MySQL

Elektronikus jegyzet
Széchenyi István Egyetem
Távközlési tanszék

Távközlés-informatika szakirány
Protokollok és Szoftverek I.

Zsiga Bálint
Kovács Ákos

Az relációs adatbázis-kezelő rendszerekről

Kis mennyiségű adat tárolására általában elegendő a sokak által jól ismert táblázatkezelő programok által nyújtott szolgáltatás. Ám nagy mennyiségű adat esetén már körülményessé válik a kezelésük. Például, ha csak egy zh eredményt szeretnénk eltárolni, elegendő egyetlen táblázat, névvel, Neptun-kóddal és eredménnyel. Viszont mi van akkor, ha a diákok eredményeit szeretnénk ugyanígy tárolni? Esetleg ugyanitt szeretnénk elérni a diákok elérhetőségét, vagy egyéb adatait? A táblázat nagyságrendekkel több sorból, fog állni, és ha mondjuk megváltozik valakinek a telefonszáma minden sorban meg kell változtatni. Arról nem is beszélve, hogy a mezők többsége ugyanazokat az adatokat fogja tartalmazni.

Az adatbáziskezelők ilyen feladatok elvégzésére alkalmasak.A relációs adatbázis-kezelő rendszereket az adatokat sorokból és oszlopokból álló táblákként és a köztük fennálló kapcsolatokként tárolják.

/* Nem csak relációs adatbázis kezelők léteznek, de a jegyzet rövidsége miatt a többi sémára nem térünk ki részletesen. */

A cél tehát, hogy egy nagy táblázat helyett az adatokat több egybefüggő csoportban kezeljük, melyek között a kapcsolatokat a közös tulajdonságok adják.
A folyamatot, melynek során pedig az ilyen szerkezethez eljutunk normalizálásnak hívjuk.

Az SQL nyelv és a MySQL

Az SQL betűszó feloldása a Stuctured Query Language, ami szabványos lekérdező nyelvet jelent. A 70 es években az IBM alkotta meg és kereskedelmi forgalomba először 1979 -ben az Oracle rendszerében jelent meg. Később az ANSI szabványosította ezzel nagyban hozzájárult az SQL népszerűségéhez és a mai adatbázis rendszerek közötti hordozhatóság megteremtéséhez.

A MySQL a világon az egyik legelterjedtebb SQL alapú adatbáziskezelő-rendszer (RDBMS – Relational Database Management System), mely a már megszokott szerver-kliens architektúrát követi.. Előnyei a gyorsaság, a több szálú végrehajtás, a hordozhatóság, valamint, hogy a nyílt forráskódú változat a GPL betartása mellett ingyenes.
A többszálúság azt jelenti, hogy minden egyes új kapcsolat létrehozásakor létrejön egy új kiszolgáló folyamat, és ha az túlterhelné a kiszolgálót valamilyen hiba folytán, csak az adott folyamat áll le, nem az egész rendszer.
A hordozhatósághoz az is hozzájárul, hogy gyakorlatilag minden nyelven létezik hozzá felhasználói felület (PHP, Perl, C, C++, Java stb.).
Az nem ingyenes változatokról a http://www.mysql.com/ címen kaphatunk több információt, valamint a program részletes dokumentációja is megtalálható a http://www.mysql.com/doc/ címen.

Fogalmak

Egy MySQL rendszer több adatbázis is kezelhet egy időben, ezekre adatbázis nevekkel hivatkozhatunk.

A táblák: A táblák az adatbázisunk legnagyobb alkotórészei. Összefüggő adatstruktúrák tárolására használjuk. A MySQL rendszerekben egy tábla létrehozásakor három fájl jön létre. Ezek a tábla meghatározását, a tárolt adatokat és az adatokhoz tartozó indexeket tároljuk, ugyanakkor egységként kell kezelni őket.

ÁBRA (Adatbázis -> tábla)

Meg kell említeni még azt is, hogy a táblák többféle típusúak lehetnek. Az alapértelmezett a MyISAM típusú tábla, mi csak ilyen típusú táblákat fogunk használni.

A mezők: A mezők adják a tábla szerkezetét, és határozzák meg a benne tárolt adat típusát, annak hosszát, ezért a mezők létrehozása a tábla létrehozásával egy időben történik. Lényegében egy táblázatkezelő oszlopainak lehetne megfeleltetni őket. Egy táblában pontosan 3398 mező lehet maximálisan. Mivel azonban azt mondtuk, hogy kisebb egybefüggő részeket tárolunk egy táblában, ha ezt az értéket akár csak megközelíti is a mezőink száma, át kell gondolnunk az adatbázisunk szerkezetét.

Fontos dolog a tervezés közben a mezők típusainak megfelelő kiválasztása, mivel ez fogja meghatározni a később bevitt adatok típusát is. Fontos, hogy a mezők mérete ne legyen túl kicsi, valamint a „verébre nem lövünk ágyúval” szabály is fontos.

A recordok: A record egy bejegyzés a táblában. Azt az értéket tartalmazza, amelyet valójában tárolni szeretnénk.
A recordok lehetnek teljesen kitöltött, és kitöltetlen recordok, ami nem biztos, hogy üres recordot jelent. Mivel a mezőkhöz típustól függően alapértelmezett értéket állíthatunk be. Ha nem töltjük ki az adott mezőt, ez az alapértelmezett érték fog szerepelni a táblázatunkban.

A kulcsok: A kulcsok a táblák közötti kapcsolatok megteremtésének eszközei. A MySQL kétféle kulcstípust ismer, az elsődleges (un. primary key) és az egyedi (unique key) kulcsokat.
Az elsődleges kulcsnak minden táblában szerepelnie kell. Ez az az érték, amellyel a kapcsolatokat meghatározzuk. Alapértelmezés szerint ez az érték is egyedi, valamint a gyors működés miatt numerikus adatnak kell lennie. Nem tartalmazhat NULL értéket sem. (A NULL érték nem azonos a 0 értékkel, hiszen az előbbi azt jelenti, hogy a mező értéke nem tartalmaz adatot, míg az utóbbinál a mező értéke 0).
Az egyedi kulcsok megadása nem kötelező, de nagyon hasznos lehet, míg elsődleges kulcsból csak egy lehet táblákként, addig akárhány unique key létezhet.

Az indexek: Az indexek a könnyebb keresést segítik az adatbázisban. Az elsődleges kulcsokhoz automatikusan létrejön egy index tábla, ám mi is létrehozhatunk saját indexeket a tábla többi mezőjére, sőt akár több mezőt is indexelhetünk együttesen.
Az indexelt recordokon sokkal gyorsabban lefut egy kiválasztó lekérdezés, mint az indexeletlen recordok esetén, de mégis kerülni kell a felesleges indexek létrehozását. Hiszen, ha egy indexelt mezőt akarunk módosítani, akkor a az indextáblát is módosítanunk kell vele együtt, ami viszont sokkal időigényesebb feladat.

Általános elv nincs rá, hogy mikor hozzunk létre új indexeket, de mégis azt mondhatjuk, hogy ha a mindennapos használat során gyakran történik kiválasztó lekérdezés az adott mezőkombinációra akkor érdemes létrehozni egy indexet.

A lekérdezések az adatbázis-kezelő rendszerekkel történő kommunikáció elemei. igazából SQL nyelven írt parancsok. Fajtáikról és használatukról később még részletesen szót ejtünk.

A kapcsolatok: A tábláink között a kulcs mezők teremtenek kapcsolatot. A kapcsolatoknak több fajtája lehet:

Egy-egy kapcsolat (1 ->1): Az ilyen kapcsolatok esetében a kulcsmező csak egyszer szerepel a kapcsolt táblában. Ilyen esetre jó példa lehet, amikor a vállalat kiosztja a céges autókat az alkalmazottai között.

Egy-több kapcsolat (1->N): Az egy-több kapcsolat esetén a kulcsmezők a kapcsolódó táblában többször is megjelennek. Példa rá, ha a cég osztályaihoz rendeljük hozzá a dolgozókat, hiszen egy ember csak egy osztályon dolgozhat a cégnél.

Több-több kapcsolat (N ->M): A több-több kapcsolatok esetén a kapcsolt tábla kulcsértéke többször is megjelenhet, ám ugyanez a másik táblára nézve is igaz. A céges példánál maradva ilyen amikor a projekteket és az abban dolgozó embereket ábrázoljuk. Nyilván több ember is dolgozik ugyanazon a a projekten és egy ember nem csak egy projektben vehet részt. Az ilyen kapcsolatokat általában föl szokás bontani több 1 több kapcsolatra, egy köztes tábla bevezetésével.

A lefoglalt szavak: A lefoglalt szavak azokat a szavakat jelentik, amelyek jelentéssel bírnak az SQL nyelvben ezért nem használhatjuk őket azonosítóként.
Pl: SELECT, CREATE ALTER, stb

A mező típusai:

Egész:

- *tinyint:* 1 byteon tárolt egész szomok. Használhatunk Signed és Unsigned tinyint típust is.Alapértelmezett az előjeles.
- *Smallint:* 2 byte on tárolt egész szám.
- *mediumint* 4 byteon tárolt egész szám
- *int* 8 byteon tárolt egész szám.
- *bigint* 16 byte byteton tárolt egész.

Lebegőpontos:

- *float* (m,d) ahol m az összes számjegyek, míg d a tizedesek számát jelenti, Alapértelmezetten 10 számjegy és ebből 2 a tizedesek helye. Maximum 24 számjegyből állnak.
- *double* (m,d) a float típus nagytestvére.16,4 max 53

Itt jegyezném meg azt az irányelvet, hogy az adatbázisban csak akkor használjunk szám típusokat, ha azokkal valóban műveletet szeretnénk végezni.

Idő

- *DATE* 1001-01-01 től 9999-12-31 terjedő dátum.
- *DATETIME* A Date típus csak a dátumot tárolja, míg a datetime az időpontot is.
- *TIMESTAMP* 1970 jan 1 tól 2037 ig eltelt másodpercek száma. Tipikusan időbélyegnek használjuk,
- *YEAR* évet tárolja ha két számjeggyel, akkor 70 -69 ig (1970 – 2069) ha négy számjegy akkor 1901 tól 2155 ig

karakterlánc

- *char* 1-255 karakter tárolására használt típus. Alapértelmezetten 1 karakter hosszú. Fontos megjegyezni, hogy az ebben a típusban tárolt karakterlánc karaktereinek száma nem térhet el a deklarációban megadottaktól. Ez előnye és egyben hátránya is.
- *varchar* 1-255 karakter változó hosszusággal. Nincs alapértelmezett hosszúsága, ezért mindenképpen meg kell adni. Ez azt jelenti, hogy alkalmas a megadott hosszúságig bármilyen hosszú karaktersor tárolására.
- *blob vagy text*: 65535 karakterlánc. Nagyobb hosszúságú szövegek tárolására használhatjuk őket. A Blob típus Case Sensitive, azaz érzékeny a kis és nagy betűk közötti különbségekre, valamint alkalmas nagy mennyiségű bináris adat tárolására, például képek. Nem elterjedt, viszont hogy magát a képet tároljuk el adatbázisban.

- *Enum:* Felsorolás típus. A felsorolás elemeit mi adjuk meg és azokat az értékeket veheti fel a record. Hasznos lehet, ha valami csak bizonyos számú értéket vehet fel, például férfi/nő tavasz/nyár/ősz/tél.

Lekérdezések:

Az adatbázis motorral történő kommunikáció lekérdezések segítségével történik. A három alapvető fajtája a sima, (SELECT) az adatmódosító (INSER INTO, UPDATE, DELETE) és a tranzakciót támogató lekérdezések. Az utóbbival a tárgy keretein belül nem fogunk foglalkozni.
A lekérdezések egy utasításból és több egymást követő úgynevezett záradékból állnak. Ezek lehetnek:

- FROM: a lekérdezésben használt adatok helyét adhatjuk meg itt.
- WHERE: egy vagy több szűrőfeltétel megadására szolgál.
- GROUP BY: A lekérdezés eredményét rendezi sorba valamilyen szempont alapján. Általában arra szokás használni, hogy a végeredményben kapott sorokból a redundáns adatokat.
- HAVING: A GROUP BY záradék eredményét szűri hasonló módon mint a group by.
- ORDER BY Valamilyen record szerint növekvő, vagy csökkenő sorba a lekérdezés eredményét.

A záradékok egymás után futnak le és ebben a sorrendben. Ebből következik, hogy ORDER BY záradék mindig az utolsó és Group by -t nem követhet FROM.

Függvények:

A MySQL-nek mint a legtöbb programnyelvnek vannak beépített függvényei, melyek megkönnyítik az adatbázis-adminisztrátor dolgát a lekérdezéseknél. Ezek lehetnek matematikai műveletet végzők, karakterláncra vonatkozó függvények.

például:

Matematikai: sin(), cos(), sqrt(), pow().
Karakterláncra vonatkozó: trim (levágás), concat (összefűzés)

Léteznek még úgynevezett constraint-ek, melyek a mező létrehozásakor megadható, ezek a mező valid értékeinek szabályozására szolgálnak.

Ezeket ugyan a MySQL nem támogatja, de az SQL nyelv specifikációjában benne van, ezért ezeket a bejegyzéseket figyelmen kívül hagyja. Ebből kifolyólag a bevitt adatok ellenőrzését kliens oldalon kell megoldani.

Néhány példa:

adatbázis létrehozása:

```
create database proba;
```

Adatbázis használata:

```
use proba
```

Tábla létrehozása:

```
create table accounts(
id int not null auto_increment,
username varchar(16),
password varchar(16),
realname varchar(50),
primary key(id),
);
```

Adatbázis feltöltés:

```
insert into accounts values( 1, 'jampy', 'bela', 'Kovács
Ákos');
insert into accounts values( 2, 'gecko', 'jozsi', 'Sinko
Gergely');
```

Egyszerű lekérdezések:

```
select * from accounts where username='jampy';

select realname from accounts where id>1;

update accounts set password='feri' where username='gecko';
```