BME Villamosmérnöki és
Informatikai Kar

Távközlési és Médiainformatikai Tanszék
Adatbázisok Laboratórium

Elosztott adatbázis-kezelő rendszerek
Laboratóriumi gyakorlat

Oktatási segédanyag a Hálózatba kapcsolt adatbázisok tárgyhoz

1. változat

Egyetemi belső használatra!

Konzulens:

Erős Levente
Gajdos Sándor

Szerző:

Sajó Dániel

2014.

Bevezető

Az elosztott adatbázis-kezelő rendszer csomópontok összessége, amelyeken egy-egy számítógép üzemel saját CPU-val, háttértárral és adatbázis-kezelővel. A csomópontok egymással össze vannak kötve adatcserére alkalmas módon úgy, hogy az adatbázis-kezelők felhasználói csupán egyetlen, logikailag egységes adatbázist érzékelnek [1].

A hivatkozott Adatbázisok jegyzet külön fejezetet tartalmaz az elosztott adatbázisokról, melyben megtalálható a vonatkozó fogalmak pontos definíciója és a működés számos elemének leírása.

Az elosztott adatbázis-kezelő rendszerek az elosztott rendszerekhez hasonló előnyös tulajdonságokkal rendelkezhetnek [5], mint pl.:

- magas rendelkezésreállás
- értékes adatok hatékony védelme
- jobb skálázhatóság.

Egy elosztott adatbázis-kezelő rendszerrel lehetőség van a rendszer ugyanazon funkcióit ellátó egységeiből és/vagy az adatokból több példányt készíteni és ezeket akár földrajzilag akár nagy távolságra elhelyezni egymástól. Ez védhet egyes emberi hibák és/vagy természeti katasztrófák miatti szolgáltatás-kieséstől is.

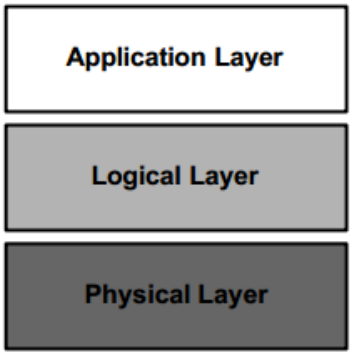
A gyakorlat során a MySQL Cluster [3] nevű elosztott adatbázis-kezelő rendszerrel lesz lehetőség megismerkedni, elsősorban annak rendelkezésreállására összpontosítva.

MySQL-ismertető

A MySQL Cluster a MySQL DBMS rendszerre [4], [6] épül. A MySQL egy széles körben használt nyílt forráskódú, relációs adatbázis-kezelő rendszer. Előnyös tulajdonságai (gyorsaság, platformfüggetlenség, költséghatékonyság, nagy adattömeg kezelése) miatt gyakran kínálkozik kedvező választási lehetőségként. Számos programozási nyelvnek nyújt egyedi illesztőfelületet (Application Programming Interface – API). Hatékonyan integrálható a teljesen nyílt forráskódú LAMP (Linux–Apache–MySQL–PHP) összeállításba, melynek részeként költséghatékony és egyszerűen beállítható megoldást ad dinamikus webhelyek szolgáltatására, akár elosztott információs rendszerek kialakítására.

A szoftver eredeti fejlesztője a svéd MySQL AB cég, melyet a Sun 2008 januárjában felvásárolt. Ezt követően a Sun-t felvásárolta az Oracle Corporation, így a MySQL is az Oracle tulajdonába került. Az új tulajdonos megtartotta a nyílt forráskódú fejlesztési modellt, tovább térítésért támogatást és bizonyos kiegészítő eszközöket is biztosít hozzá [4], [6].

A MySQL architektúrája három rétegre bontható: alkalmazási, logikai illetve fizikai (ld. az 1. ábra. ábrán).

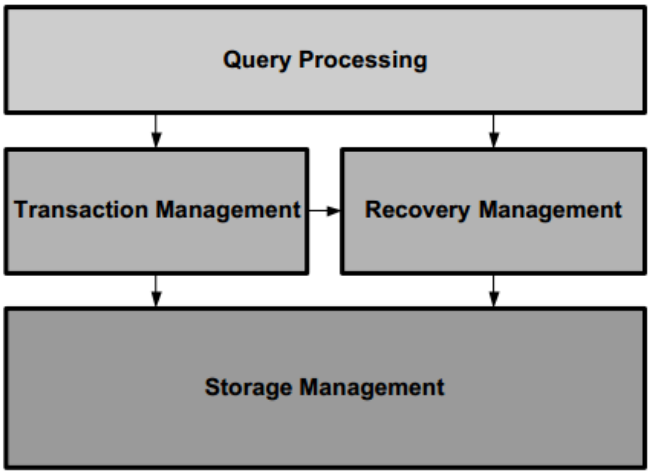


1. ábra: A MySQL magasszintű architektúrája [7]

Az **alkalmazási réteg** feladata, hogy biztosítsa az adatbázis logikai rétege és az adatbázis felhasználói közötti interakciót. A réteg által kiszolgált három felhasználótípus az adatbázis-adminisztrátor, a kliensprogramok valamint az adatbázist lekérdezésre használó felhasználók.

Az alkalmazási réteg alatt elhelyezkedő **logikai réteg** négy fő modulból áll (ld. 2. ábra ábra):

- Lekérdezés-feldolgozó (Query processor)
- Tranzakció-kezelő (Transaction management)
- Visszaállítás-kezelő (Recovery management)
- Tárkezelő (Storage management)



2. ábra: A MySQL logikai rétegének architektúrája [7]

A **lekérdezés-feldolgozó modul** feladata az adatbázis-kezelő számára értelmezhető SQL utasítások kinyerése az alkalmazási rétegből érkező kérésekből, a lekérdezések értelmezése, előfeldolgozása, optimalizálása (lásd [1]), a lekérdezéshez szükséges jogosultságok ellenőrzése és a lekérdezés végrehajtása.

A **tranzakció-kezelő modul** feladata, hogy az adatbázis tranzakciói (lásd [1]) végrehajtásának atomicitását (a tranzakció vagy minden adatbázis-utasítása végrehajtódik, vagy egyik sem) biztosítsa.

A **visszaállítás-kezelő modul** feladata azt biztosítani, hogy minden egyes adatbázis-művelet bekerüljön az adatbázis-kezelő naplójába, valamint hogy a rendszer összeomlása után az adatbázis visszaálljon az utolsó konzisztens állapotába. Ezen fogalmakról bővebben a [1] anyagban olvashatunk.

A **tárkezelő modul** feladata, hogy a tranzakció-kezelő és visszaállítás-kezelő modulokból érkező lekérdezéseket adatbázis-objektumokra (táblák) vonatkozó kérésekké alakítsa, és azok tartalmát kinyerje a háttértárról a memóriában lévő puffereken keresztül. A pufferek kezelése (memóriafoglalás és –felszabadítás) is a tárkezelő modul feladata.

A MySQL **fizikai rétege** kezeli az adatbázissal kapcsolatos összes adat tárolását, amelyek közül a legfontosabbak

- az **adatfájlok**, amelyek a felhasználók által az adatbázisban elhelyezett adatokat tárolják,
- az **adatszótár**, amely az adatbázis szerkezetével kapcsolatos metaadatokat tárolja,
- az **indexek**, amelyek a táblákban tárolt adatokhoz való gyorsabb hozzáférést biztosítják (bővebben lásd [1]),
- a **statisztikai adatok**, amelyeket a lekérdezés-optimalizáló használ annak érdekében, hogy az adott lekérdezést minél hatékonyabban tudja végrehajtani az adatbázis-kezelő, végül pedig
- a már korábban is említett **naplók**, amelyek az adatbázisban végrehajtott műveleteket tárolják, így segítve a visszaállítást egy rendszerösszeomlást követően.

A MySQL Cluster bemutatása

Ebben a fejezetben röviden bemutatjuk a MySQL Cluster működését, amelyről bővebben a [2] és [3] anyagokban olvashatunk.

A MySQL Cluster egy nyílt forráskódú elosztott relációs adatbázis-kezelő rendszer, melyet a népszerű MySQL adatbázis-kezelő egy különálló ágaként fejlesztenek. A fejlesztés eredménye egy fizikailag teljesen elosztott relációs adatbázis, mely magas rendelkezésreállást és redundanciát kínál.

Adatok elosztottsága kapcsán beszélhetünk replikációról, amikor az egyes adatrekordok több kiszolgálón is megtalálhatóak a jobb hibatűrés érdekében, illetve particionálásról, amikor az adatok az egyes kiszolgálókon nem többszörözve, hanem elosztva találhatók meg. Ezen két tulajdonság kombinálásával skálázható és hibatűrő információs rendszereket alakíthatunk ki a MySQL Cluster rendszerben; a MySQL Cluster adatbázis-kezelő rendszert úgy tervezték, hogy szükség esetén egyik elem meghibásodása se okozzon szolgáltatás-kiesést.

Memóriaorientált működés

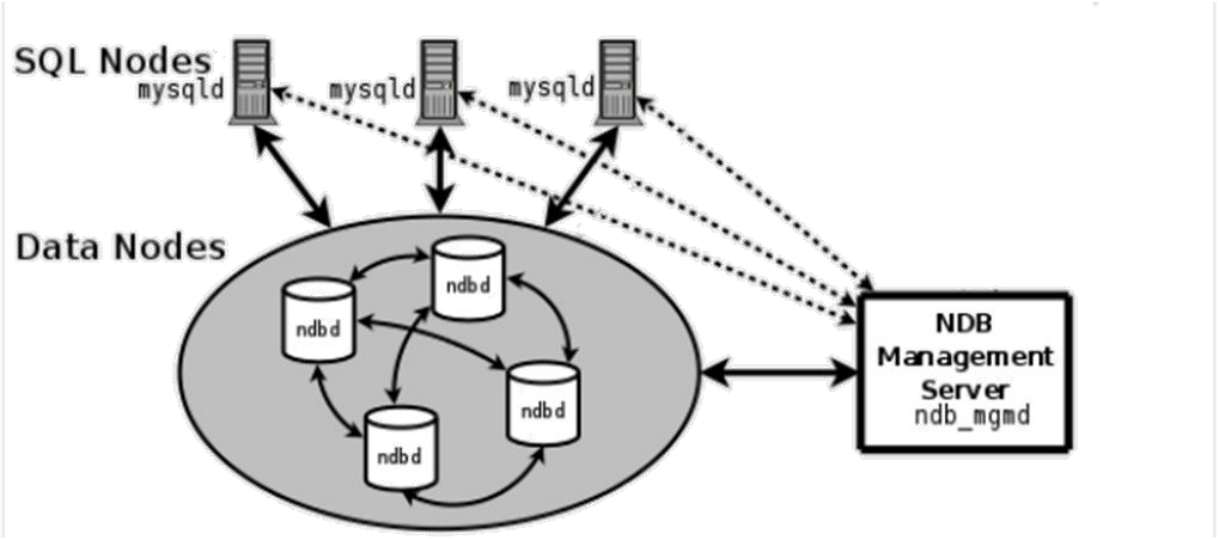
A MySQL Cluster legmeghatározóbb jellemzője, hogy miközben üzemel, az adatokat teljes mértékben a kiszolgáló (adatbázisszerver) memóriájában tárolja, és csak a tranzakciós naplókat írja

folytonosan a merevlemezre, hirtelen rendszerleállás esetén segítve a helyreállítást. Természetesen a kiszolgáló leállításakor nem vesznek el az adatok, olyankor a teljes adatbázis a merevlemezre íródik. A teljesen memóriaorientált működés előnye a nagyobb teljesítmény; a lekérdezések nagyságrendekkel gyorsabban futnak, ha az összes adat a memóriában áll rendelkezésre.

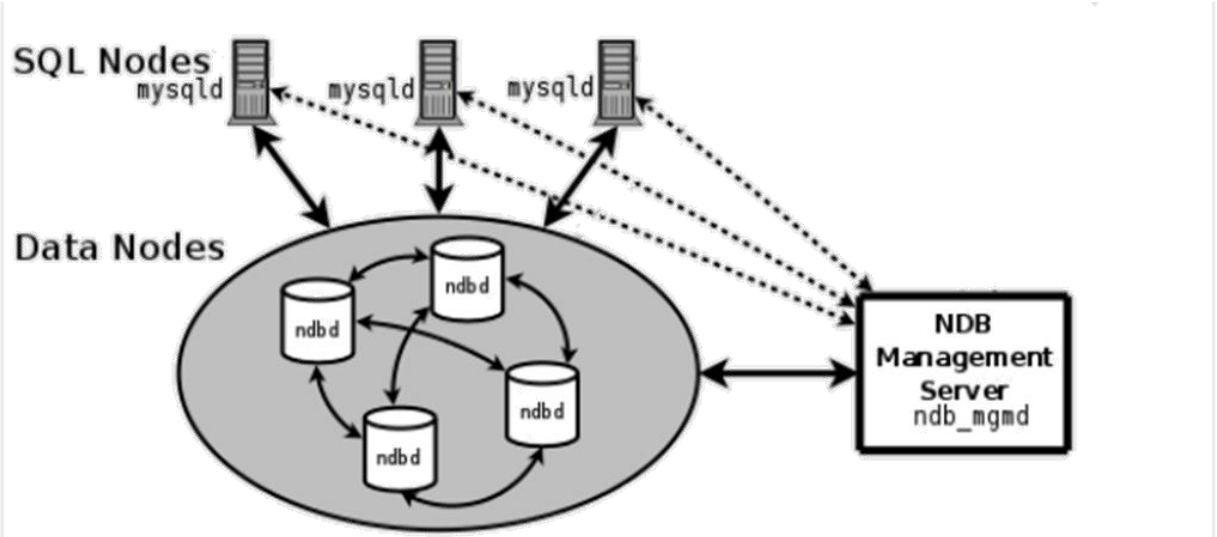
A megoldás hátránya, hogy a tárolható adatmennyiség a memória korlátozott kapacitása miatt messze elmarad a hagyományos adatbázisokhoz képest. Ezt felismerve, az újabb verziókban már lehetőség van adatok háttértáron történő tárolására is.

A rendszer alkotóelemei

Egy MySQL Cluster konfiguráció háromféle szoftverkomponensből áll: SQL, adat és menedzsment node-ból, ahogy a



3. ábra ábrán is látszik.



3. ábra: A MySQL Cluster felépítése [3]

SQL node

Az SQL kiszolgáló (SQL node) a hagyományos relációs adatbázis-kezelő rendszereknél megszokott módon viselkedik, feladata a kliensek felől érkező igények fogadása, az SQL lekérdezések

értelmezése és a feldolgozási terv előkészítése, melyet azután az adatkiszolgálók (data node-ok) felé továbbít. Az SQL node a gyakorlatban egy hagyományos MySQL kiszolgálóként látszik.

A hibatűrő működés eléréséhez illetve az adatműveletek esetleges párhuzamosítása érdekében legalább két SQL kiszolgáló telepítése ajánlott.

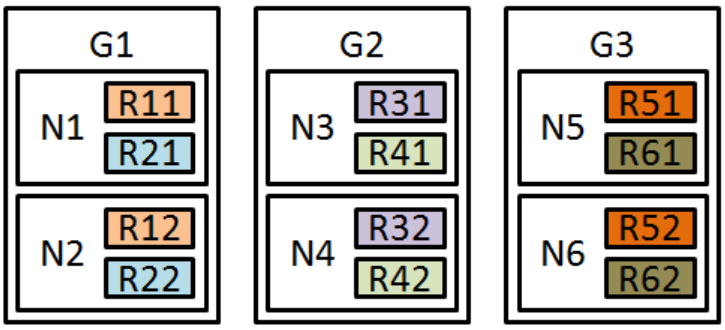
Data node

Az adatkiszolgálók (adatcsomópont vagy data node) feladata az adatok (adatrekordok) tényleges tárolása, amely szükség esetén az egyes kiszolgálók között elosztva, többszörözve és szinkronizálva valósul meg. Egy MySQL Cluster konfiguráció összeállítása során az előírt hibatűrési követelmények alapján választhatjuk meg, hogy milyen mértékű redundanciát és hány adatkiszolgálót alkalmazunk. Ezen változók alapján a menedzsment-kiszolgáló (ld. alább) úgynevezett **kiszolgálócsoportokba** (node group) szervezi az adatkiszolgálókat. A fentiek bővebben a következőket jelentik:

Partíciónak horizontális adatbázistábla-töredékek logikai szintű, a többi partícióval diszjunkt halmazát nevezzük. A labor során használt partícionálási eljárásnak köszönhetően alapesetben valamennyi partíción megtalálható valamennyi adatbázistábla valamekkora töredéke. Logikai szintű alatt azt értjük, hogy egy partíciót fizikailag több példányban tárolhatjuk, de ezen példányok logikailag ugyanazt az adathalmazt jelentik. Ezen példányokat az adott partíció **replikáinak** hívjuk. A MySQL Cluster-ben valamennyi partícióról ugyanannyi replikát tárolunk. Egy partíció replikáinak száma a **redundancia szintje.** Definíció szerint a rendszerben annyi partíció van, ahány adatkiszolgáló, és egy kiszolgálócsoportban a redundancia szintjével megegyező számú csomópont található. Egy partíció pontosan egy kiszolgálócsoporton tárolódik (azaz a partíciók replikái nem nyúlnak át kiszolgálócsoportokon, hanem egy partícióhoz tartozó valamennyi replika ugyanazon a kiszolgálócsoporton tárolódik). Továbbá egy kiszolgálócsoport pontosan annyi partíciót tárol, ahány csomópont van egy kiszolgálócsoportban. A kiszolgálócsoport valamennyi csomópontja a kiszolgálócsoporthoz tartozó valamennyi partíciónak tárolja egy-egy replikáját. A fentiekből következik, hogy a redundancia szintjét úgy kell megválasztanunk, hogy a rendszerben lévő csomópontok száma annak egész számú többszöröse legyen [2]. Tehát egy adott számú adatkiszolgálóból álló rendszer felkonfigurálása során a redundancia szintjét adjuk meg, amiből már következik a kiszolgálócsoportok száma és az egy csoportban lévő adatkiszolgálók száma is.

A fenti leírás összefüggéseire világít rá a 4. ábra. ábra egy példán keresztül. A példában 6 adatkiszolgáló van, és a redundancia szintjét 2-re választjuk. Előbbiből következik, hogy a rendszerben tárolt adathalmaz 6 partícióra oszlik, és 3 kiszolgálócsoport lesz, egyenként két adatkiszolgálóval (N1-6). A P1 és P2 partíciót a G1 kiszolgálócsoport, a P3 és P4 partíciót a G2 kiszolgálócsoport, a P5 és P6 partíciót pedig a G3 kiszolgálócsoport tárolja. A P1 partíció két replikáját R11 és R12 jelöli, hasonlóképpen jelöljük a többi partíció replikáit is. Kiszolgálócsoporton belül a csoporthoz rendelt valamennyi partíciót a csoport valamennyi adatkiszolgálója tárolja. Egy adott partíció replikáit az ábra azonos színnel jelöli.

[2] A node csoportok számának meghatározása: $\text{node csoportok száma} = \frac{\text{adat node-ok száma}}{\text{rendundancia értéke}}$.



4. ábra: Partíciók replikáinak elhelyezkedése a MySQL Cluster rendszeren

Ennek a felépítésnek köszönhetően az adatok mindaddig elérhetőek maradnak, amíg mindegyik csoportból legalább egy kiszolgáló működőképes, amit megfelelő redundanciaérték választásával biztosíthatunk. Az adatkiszolgálók száma egyrészt az adatbázis tárolási kapacitására van hatással, másrészt bizonyos esetekben nagyobb fokú párhuzamosítás is elérhető a partíciószámok növelésével. Futás közben is van lehetőség a klaszter új adatkiszolgálókkal való bővítésére, erre nézünk is példát a gyakorlaton.

Management node

A menedzsment-kiszolgáló feladata a klaszter logikai összefogása és a többi komponens monitorozása. A klaszter indításakor az egyes elemek a menedzsment-kiszolgálóra való kapcsolódással értesülnek az adatbázis kezdeti konfigurációjáról, és ezen keresztül találják meg egymást. A menedzsment-kiszolgálótól - megfelelő kliens használatával - a rendszer állapota is lekérdezhető (például a rendszert alkotó node-ok közül melyik üzemképes, az adat node-ok tárterületének mekkora része szabad stb.).

A menedzsment-kiszolgáló az adatok kezelésében közvetlenül nem vesz részt, csak a klaszter konfigurációjának változásakor (hálózati vagy adatbázis-beállítások változása, a klaszter bővítése új csoporttal) van szerepe, így a kiesése nem befolyásolja az adatbázis elérhetőségét. Emiatt ezt a komponenst nem szükséges duplikálni, bár a lehetőség megvan rá.

Egy MySQL Cluster adatbázis üzemeltetése során első lépésként fel kell térképezni, hogy az adatbázist későbbiekben felhasználó alkalmazás milyen fokú hibatűrést igényel és nagyságrendileg mekkora adatmennyiségetkezel. Ezen információk alapján kiválasztható a redundancia mértéke és a szükséges SQL és adatkiszolgálók száma.

A gyakorlatban használt konfigurációt, annak részletes leírását illetve az elvégzendő feladatokat a *feladatlapon* ismertetjük.

Felhasznált irodalom

[1] Gajdos Sándor: Adatbázisok jegyzet, 5. kiadás, BME Villamosmérnöki és Informatikai Kar, 2013

[2] Budai Péter István és Goldschmidt Balázs: 2013, Elosztott relációs adatbázis-kezelő rendszerek vizsgálata cloud környezetben, http://nws.niif.hu/ncd2013/docs/ehu/088.pdf , Letöltve: 2013. 09. 03.

[3] MySQL Cluster 5.6 dokumentáció, http://dev.mysql.com/doc/refman/5.6/en/ , Letöltve: 2014. 09. 06.

[4] MySQL Cluster 5.6 dokumentáció MySQL bemutató, http://dev.mysql.com/doc/refman/5.6/en/what-is-mysql.html, Letöltve: 2014.09.06.

[5] Wikipedia Distributed Database szócikk, http://en.wikipedia.org/wiki/Distributed_database , Letöltve: 2014. 09. 06.

[6] Wikipedia, MySQL szócikk, http://hu.wikipedia.org/wiki/MySQL, Letöltve: 2014.09.06.

[7] Ryan Bannon, Alvin Chin, Faryaaz Kassam, Andrew Roszko: MySQL Conceptual Architecture, http://ece.ut.ac.ir/dbrg/seminars/AdvancedDB/Fall%202008/hashemi/Project_MySQL_Benchmark/References/MySQL%20Conceptual%20Architecture.pdf, letöltve: 2014.09.17.