzá is fűzhetünk a táblák adataihoz velük. Az alábbi lekérdezések közül választhatunk:

- **Választó lekérdezés:** A leggyakrabban ezt a típust használjuk. Több táblából válogathatjuk a mezőket. A tábla adatainak megváltozásával a lekérdezés eredménye is megváltozik (dinamikus lekérdezés)



7. ábra. Választó lekérdezés

- **Kereszttáblás lekérdezés**: Egy tábla egyik mezőjének összegzett értékeit (összegét, számát, átlagát) jeleníti meg. Az egyes mezőket sor ill. oszlop fejlécként is megadhatjuk.
- **Táblakészítő lekérdezés**: A táblakészítő lekérdezés egy vagy több táblából kér le adatokat, majd az eredményhalmazt betölti egy új táblába. Az új tábla lehet a megnyitott adatbázisban, vagy létrehozható egy másik adatbázisban.
- **Frissítő lekérdezés**: Adott feltételt kielégítő rekordok valamely mezőjének (ill. mezőinek) módosítása
- **Hozzáfűző lekérdezés**: Segítségével rekordokat másolhatunk át egyik táblából egy másik táblába, ill. új rekordot vihetünk fel egy adott táblába
- **Törlő lekérdezés**: Az ilyen típusú lekérdezés töröl egy adott feltételnek eleget tevő minden rekordot.

7. Lekérdezések létrehozása

Lekérdezéseket háromféleképpen hozhatunk létre:

- Az SQL (Structured Query Language) nyelv segítségével
- QBE rács segítségével: Lekérdezés létrehozása tervező nézetben
- Varázsló segítségével. Ez a lekérdezések legegyszerűbb módja

a. Lekérdezés létrehozása tervező nézetben

A tervező nézetben hozhatunk létre összetett lekérdezéseket, olyan módon, hogy a lekérdezés feltételeit a lekérdezés-ablak alsó részében megszerkesztjük.

A tervező nézetben az adatbázis tábláit, és a közük létesített kapcsolatokat a segédablak felső részében jeleníthetjük meg. A segédablakon jobb egérgombbal kattintva, vagy a Nézet menüben válthatunk a tervező nézet, az eredményeket mutató adatlap nézet és a lekérdezést SQL nyelven tartalmazó SQL nézet között.

A tervező nézet lehetővé teszi, hogy a Lekérdezés menüben kiválasztva a megfelelő menüpontot, létrehozzunk a kereszttáblás, frissítő, hozzáfűző, törlő lekérdezéseket, vagy a lekérdezéssel új táblát hozzunk létre.

Lekérdezéskor felhasználhatjuk az SQL-t is a könnyebb használat érdekében.

Nyissuk meg a módosítandó lekérdezést, majd kattintsunk az eszköztár Nézet gombja melletti nyílra, majd válasszuk az SQL nézet elemet.

SQL

Az SQL, a Structured Query Language angol elnevezésből származtatott mozaikszó, mely a relációs adatbázis-kezelő rendszerek ma legelterjedtebb, szabványosított adatbázis-kezelő nyelvének rövidített megnevezése. Az angol elnevezés szószerinti magyar fordítása Strukturált Lekérdező Nyelv, ám a nyelv valójában komplett adatbázis-kezelő nyelv, amely tartalmaz adatdefiníciós, adatmanipulációs és az adatfelügyelet körébe tartozó utasításokat is.

1. Az SQL szerepe és tulajdonságai

Az SQL egy **szabványosított** lekérdező nyelv, amit több relációs adatbázis-kezelő ismer, különböző operációs rendszeri környezetben. Ennek óriási jelentősége van az adatbázis alkalmazások fejlesztőinek körében, mert így az alkalmazások a különböző operációs rendszerek és adatbázis-kezelők közötti módosítás nélkül vagy csekély módosítással átvihetők. Tömör, felhasználó közeli nyelv, mely alkalmas hálózatokon adatbázis-kezelő szerver és kliensek közötti kommunikációra.

Az SQL **nem algoritmikus** nyelv, hiszen nem tartalmaz algoritmikus szerkezeteket (elágazás, ciklus, szekvencia), illetve nem tartalmaz normál fájlkezelésre vagy felhasználói képernyőkezelésre vonatkozó utasításokat, kimondottan az adatbázis-kezelés, adatkezelés megvalósítására szolgál. Az SQL-ben az eredményként keresendő adatokat specifikálni kell, de semmilyen előírást nem kell megadni az eredmény előállítására vonatkozóan.

Az SQL **halmaz orientált** nyelv, mely a relációkon dolgozik. A halmaz orientáltság azt jelenti, hogy a feladat nem eljárásszerű megfogalmazását kell megadni, mely a reláció vagy relációk kiválasztott sorain hajtódik végre, tehát nem kell a művelet végrehajtásának lépéseit definiálni. Az SQL **nem rekurzív nyelv**, tehát nem tartalmaz önmagát meghívó relációt.

Az SQL és a relációs adatmodell kapcsolatáról elmondható, hogy az SQL több különböző szinten keresztül közelíti az elméleti relációs adatmodellt, melyre igaz, hogy bizonyos elemeket nem tartalmaz, ezért az SQL a relációs adatmodellnek csak részleges leképezése.

Hogy az SQL már önmagában használható legyen, ne kelljen hozzá más kiegészítő nyelv, ezért az adatkezelési funkciókon belül a legszélesebb igényhalmaznak kell megfelelnie. Az SQL az előbb említett kritériumok miatt négy résznyelvből épül fel:

- **SQL DDL (Data Definition Language = Adat Definíciós Nyelv)**, azaz az SQL adatdefiníciós nyelve, mely lehetővé teszi a felhasználók számára, hogy új adatbázisokat és azok sémáját hozhassák létre, vagyis az adatok logikai sémáját egy speciális nyelven adhassák meg. A használatos sémakezelő utasítások közé tartozik például az adatbázisok, táblák létrehozása, illetve ezek törlése, módosítása.

- **SQL DML (Data Manipulation Language = Adat Manipulációs Nyelv)**, azaz az SQL adatmanipulációs nyelve, mely lehetővé teszi a felhasználók számára, hogy az adatokat módosíthassák. A használatos adatkezelő utasítások például az adatok törlése, módosítása, vagy új adat felvétele.
- **SQL DQL (Data Query Language = Adat Lekérdező Nyelv)**, azaz az SQL adatlekérdező nyelve, mely segítségével a felhasználók az adatokat lekérdezhetik, kilistázhatják valamilyen szisztéma szerint. Az adatok lekérdezését egyes irodalmakban legtöbbször a DML körében említik, de a relációs modellben betöltött fontosságuk miatt a lekérdező utasítások külön csoportot alkotnak, melyekre a Query elnevezést használjuk.
- **SQL DCL (Data Control Language = Adatelérést Vezérlő Nyelv)**, azaz az SQL adatelérést vezérlő nyelve. Ezzel szabályozhatjuk, vezérelhetjük a műveletek végrehajtását, hiszen a megvalósított relációs adatbáziskezelő nyelvek hatásának köszönhetően az SQL tartalmaz a relációs adatmodellhez szorosan nem kötődő utasításokat is.

Az SQL utasításokat több különböző módon juttathatjuk el egy relációs adatbázis-kezelő rendszerhez. A legegyszerűbb egy interaktív SQL parancsértelmező használata, ami közvetlenül kapcsolódik a relációs adatbázis-kezelőhöz. Mivel ilyen esetben csak adatkezelő utasításokat adhatunk meg, ezért ez nem alkalmas normál alkalmazói programok készítésére. Ehhez az SQL utasításokat algoritmikus elemekkel kell bővíteni, melyre kétféle lehetőség kínálkozik. Egyrészt magát az SQL-t lehet egy létező algoritmikus nyelvbe beépíteni, beágyazni, másrészt az SQL-t lehet kibővíteni algoritmikus elemekkel. Persze ez utóbbi esetben kapott nyelv már messze esik az SQL szabványtól. Az SQL nyelvnek tehát két felhasználási területe van:

- **Önálló SQL**: Az SQL nyelv önálló alkalmazása esetén csak a nyelv utasításai állnak rendelkezésre. Erre általában akkor kerül sor, ha nincs megfelelő alkalmazás a feladat elvégzésére, illetve a negyedik generációs nyelvekbe építve használják az alkalmazások fejlesztői. Ilyen eszközök például a jelentéskészítő, az űrlapkészítő, vagy a menükészítő.
- **Beágyazott SQL**: A beágyazott SQL esetén egy harmadik generációs nyelvbe (C, PL/SQL, Pascal, Fortran, ... stb.) ágyazva alkalmazzuk az SQL nyelv elemeit. Az algoritmikus feladatokat a harmadik generációs nyelv, az adatbázis műveleteket, pedig az SQL végzi el. Jelen esetben a beágyazott SQL alkalmazását nem tárgyaljuk.

Összefoglalóan tehát a következőket mondhatjuk el az SQL-ről:

- Szabványosított lekérdező nyelv;
- Relációs algebrán alapszik;
- Szöveges, nem algoritmikus, előíró jellegű utasításokat tartalmaz;
- Halmazorientált;
- Nem rekurzív nyelv;
- Négy utasításcsoportot tartalmaz:
    - Adatdefiníciós,
    - Adatmanipulációs,

    - Adatlekérdező,
    - Adatvezérlő;
- Felhasználható önálló vagy beágyazott SQL-ként is.

8. ábra. SQL logó[2]

Az SQL, hasonlóan más programozási nyelvekhez, utasításokból épül fel. Az egyes utasítások tartalmazhatnak alapszavakat, azonosítókat, kifejezéseket, konstansokat, listákat, és elválasztó jeleket is. A nyelv utasításai meglehetősen összetettek, úgynevezett záradékokból állnak. A záradékok használatának egy része nem kötelező, azonban a használt záradékok sorrendje minden esetben kötött.

Az azonosítókat az adatbázis, az adatbázis elemek, az alaptáblák és a nézettáblák, valamint a táblák oszlopainak megnevezésére használják. Az azonosítók mindig betűvel kezdődnek, de a második pozíciótól kezdve tartalmazhatnak betűket, számjegyeket és akár aláhúzás jelet is. Egyes számítógépes alkalmazások további karakterek használatát is engedélyezik. Az azonosítók hosszát a különböző számítógépes alkalmazások saját hatáskörükben szabályozzák. A nyelv alapszavai, illetve kulcsszavai azonosítóként nem használhatók. Úgy kell őket megválasztani, hogy azok minden esetben egyediek legyenek, az oszlopnevek táblázaton belül, a táblázat-, nézet-, indexnevek, pedig adatbázison belül.

[2] http://www.cyberhades.com/wp-content/uploads/2009/01/sql-logo.png

Mivel a nyelv szabad írásmódú, ezért az egyes utasítások tetszés szerint tördelhetők. A záradékokat általában új sorba szokás írni. Az utasítások végét a pontosvessző jelzi. Az utasítások kisbetűs, nagybetűs, és kevert formában is kiadhatók (pl.: a FROM nagybetűkkel ugyanazt jelenti, mint a from, vagy akár a FrOm is). Az utasításokban feltűntetett listákban az egyes elemeket egymástól, vesszővel kell elválasztani, és bizonyos esetekben a listákat kerek zárójelpárba kell helyezni. Az utasítások egy bizonyos részébe szabályok szerint más utasítások is beágyazhatók. Az utasítások alapszavai és egyéb elemei közé elválasztójelként szóközt, vagy soremelést kell tenni. A zárójelek, és vesszők betöltik az elválasztójelek szerepét. Az elemek elválasztása során alkalmazott felesleges szóközök és zárójelpárok nem okoznak hibát.

2. Adattípusok, műveletek

Az SQL nyelvben lehetőség van numerikus, szöveges, dátum jellegű, bináris, logikai, valamint szerkezet nélküli adattípusok használatára.

a. Adattípusok és értelmezésük:

- **BIT (n):** Rögzített hosszúságú bitsorozat
- **BIT VARYING (n)**: Változó hosszúságú bitsorozat
- **SMALLINT; INTEGER; LONG:** Egész
- **DECIMAL (x, [y])**: Fixpontos valós (x az összes jegyek, y a tizedes jegyek száma)
- **FLOAT; REAL; DOUBLE PRECISION**: Lebegőpontos
- **DATE**: Dátum
- **TIME**: Időpont
- **TIMESTAMP**: Dátum és időpont
- **INTERVAL**: Időtartam
- **CHAR (n)**: Rögzített hosszúságú karaktersorozat
- **CHAR VARYING (n)**: Változó hosszúságú karaktersorozat
- **LOGICAL; BOOLEAN**: Logikai típusok (az ilyen adatokat bitsorozatként is tárolhatjuk)
- **BLOB**: Változó hosszúságú bit- és karakter-sorozat (szerkezet nélküli adattípus), lehetővé teszi nagyméretű kép, hang, szöveg ... stb. tárolást.

Az SQL utasításokban az egyes adattípusoknak megfelelő konstansokat is alkalmazhatunk. A szöveg és dátum típusú konstansokat idézőjelek között kell megadni. A dátumformátum beállítására két szinten van lehetőség: az adatbázis-kezelő rendszerben, vagy az operációs rendszer szintjén. Ez a beállítás meghatározza az év-, hónap-, nap- adatok megadási sorrendjét, illetve a közöttük alkalmazandó elválasztójeleket. A dátumoknál használatos szabvány elválasztójel a kötőjel, az idő konstansokban az óra, perc, másodperc elhatárolására pedig, a kettőspontot használják. A dátumot és az időt szóközzel választják el egymástól, ha a konstans mindkettőt tartalmazza. Viszont azt fontos észben tartani, hogy a különböző számítógépes megvalósítások a felsoroltaktól eltérő jeleket is tartalmazhatnak az egyes konstansok esetén.

b. Adattípusok körében támogatott műveletek

A nyelv az alapvető adattípusok körében a következő műveleteket támogatja:

- Matematikai műveletek:
  - előjelváltás (+,-)
  - számtani alapműveletek (+,-,*,/)
  - hatványozás (** vagy ^)
- Karaktersorozattal végezhető műveletek: összefűzés (+,||, &)
- Bináris és logikai műveletek:
  - Tagadás (NOT)
  - Logikai ÉS (AND)
  - Logikai VAGY (OR)
- Dátumok, időpontok és időtartamok körében használható műveletek:
  - Számtani alapművelet (+,-,*,/)

c. Összehasonlító műveletek

A keresési feltételekben szerepelhetnek összehasonlító műveletek, melyek egyaránt használhatók dátum-, numerikus- és karakteres típusú adatok esetén is. Ezekre az összehasonlító műveletekre théta operátorok összefoglaló névvel szokás hivatkozni. Az említett operátorok a következők:

= (egyenlő)	> (nagyobb)
>= (nagyobb egyenlő)	< (kisebb)
<= (kisebb egyenlő)	<>, !=, ^= (nem egyenlő)

Az összehasonlító operátorok tagadására a NOT kulcsszó, illetve az összehasonlító műveletek tagadására a ! jel használható. Például a nem egyenlő összehasonlító művelet lehetséges formái: !=, ^=, <>, NOT= lehetnek.

A numerikus adattípusok egymással kompatibilisek, tehát bármelyik két típus egymással összehasonlítható. A karaktersorozatok összehasonlítása balról jobbra, a karakterek belső kódja alapján történik. Mivel a kis- és nagybetűk kódja nem egyezik meg, ezért ezek azonos alakra való konvertálása szükséges a megfelelő függvények felhasználásával. Ez a megállapítás azonban csak az idézőjelek közé tett kifejezések esetén igaz. Az attribútumnevek, a relációnevek, stb. mind-mind függetlenek attól, hogy kis- vagy nagybetűvel írtuk őket.

d. Predikátumok

Az SQL nyelv a fenti összehasonlító műveleteken kívül más műveleteket, úgynevezett predikátumokat is definiál, melyek a következők:

IS: Olyan értékegyezést vizsgál, melyben azok a sorok kerülnek kiválasztásra, ahol az érték megegyezik az előírttal. Az adatbázis-kezelésből már jól ismert NULL érték vizsgálata nem lehetséges az =, <> operátorokkal, csak az IS predikátummal, vagy az adatbázis-kezelő rendszer egy megfelelő függvényével.

Szintaktikája: oszlopnév1 IS NULL

oszlopnév2 IS NOT NULL

LIKE: Azokat a sorokat választja ki, ahol a karakteres típusú oszlop vagy kifejezés a megadott mintát tartalmazza. A mintában két speciális helyettesítő karakter adható meg: a % jel tetszőleges hosszúságú karaktert jelöl, vagy az _ aláhúzás karakter, mely egy tetszőleges karaktert jelent. Ezek helyén a vizsgált karaktersorozatban tetszőleges karakterek szerepelhetnek.
Szintaktikája: oszlopnév1 LIKE 'minta'

Operátor	Értelmezés
LIKE 'a%'	Kilistáz minden 'a' betűvel kezdődő kifejezést
LIKE 'x_'	Megadja az összes 'x'-szel kezdődő kétbetűs kifejezést
LIKE '%a%'	Megad minden 'a' betűt tartalmazó kifejezést
LIKE '_a%x'	Megadja az összes olyan kifejezést, melynek második betűje 'a' és 'x'-re végződik

BETWEEN: Azokat a sorokat választja ki, ahol az oszlop adatértéke, vagy annak behelyettesítésekor a kifejezés értéke a megadott értékhatárok közé esik. (x <= oszlopnév1 <= y)

Szintaktikája: oszlopnév1 BETWEEN x AND y

IN: Azok a sorok kerülnek kiválasztásra, ahol az oszlop adatértéke vagy annak behelyettesítéskor a kifejezés értéke megegyezik az IN predikátumot követő értékek egyikével. A halmazban szereplő értékek egy kerek zárójelpárban adhatók meg. Az értékhalmaz egy allekérdezés eredménye is lehet.

Szintaktikája: oszlopnév1 IN ('a','b','c')

oszlopnév2 IN (SELECT oszlopnév3 FROM táblázatnév1)

ANY, SOME: Olyan összehasonlító művelet teljesülését vizsgálja allekérdezéssel előállított értékhalmazban, ami akkor eredményez igaz értéket, ha az allekérdezés során kiválasztott értékek közül legalább egyre teljesül a kijelölt művelet.

Szintaktikája: WHERE oszlopnév1 <= ANY (SELECT oszlopnév2 FROM táblázatnév1)

ALL: Olyan összehasonlító művelet teljesülését vizsgálja allekérdezéssel előállított értékhalmazban, ami akkor eredményez igaz értéket, ha az allekérdezés során kiválasztott valamennyi értékre teljesül a kijelölt művelet.

Szintaktikája: WHERE oszlopnév1 < ALL (SELECT oszlopnév2 FROM táblázatnév1)

EXISTS: Ha az allekérdezés eredménye nem üres, tehát kiválaszt legalább egy sort, akkor igaz értéket ad vissza.

Szintaktikája: WHERE EXISTS (SELECT * FROM táblázatnév1)

UNIQUE: Olyan predikátum, mely az allekérdezés eredményeként ismétlődő sorok értékét vizsgálja. Ha nem talál két megegyező sort, akkor igaz értéket, egyébként pedig, hamis értéket ad vissza.

Szintaktikája: WHERE UNIQUE (SELECT oszlopnév1 FROM táblázatnév1)

A predikátumokkal bizonyos feltételeket írhatunk elő, melyek teljesülését a rendszer ellenőrzi, és az eredményt egy logikai értékkel jelzi. A predikátumok a kiválasztási, illetve vizsgálati feltételekben, tehát az adatdefiníciós utasításokban (CHECK), valamint az adatkezelő utasításokban (WHERE) és azok záradékaiban (HAVING) használhatók. Ezeket majd később, a SELECT utasítás tárgyalása során ismertetjük.

Az adatbázis-kezelő rendszerekben számos karaktersorozatot, dátumot, időpontot valamint matematikai és típus-átalakító függvényt definiáltak. Ilyen függvények használhatók az SQL nyelvben is. Az SQL nyelv maga is definiál bizonyos függvényeket, ezeket aggregát vagy összesítő függvényeknek nevezzük. Az aggregát függvényeket a SELECT utasítás GROUP BY záradékának tárgyalása során ismertetjük. Azonban először nézzük át az SQL nyelvben használatos függvények típusaival és szerepével.

e. Karakteres függvények

Függvény	Magyarázat	Példa
ASC (szöveg)	A szöveg első karakterének ASCII kódját adja vissza.	ASC ('aerobik') = 65
CHR (egész)	A számnak megfelelő ASCII kódú karaktert adja vissza.	CHR (65) = 'a'
INITCAP (szöveg)	A szavak kezdőbetűit alakítja nagybetűssé.	INITCAP ('kis János') = 'Kis János'
INSTR (szöveg1, szöveg2, kezdet, hányadik)	A szöveg1-ben a szöveg2 hányadik előfordulását adja meg a kezdettől számolva. (Hányadik és kezdet paraméter megadása elmaradhat)	INSTR ('aerobik', 'robi') = 3

LENGTH (szöveg)	A szöveg hosszát adja meg.	LENGTH ('aerobik') = 7
LOWER (szöveg)	Kisbetűssé alakítja a szöveget.	LOWER ('JÓGA') = 'jóga'
LPAD (szöveg, hossz, karakterek)	A szöveget kiegészíti balról a megadott karakterekkel az adott hosszig. A karaktereket nem kötelező megadni, ekkor szóközt használunk helyettük.	LPAD ('k', 3) = '..k' LPAD ('k', 7, '23') = '232323k'
LTRIM (szöveg, karakterek)	A szöveg elejéről levágja a karakterekkel egyező részt. A karaktereket nem kötelező megadni, ekkor szóközt írunk helyette.	LTRIM ('..k') = 'k' LTRIM ('gyógytorna', 'gyógy') = 'torna'
RPAD (szöveg, hossz, karakter)	A szöveget kiegészíti jobbról a megadott karakterekkel az adott hosszig. A karaktereket nem kötelező megadni, ekkor szóközt írunk helyette.	RPAD ('k', 3) = 'k..' RPAD ('k', 7, '23') = 'x232323'
RTIM (szöveg, karakter)	A szöveg végéről levágja a karakterekkel megegyező részt. A karaktereket nem kötelező megadni, ekkor szóközt írunk helyette.	RTIM ('k..') = 'k' RTIM ('gyógytorna', 'anrot' = 'gyógy'
SUBSTR (szöveg, n, m)	Kiválasztja a szöveg egy olyan részszövegét, amely annak n sorszámú karakterétől kezdve m darab karaktert tartalmaz.	SUBSTR ('aerobik', 2, 1) = 'e' SUBSTR ('aerobik', 3) = 'robik'
TRANSLATE (szöveg, mit, mire)	A szövegben előforduló mit karaktereket kicseréli a mire karaktereire.	TRANSLATE ('stepaerobik', 'step', 'gyermek') = 'gyermekaerobik'
UPPER (szöveg)	A megadott szöveget nagybetűssé alakítja.	UPPER ('jóga') = 'JÓGA'

f. Dátum függvények

Függvény	Magyarázat	Példa
ADD_MONTH (dátum, n)	A dátumhoz n hónapot ad.	ADD_MONTH ('15-MAY-95', 2) = '15-JUL-95'
LAST_DAY (dátum)	A dátumban szereplő hónap utolsó napját adja.	LAST_DAY ('3-APR-95') = '30-APR-95'
MONTH_BETWEEN (dátum1, dátum2)	A két dátum közötti idő hónapokban	MONTH_BETWEEN ('2-JAN-93', '3-DEC-94')= '23'
NEXT_DAY (dátum, nap)	A dátum utáni első napra eső dátumot adja meg.	NEXT_DAY ('10-APR-85','TUESDAY') = '11-APR-85'
ROUND (dátum,	Dátum kerekítése a megadott	

formátum)	formátum szerint.	
TO_CHAR (dátum, formátum)	Az adott dátum konvertálása a megadott karakteres formátumba.	TO_CHAR (1-JAN-83, 'YY.MM.DD') = '83.01.06.'
TO_DATE (szöveg, formátum)	Az adott szöveg dátummá alakítása a megadott formátum szerint.	TO_DATE ('83.01.06', 'DD-MM-YY') = '6-JAN-83'
TO_NUMBER (szöveg)	Megfelelő alakú szövegből szám jellegű mezőt képez.	
TRUNC (dátum, formátum)	Dátum csonkítása a megadott formátum szerint.	

g. Numerikus függvények

Függvény	Magyarázat	Példa
ABS (érték)	Abszolút érték	ABS (-5) = 5
CEIL (érték)	Az értéknél nagyobb vagy egyenlő legkisebb egész.	CEIL (5.15) = 6
FLOOR (érték)	Az értéknél kisebb vagy egyenlő legnagyobb egész	FLOOR (2.83) = 2
MOD (érték, osztó)	Osztási maradék/td>	MOD (7, 3) = 1
POWER (érték, kitevő)	Hatványozás	POWER (3, 4) = 81
ROUND (érték, pontosság)	Kerekítés a megadott jegyig. Negatív pontosság is megadható.	ROUND (233.523, 1) = 233.5 ROUND (132.234,-2) = 200
SIGN (érték)	Előjel függvény	SIGN (-3) = -1
SQRT (érték)	Négyzetgyök-vonás	SQRT (121) = 11
TRUNC (érték, pontosság)	Csonkítás a megadott jegyig. Negatív pontosság is megadható.	TRUNC (253.456, 1) = 253.4 TRUNC (153.456,-2) = 100

h. Precedencia szabály

Az SQL-ben használatos kifejezéseknek van egy kiértékelési sorrendje, vagyis precedencia szabálya, mely szerint az azonos szinten levő műveletek eredménye balról jobbra kerül meghatározásra. Ez a sorrend a következő:

- zárójelek, belülről kifelé
- függvényhívások
- matematikai, szöveg és dátum műveletek
    - előjelváltás
    - hatványozás
    - multiplikatív műveletek (szorzás, osztás)
    - összevonási műveletek (összeadás, kivonás)
- összehasonlító műveletek: =, >, >= , <, <=, <>, !=, ^=
- logikai műveletek:
    - NOT
    - AND
    - OR

3. Az SQL DDL, vagyis az adatdefiníciós nyelv utasításai

Az adatdefiníciós nyelv segítségével hozhatjuk létre az adatbázisokat, a táblákat, a nézettáblákat és az indexeket, illetve ezeket meg is szüntethetjük.

a. Táblák létrehozása

A táblák létrehozására szolgáló SQL utasítás a CREATE TABLE, melynek általános alakja a következő:

```
CREATE TABLE táblanév

(mezőnév1 adattípus (szélesség) [NOT NULL],

mezőnév2 adattípus (szélesség) [NOT NULL],

...

mezőnévN adattípus (szélesség) [NOT NULL] );
```

A mezők típusát megadva, a tábla azon oszlopába más típusú adat nem vihető fel, erről az adatbázis-kezelő gondoskodik, illetve szükség esetén figyelmeztet. A NOT NULL a mező definíciójában arra utal, hogy az adat megadása kötelező, azaz nem lehet olyan sor a táblában, ahol az így definiált adat nincs kitöltve.

	Mező	Típus	Null
☐	tanár_id	int(4)	Nem
☐	név	char(40)	Nem
☐	tel	char(15)	Nem
☐	kor	char(3)	Nem
☐	irsz	int(4)	Nem
☐	város	char(50)	Nem
☐	cím	char(50)	Nem
☐	idegennyelvű	char(1)	Igen

9. ábra. SQL tábla[3]

(tanár_id: a tanár azonosítója, név: a tanár neve, tel: a tanár telefonszáma (pl.:06205849212). kor: 0-100 éves korú emberekkel foglalkozunk. irsz: a tanár irányítószáma, város: az a város, ahol a tanár lakik, cím: a tanár címe (utca, házszám), idegen nyelvű: értéke I/N lehet attól függően, hogy tart-e idegen nyelvűeknek órát vagy sem)

b. Táblák módosítása, törlése

A CREATE TABLE utasítással létrehozott táblák definícióját csak korlátozott számban módosíthatjuk, újabb mezőt adhatunk a táblához, vagy egy mező szélességét megnövelhetjük. Egy mező szélességének csökkentésére illetve törlésére nincs közvetlen mód. Ez csak abban ez esetben érhető el, ha a módosításoknak megfelelő üres táblát hozunk létre, amibe a szükséges adatokat átmásoljuk, majd töröljük az eredeti táblát. A tábla újabb mezővel való bővítésére az alábbi parancs szolgál:

ALTER TABLE táblanév

ADD mezőnév adattípus (szélesség) NOT NULL;

Az új mező a tábla legutolsó oszlopa lesz. A NOT NULL predikátum nem adható meg abban az esetben, ha a tábla már fel van töltve adatokkal, a mezők értéke pedig, a már meglevő sorokban NULL (definiálatlan) lesz. Az SQL nyelv a NULL, vagyis a még nem definiálatlan értékeket megkülönbözteti a nulla numerikus értéktől. Példaként vegyünk fel a tanár táblába egy újabb mezőt, melyben megadjuk a tanárok születési évét:

ALTER TABLE Tanár

ADD születési év INT(4) NOT NULL;

[3] http://ikon.inf.elte.hu/

Egy tábla mezőjének szélességét az ALTER TABLE paranccsal lehet megnövelni, mely a következőképp valósítható meg:

ALTER TABLE táblanév

MODIFY mezőnév adattípus (új szélesség) [NOT NULL];

Abban ez esetben, ha az adott mező csak NULL értéket tartalmaz, akkor lehetőség van az adattípus módosítására és a szélesség csökkentésére vagy növelésére is.

Teljes táblák törlésére is lehetőséget ad az SQL nyelv. Ezt az alábbi módon végezhetjük el:

DROP TABLE táblanév;

A parancs kiadása után a táblákban tárolt valamennyi adat, és a tábla definíciója is törlődik.

Az SQL nyelv lehetőséget ad a táblákban szereplő oszlopok törlésére is. Az erre szolgáló parancs a következő:

ALTER TABLE táblanév

DROP COLUMN oszlopnév;

c. Nézettáblák létrehozása, törlése

A nézet tábla az adatbázisban fizikailag nem létező relációs műveletek (szelekció, projekció, összekapcsolás, halmazműveletek) segítségével létrehozott tábla, mely a relációkhoz hasonlóan kezelhető. A táblázatok adataiból a SELECT utasítás segítségével nézethez rendelhetjük adatainkat. A nézettáblák alkalmazási lehetőségei:

Származtatott adattáblák létrehozása, amelyek a törzsadatok módosításakor automatikusan módosulnak (pl. összegző táblák).

Bizonyos adatok elrejtése egyes felhasználók elől (adatbiztonság vagy egyszerűsítés céljából).

A nézettábla tehát az adatbázisban létező táblán vagy táblákon végrehajtott művelet eredményét tartalmazó olyan új tábla, mely mögött a valóságban nem áll megfelelő reláció, azaz a virtuális tábla nem tartalmaz adatokat. Egy nézettáblát az alábbi paranccsal határozhatunk meg:

CREATE VIEW nézettábla név [aliasnév, aliasnév,...]

AS SELECT...;

A CREATE VIEW végrehajtásakor a rendszer csak letárolja a nézettábla definícióját, és majd csak a rá való hivatkozáskor generálja a szükséges adatokat.

Az alaptáblázatok változásai a nézet táblában is megjelennek. A nézet tábla örökli az alaptáblázatokban definiált értékhalmazok, a kulcsok egyediségére és az idegen kulcs hivatkozási épségének fenntartására vonatkozó megszorításokat. A nézet tábla oszlopaihoz új név megadása nem szükséges, ekkor az alaptáblázat oszlopainak nevét örökli. Ha egy oszlopnevet megadtunk, akkor a többit is jelölni kell. Új oszlopnevet csak abban az esetben kell megadni, ha

- a SELECT utasítás oszloplistája származtatott oszlopot és/vagy aggregát függvényt tartalmaz
- két vagy több alaptáblázat azonos nevű oszlopa szerepel a SELECT utasítás oszloplistájában.

A nézettáblák megszűntetése a relációkhoz hasonlóan a DROP parancs segítségével végezhető el. Példaként töröljük az előbb létrehozott nézettáblát:

DROP VIEW nézettáblanév;

d. Indexek létrehozása, törlése

A táblákhoz rendelhetünk indexeket, melyek helyes megválasztása esetén a lekérdezések felgyorsíthatók. Az indexek létrehozására a következő utasítás szolgál.

CREATE [UNIQUE] INDEX indexnév

ON táblanév (mezőnév1, mezőnév2, ... [ASC/DESC]);

Az utasítás a megadott tábla felsorolt oszlopaira, mint indexkulcsra generál le egy indexet. Az utasításban szereplő ASC és DESC alapszavak az indexkulcs értékének nagyság szerinti növekvő illetve csökkenő sorrendjét jelölik. A növekvő sorrend az alapértelmezés szerinti, tehát nem kell megadni az ASC alapszót, ha ilyen rendezettséget szeretnénk. Az index elkészítésekor, ha az indexkulcs egynél több oszlopot is tartalmaz, továbbá az első oszlopbeli értékek azonosak, akkor a második oszlop alapján történik a rendezés.

Mivel indexeket hozhatunk létre, így azokat törölhetjük is. Az indexek megszűntetése az alábbi paranccsal történhet:

DROP INDEX indexnév ON tábla;

4. Megszorítások

a. Táblaszintű megszorítások

- elsődleges kulcs
- idegen kulcs
- egyediség
- logikai feltétel

b. Mezőkre vonatkozó megszorítások

- alapértelmezett érték
- kötelezőség (NOT NULL érték)

Először is tekintsük át, hogyan néz ki a CREATE TABLE utasítás megszorítással

CREATE TABLE táblanév

(mezőnév1 adattípus (szélesség) [mező szintű megszorítás1]

mezőnév2 adattípus (szélesség) [mező szintű megszorítás2],

...
mezőnévN adattípus (szélesség) [mező szintű megszorításN],

[táblaszintű megszorítás1],

[táblaszintű megszorítás2],

...

[táblaszintű megszorításN]

);

Az itt említett mezőszintű megszorításokat már ismertettük a Predikátumok című témakör során. Gondolok itt a NULL, vagy a NOT NULL megszorításokra. A táblaszintű megszorításoknak viszont van egy általános szintaxisa:

CONSTRAINT megszorítás neve [megszorítás definíciója]

A CONSTRAINT záradék segítségével ellenőrzési feltételeket adhatunk meg az adott oszlop esetében. Az ilyen ellenőrzési feltételek egyik legfontosabbika az elsődleges kulcs megadására vonatkozó feltétel.

c. Elsődleges kulcs

Az elsődleges kulcs egyedi azonosítója a sornak, tehát az egész rekordot meghatározza. Az esetek 95%-ában használunk elsődleges kulcsot.

Leggyakrabban használt elnevezése: pk_Táblanév.

CONSTRAINT megszorítás neve PRIMARY KEY (mezőnév1, mezőnév2)

d. Idegen kulcs

Az idegen kulcs olyan mező vagy mezők a táblában, amely egy másik táblában elsődleges kulcsként előforduló értékeket vehet fel. Ezek létrehozása esetén nagyon fontos, hogy figyeljünk a típus kompatibilitásra. A hivatkozott táblából sosem törölhető olyan rekord, amely elsődleges kulcsára hivatkozik a hivatkozó tábla.

Az idegen kulcs jelölésére általában nyilat használunk a hivatkozó táblától a hivatkozott tábláig. Az idegen kulcs mindig egy a sokhoz kapcsolatot valósít meg.

e. Egyediség

Az egyediséget akkor használjuk, ha több kulcs is szerepel a táblában. Mivel olyan nincs, hogy másodlagos vagy harmadlagos kulcs, ezért minden további, azaz az elsődleges kulcs után megadott kulcsot egyediség megszorítással adunk meg. Ez logikailag ugyanazt jelenti, mint az elsődleges kulcs, tehát minden értéknek különbözőnek kell lennie az adott mezőben. Azonban annyi eltérés van az elsődleges kulcshoz képest, hogy tartalmazhat NULL értékű mezőt is, de csak egyszer. Amennyiben a megszorítás több mezőre vonatkozik, akkor a mezőkben szereplő értékek együttesének kell egymástól különbözőnek lennie.

Szokásos elnevezése: uq_táblanév.

f. Logikai feltétel

Segítségével logikai feltételeket adhatunk meg a tábla mezőire vonatkozólag. A feltételben különböző logikai operátorokat (pl.: IN, OR, AND), összehasonlító relációkat, és egyéb műveleteket is használhatunk.

Szokásos elnevezése: ck_táblanév.

g. Alapértelmezett érték

Ha egy rekordnál nem adjuk meg az adott mező értékét, akkor az, az alapértelmezett értéket fogja felvenni.

Szintaxisa: DEFAULT érték.

h. Kötelezőség (nem null érték)

A kötelezőség olyan megszorítás, amely nem engedi meg olyan sorok előfordulását, amelyben az adott attribútum érték nincs megadva. Az elsődleges kulcs mezői mind-mind NOT NULL értékűek. Ha egy definícióban van alapértelmezett érték és kötelezőség is, akkor a sorrend a következő:

DEFAULT érték NOT NULL.

i. Táblaszintű megszorítások utólagos kezelése

Az előbbiekben már láttuk, hogy egy elkészült táblát utólag is módosíthatunk. Egy táblához adhatunk megszorításokat, de el is dobhatunk már elkészített megszorításokat. A megszorítások hozzáadása esetén az utasítás szintaxisa:

ALTER TABLE táblanév

ADD CONSTRAINT megszorítás neve;

j. Mezőszintű megszorítások utólagos kezelése

A mezőszintű megszorítások utólagos kezelésére szolgáló SQL utasítás a MODIFY. Használatának általános szintaxisa:

ALTER TALBE táblanév

MODIFY mezőnév adattípus [mező szintű megszorítás];

TANULÁSIRÁNYÍTÓ

A tananyagban áttekintettük az adatbázis kezelés alapfogalmait és a leggyakrabban használt eszközeit

Először néhány kérdésre kell válaszolnia, ehhez segítségül használhatja az információtartalom részben leírtakat.

Ezután gyakorlatban elvégzendő feladatokat kap, ehhez kifejezetten ajánlatos a szakmai információtartalomban leírtak használata. Ha ebben a részben valami nem "megy", kérje oktatója, tanára segítségét!

ÖNELLENŐRZŐ FELADATOK

1. feladat

Milyen mezőtípust használunk az azonosítókhoz és elsődleges kulcsokhoz?

2. feladat

Milyen nézetei vannak egy táblának?

3. feladat

Mi az idegen kulcs?

4. feladat

SQL parancs segítségével hozzon létre tetszőleges táblázatot!

5. feladat

A 4. feladatban létrehozott táblát tetszőlegesen módosítsa SQL parancs segítségével!

MEGOLDÁSOK

1. feladat

Számláló típusút.

2. feladat

Tervező és adatlap nézet.

3.feladat

Olyan azonosító amelynek segítségével egy másik táblázat elsődleges kulcsára hivatkozhatunk.

4.feladat

CREATE TABLE táblanév (mezőnév1 adattípus (szélesség) [NOT NULL],

5.feladat

ALTER TABLE táblanév

MODIFY mezőnév adattípus (új szélesség) [NOT NULL];

IRODALOMJEGYZÉK

FELHASZNÁLT IRODALOM

http://ecdlweb.hu/ (2010.10.30)

http://office.microsoft.com/hu-hu/access-help/a-microsoft-office-access-2003-bemutatasa-HA001071497.aspx (2010.10.30)

http://m-forum.hu/feladatok-peldak.php?lap=elmelet (2010.10.30)

http://ikon.inf.elte.hu/ (2010.10.30)

http://hu.wikipedia.org/wiki/SQL (2010.10.30)

AJÁNLOTT IRODALOM

Adatbázis-kezelés Bodnár István, Magyary Gyula: Megjelenés: 2003 Kiskapu Kiadó

Access 2003 zsebkönyv Bártfai Barnabás BBS-INFO Kiadó

SQL Trudy Pelzer, Peter Gulutzan Kiskapu Kiadó

SQL-lekérdezések földi halandóknak (CD melléklettel) John L. Viescas, Michael J. Hernandez Kiskapu Kiadó

http://ecdlweb.hu/ (2010.10.30)

http://office.microsoft.com/hu-hu/access-help/a-microsoft-office-access-2003-bemutatasa-HA001071497.aspx (2010.10.30)

http://m-forum.hu/feladatok-peldak.php?lap=elmelet (2010.10.30)

http://ikon.inf.elte.hu/ (2010.10.30)

http://hu.wikipedia.org/wiki/SQL (2010.10.30)

A(z) 1618-06 modul 006-es szakmai tankönyvi tartalomeleme felhasználható az alábbi szakképesítésekhez:

A szakképesítés OKJ azonosító száma:	A szakképesítés megnevezése

A szakmai tankönyvi tartalomelem feldolgozásához ajánlott óraszám:
30 óra

A kiadvány az Új Magyarország Fejlesztési Terv
TÁMOP 2.2.1 08/1-2008-0002 „A képzés minőségének és tartalmának fejlesztése” keretében készült.
A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Kiadja a Nemzeti Szakképzési és Felnőttképzési Intézet
1085 Budapest, Baross u. 52.
Telefon: (1) 210-1065, Fax: (1) 210-1063

Felelős kiadó:
Nagy László főigazgató