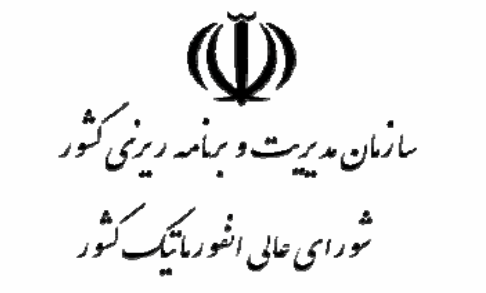
راهنمای سریع استفاده از پایگاه داده MySQL

گردآورندگان:
حمزه خزایی
محسن علی مومنی
محمدامین صابریان
علی مجدزاده

درحال ویرایش



طرح ملی نرم افزارهای آزاد/متن‌باز

(گنو/ لینوکس فارسی)

با همکاری مرکز تحقیقات و صنایع انفورماتیک

و هدایت علمی مرکز تحقیقاتی فناوری اطلاعات و ارتباطات پیشرفته

دانشگاه صنعتی شریف

و حمایت دبیرخانه شورای عالی اطلاع رسانی

فهرست مطالب

مقدمه

این کتابچه توسط تیم توسعه‌ای پروژه، افزودن امکانات فارسی به پایگاه داده MySQL تهیه شده است. سعی شده است که مطالب جمع‌آوری شده در این مجموعه، تا حد امکان مرتبط با موضوع پروژه باشد.

تقریبا تمامی مطالب و اطلاعاتی که برای کار کردن با پایگاه داده MySQL در مورد زبان فارسی نیاز است در این مقوله جمع‌آوری شده است. کلیه اطلاعات مربوط به پروژه فارسی سازی پایگاه داده MySQL در وب‌گاه این پروژه www.farsilinux.org موجود می‌باشد.

امیدواریم که این مجموعه راهنمایی مفید برای دوستان علاقه‌مند باشد. هرگونه انتقاد، نظر و پیشنهاد خود را به آدرس تماس E_Mail lab@rcii-ir.org ارسال فرمائید.

تیم توسعه پروژه MySQL
مرکز تحقیقات صنایع انفورماتیک

بخش اول
مروری بر پایگاه داده MySQL
۱-۱- سیستم مدیریت پایگاه داده MySQL

MySQL پرکاربردترین سیستم مدیریت پایگاه داده SQL متن باز است که توسط شرکت MySQL AB پشتیبانی می‌شود. MySQL AB شرکتی تجاری است که توسط توسعه دهندگان MySQL تاسیس شده است، و دومین شرکت تولید محصولات متن باز با یک مدل تجاری موفق است. آخرین اطلاعات در مورد MySQL و MYSQL AB را می‌توان از وب‌گاه http://www.mysql.com بدست آورد.

- **MySQL یک سیستم مدیریت پایگاه داده است.**

پایگاه داده مجموعه‌ای از داده‌های ساخت یافته است. برای اضافه کردن، دسترسی و پردازش داده‌های ذخیره شده در پایگاه داده، رایانه نیاز به یک سیستم مدیریت پایگاه داده مانند کارساز MySQL دارد. در رایانه‌هایی که روی حجم زیادی از داده‌ها کار می‌کنند، سیستم مدیریت پایگاه داده نقشی مرکزی را در محاسبات ایفا می‌کند.

- **MySQL یک سیستم مدیریت پایگاه داده رابطه‌ای است.**

پایگاه داده رابطه‌ای، داده را به جای ذخیره در یک مکان بزرگ، در جداول جداگانه ذخیره می‌کند که باعث افزایش سرعت و انعطاف پذیری می‌شود. SQL[1] بخشی از MySQL و زبان استاندارد MySQL است که توسط ANSI/ISO SQL تعریف شده است.

- **MySQL نرم‌افزاری متن باز است.**

منظور از متن باز این است که هر کسی می‌تواند آن را توسعه و تغییر دهد. هر کسی می‌تواند آن‌را از اینترنت بدون پرداخت هیچ وجهی دریافت نماید. هر فردی می‌تواند متن آن‌را مطابق با نیاز خود تغییر دهد. MySQL دارای یک GPL[2] است که مشخص می‌کند در چه جاهایی می‌توان و در چه جاهایی نمی‌توان از این ابزار استفاده نمود.

* کارساز پایگاه داده MySQL بسیار سریع، انعطاف پذیر و استفاده از آن آسان است.
* MySQL بصورت کارساز/کارخواه[3] استفاده می‌شود.
* زبانها و برنامه‌های کاربردی بسیاری امکان استفاده از پایگاه داده MySQL را فراهم می‌کنند.

۱-۱-۱- ویژگیهای اصلی پایگاه داده MySQL

از جمله ویژگیهای مهم پایگاه داده MySQL می‌توان به موارد زیر اشاره کرد:

- با زبانهای برنامه نویسی C و C++ نوشته شده است.
- با کامپایلرهای مختلف زیادی تست شده است.
- در بسترهای مختلف امکان استفاده از آن وجود دارد.
- از GNU Automake ، Autoconf، و Libtool برای قابلیت حمل استفاده می‌کند.

---

[1] Client

[2] GNU General Public License

[3] Structured Query Language

- دارای تعدادی API برای C, C++, Eiffel, Java, Perl, PHP, Python, Ruby و Tcl است.
- با استفاده از پردازشهای kernel، به طور کامل چند پردازه شده است و می‌تواند در صورت وجود از چندین CPU استفاده کند.
- از جداول درخت باینری خیلی سریع، با ایندکس فشرده شده استفاده می‌کند.
- به آسانی امکان استفاده از موتور ذخیره اضافی را می‌دهد.
- دارای یک سیستم تخصیص حافظه خیلی سریع است.
- در حافظه از جداول هش به عنوان جداول موقتی استفاده می‌کند.
- توابع SQL بکار برده شده از کتابخانه‌ای سریع، استفاده می‌کنند.
- کد MySQL به خوبی تست شده است.
- کارساز آن بصورت یک برنامه جداگانه برای استفاده در محیطهای شبکه شده کارساز/کارخواه موجود است.
- انواع فیلدها را پشتیبانی می‌کند.
- انواع رکورد با طول ثابت و طول متغیر را پشتیبانی می‌کند.
- مملو از تابع و عملگر برای استفاده در پرس و جوها است.
- GROUP BY و ORDER BY را بطور کامل پشتیبانی می‌کند.
- LEFT OUTER JOIN و RIGHT OUTER JOIN را هم برای SQL استاندارد و هم برای ساختار ODBC پشتیبانی می‌کند.
- آلیاس برای جدول و ستون، مورد نیاز SQL استاندارد را پشتیبانی می‌کند.
- در هنگام استفاده از عبارات ELETE, INSERT, REPLACE و UPDATE تعداد ردیفهایی را که تغییر کرده‌اند را برمی‌گرداند.
- دستور SHOW مخصوص MySQL بوده و امکان کسب اطلاعات در مورد پایگاه داده، جداول و ایندکسها را می‌دهد.
- توابع، جداول و ستونها در نام با هم تداخل ندارد.
- در یک پرس و جو می‌توان جداول پایگاه داده‌های مختلف را با هم ترکیب کرد.
- دارای یک سیستم کلمه عبور انعطاف پذیر و امن است.
- امکان بکارگیری پایگاه‌های داده بزرگ را می‌دهد. هر پایگاه داده می‌تواند ۵۰ میلیون رکورد داشته باشد، و کارساز MySQL می‌تواند ۶۰ هزار جدول به همراه ۵ میلیارد ردیف را پشتیبانی کند.
- بالای ۶۴ ایندکس در جدول می‌توان ایجاد کرد. هر ایندکس می‌تواند شامل ۱ تا ۱۶ ستون و یا بخشی از یک ستون باشد. حداکثر عرض ایندکس ۱۰۰۰ بایت می‌تواند باشد.
- کارخواه می تواند با استفاده از سوکتهای TCP/IP تحت هر بستری با کارساز ارتباط برقرار کند.
- در ویرایشهای بالاتر از ۴،۱ کارسازهای ویندوزی نیز امکان ارتباطهای shared-memory را فراهم می‌کنند.

- برنامه‌های کارخواه که از ODBC استفاده می‌کنند می‌توانند از Connector/ODBC (MyODBC) برای پشتیبانی از MySQL استفاده کنند. برای مثال شما می‌توانید از MS Access برای اتصال به کارساز MySQL استفاده کنید.
- برنامه‌های کارخواه جاوا که از JDBC استفاده می‌کنند برای پشتیبانی از MySQL می‌توانند از Connector/J استفاده کنند.
- کارساز می‌تواند پیغامهای خطا را با زبانهای مختلف برای کارخواهان نمایش دهد.
- مجموعه نویسه‌های مختلف را پشتیبانی می‌کند.
- همه داده‌ها با همان مجموعه نویسه انتخاب شده ذخیره می‌شوند.
- مرتب سازی بر اساس همان مجموعه نویسه انتخاب شده انجام می‌شود.
- کارساز MySQL امکان چک کردن، بهینه سازی و تعمیر جداول را می‌دهد، برای این کار می‌توان از دستور mysqlcheck استفاده کرد.
- راهنمای همه برنامه‌های MySQL را می‌توان با استفاده از help--یا ?- بدست آورد.
- هر جدول در MySQL 3.22 حداکثر ۴ گیگا بایت می‌توانست باشد. توسط موتور ذخیره MyISAM این فضا در MySQL 3.23 به ۸ میلیون ترابایت افزایش یافت. موتور ذخیره InnoDB امکان ذخیره جداول InnoDB را درون یک فضای جدولی تشکیل شده از چندین فایل مختلف را می‌دهد. حداکثر اندازه این فضای جدولی 64TB می‌تواند باشد. جداول زیر محدودیت اندازه جدول در هر سیستم عامل را مشخص می‌کند:

سیستم عامل	حداکثر اندازه فایل
Linux 2.2-Intel 32-bit	2GB (LFS: 4GB)
Linux 2.4	(using ext3 filesystem) 4TB
Solaris 9/10	16TB
NetWare w/NSS filesystem	8TB
win32 w / FAT/FAT32	2GB/4GB
win32 w / NTFS	2TB (possibly larger)
MacOS X w/ HFS+	2TB

جدول (۱-۱) محدودیت اندازه جدول در هر سیستم عامل

۱-۲- طرح توسعه MySQL

جدول زیر حاوی طرح توسعه MySQL در ویرایشهای مختلف است:

ویژگی	MySQL Series
Unions	4.0
Subqueries	4.1
R-trees	4.1 (for `MyISAM` tables)
Stored procedures	5.0

Views	5.0
Cursors	5.0
Foreign keys	5.1 (already implemented in 3.23 for `InnoDB`)
Triggers	5.0 and 5.1
Full outer join	5.1
Constraints	5.1

جدول(۱-۲) طرح توسعه MySQL

۱-۳- منابع اطلاعاتی MySQL

http://lists.mysql.com

فهرستهای پستی MySQL شامل موارد زیر است:

- announce : جهت اطلاع از ویرایشهای مختلف و برنامه‌های وابسته.
- mysql : لیستی مهم جهت توصیف عمومی MySQL
- bugs : جهت گزارش خطاها و مشکلات موجود در MySQL
- internals : برای افرادی که روی متن MySQL کار می‌کنند.
- mysqldoc : برای افرادی که روی مستندات MySQL کار می‌کنند.
- benchmarks : گفتگو در مورد جذابیت کارایی MySQL
- packagers : گفتگو در مورد امکان پخش و package کردن.
- java : در مورد کارساز MySQL و جاوا
- win32 : در مورد نرم افزار MySQL در سیستم عاملهای مایکروسافت
- myodbc : در مورد ارتباط MySQL با ODBC
- gui-tools : در مورد ابزارهای GUI برای MySQL
- cluster : گفتگو در مورد خوشه MySQL
- dotnet : در مورد کارساز MySQL و بستر .NET.
- plusplus : در مورد برنامه نویسی C++ API برای MySQL[1]
- perl : مواردی را که perl از MySQL پشتیبانی می‌کند.

برای گزارش اشکالات موجود در پایگاه داده MySQL می‌توان به پایگاه http://bugs.mysql.com/ رجوع کرد.

۱-۴- استانداردهای پشتیبانی شونده توسط MySQL

در این قسمت نحوه ارتباط MySQL با استاندارد ANSI/ISO SQL مشخص می‌شود.

* انتخاب حالت SQL

[1] Bugs

جهت انتخاب حالت پیش فرض به SQL ، mysqld را با گزینه "--sql-mode="modes اجرا کنید. البته بعد از اجرا هم می‌توانید با استفاده از دستور SET حالت آنرا تغییر دهید.

* اجرای MySQL به حالت ANCI

برای اینکار در هنگام اجرای mysqld از گزینه ansi-- استفاده شود.

* تفاوتهای MySQL با SQL استاندارد

– Subqueries : از MySQL 4.1 به بعد پشتیبانی می‌شود.

– SELECT INTO TABLE : آنرا بصورت INSERT INTO ... SELECT ... بکار می‌برد:

```
INSERT INTO tbl_temp2 (fld_id)
    SELECT tbl_temp1.fld_order_id
    FROM tbl_temp1 WHERE tbl_temp1.fld_order_id > 100;
```

- Transactions and Atomic Operations : کارساز MySQL (ویرایشهای بالاتر از 4.0) ترنزکشنها را با استفاده از موتور ذخیره InnoDB و BDB پشتیبانی می‌کند.

- Stored Procedures and Triggers: از ویرایش 5.0 پشتیبانی و استفاده می‌شوند.

– کلیدهای خارجی : در کارساز MySQL 3.23.44 و ویرایشهای بالاتر آن استفاده می‌شوند.

– نماها : در ویرایش 5.0 از پایگاه داده MySQL پشتیبانی می‌شوند.

- '--' در آغاز یک فرمان : بعضی از پایگاه داده‌های SQL از '--' برای شروع فرامین استفاده می‌کنند، MySQL از '#' برای شروع فرامین استفاده می کند.

۱-۵- پایگاه داده MySQL و امکانات فارسی

ویرایشهای قبل از MySQL 4.1.3 برای ذخیره، نمایش و مرتب سازی داده‌های فارسی از قالب CP1256 استفاده می کردند که این قالب مخصوص زبان عربی است، لذا در ذخیره ، نمایش و مرتب سازی داده‌های فارسی به خصوص داده‌هایی که در آنها از حروف (گ چ پ ژ) استفاده می‌شود، با مشکل مواجه می‌شدند. ولی در ویرایشهای بعد از این نسخه به دلیل اضافه شده امکانات فارسی تحت فالب UTF8 این مشکلات (ذخیره، نمایش، مرتب‌ساری، جستجوی داده‌های فارسی) برطرف شد.

به منظور تبدیل پایگاه داده‌ها با فرمت cp1256 به یونی‌کد، ابزاری با نام CP2UTF تهیه شده، که با نصب MySQL ویرایش جدید، و اجرای این ابزار، نام پایگاه داده مربوطه را دریافت و آنرا به قالب UTF8 تبدیل می‌کند.

فعالیتهایی نیز در جهت اضافه کردن امکانات مربوط به تقویم شمسی (توابع adddate ، subdate و ...) به پایگاه داده MySQL صورت گرفته است و مکاتبات لازم با گردانندگان MySQL در جهت ثبت این امکانات در حال انجام است.

بخش دوم
نصب پایگاه داده MySQL
۲-۱- اطلاعات کلی در مورد نصب
قبل از نصب MySQL، اقدامات زیر را انجام دهید:
- آیا MySQL از قبل در سیستم نصب و اجرا نشده است؟
- یک توزیع برای نصب انتخاب شود.
- توزیع مورد نظر دریافت و صحت دریافت بررسی شود.

۲-۱-۱- سیستم عاملهای پشتیبانی شونده توسط MySQL
سیستم عامهایی که می‌توان MySQL را در انها اجرا کد عبارتند از:

- AIX 4.x,5. x with native threads.
- Amiga.
- BSDI 2.x with the MIT-pthreads package.
- BSDI 3.0, 3.1 and 4.x with native threads. .
- Digital Unix 4.x with native threads. .
- FreeBSD 2.x with the MIT-pthreads package. .
- FreeBSD 3.x and 4.x with native threads. .
- FreeBSD 4.x with LinuxThreads. .
- HP-UX 10.20 with the DCE threads or the MIT-pthreads package.
- HP-UX 11.x with the native threads .
- Linux 2.0+ with LinuxThreads 0.7.1+ or `glibc` 2.0.7+ for various CPU architectures.
- Mac OS X.
- NetBSD 1.3/1.4 Intel and NetBSD 1.3 Alpha (requires GNU make).
- Novell NetWare 6.0. .
- OpenBSD > 2.5 with native threads. OpenBSD < 2.5 with the MIT-pthreads package.
- OS/2 Warp 3, FixPack 29 and OS/2 Warp 4, FixPack 4. .
- SCO OpenServer with a recent port of the FSU Pthreads package.
- SCO UnixWare 7.1.x. .
- SGI Irix 6.x with native threads. .

- Solaris 2.5 and above with native threads on SPARC and x86. .
- SunOS4. x with the MIT-pthreads package. .
- Tru64 Unix. .
- Windows 9x, Me, NT, 2000, XP, and 2003. .

۲-۱-۲- چه نسخه‌ای از MySQL برای نصب انتخاب شود.

قبل از هر چیز باید مشخص شود که برای استفاده نسخه‌ای پایدار می‌خواهید یا نسخه‌ای برای توسعه. به عبارت دیگر نسخه‌ای برای استفاده می‌خواهید و یا اینکه نسخه‌ای را به منظور توسعه دادن(اضافه کردن قابلیتهایی به آن) می‌خواهید.

نسخه‌های که در حال حاضر موجود هستند عبارتند از:

* MySQL 5.0 : این نسخه برای توسعه دهندگان است و ویرایش آلفای آن موجود است و دوران تست خود را بسر می‌برد.
* MySQL 4.1 : این نسخه به تازگی منتشر شده است و ویژگی جدیدی به آن اضافه نخواهد شد.
* MySQL 4.0 : این نسخه قبلا منتشر شده و امکان اضافه کردن ویژگی‌های جدید را ندارد.

اگر شما از یک ویرایش قدیمی استفاده می‌کنید و می‌خواهید آن‌را بدون اعمال تغییرات ارتقا دهید، باید آنرا به آخرین نسخه از همان ویرایش ارتقا دهید (اولین رقم سمت چپ آنها باید یکسان باشد).

الگوی نامگذاری در ویرایشهای مختلف بر اساس موارد زیر است. مثال برای mysql-4.1.2-alpha :

– عدد اول (۴) : ویرایش اصلی و توصیف کننده فرمت فایل است که همه نسخه‌های منتشر شده ۴ همان فرمت فایل را دارند.

– عدد دوم (۱): سطح انتشار۱ را مشخص می‌کند. ویرایش اصلی و سطح انتشار با هم تشکیل دهنده شماره سری انتشار است.

– عدد سوم (۲) : شماره ویرایش در سری انتشار است که برای هر انتشار جدید، یکی اضافه می‌شود.

در هر به روز رسانی عدد آخر (رقم سوم) یکی اضافه می‌شود. با اضافه شدن یک ویژگی مهم یا ناسازگاری با ویرایش قبلی، عدد دوم یکی اضافه می‌شود. و با تغییر فرمت فایل عدد اول یکی اضافه می‌شود.

نام نسخه‌ها دارای پسوندی هستند که نشان دهنده سطح پایداری آنها است. پسوندهای ممکن شامل موارد زیر هستند:

- alpha : نشان دهنده اضافه شدن کدهای جدید که بطور کامل تست نشده‌اند است.

[1] Release

• beta : همه کد تست شده است و هیچ ویژگی جدیدی به کد قبلی (alpha) اضافه نشده است.

• gama : همان مورد بتا که به منظور صحت عملکرد مورد بررسی قرار گرفته و قسمتهای مورد نیاز تعمیر شده است.

تعدادی از ویژگیهای اضافی که می‌توان برای پیکربندی mysqld استفاده کرد و در حالت عادی استفاده نشده‌اند، عبارتند از:

```
--with-innodb (default for MySQL 4.0 and up)
--with-berkeley-db (not available on all platforms)
--with-raid
--with-libwrap
--with-named-z-libs (this is done for some of the
binaries)
--with-debug[=full]
```

در به روز رسانی MySQL از سیاستهای زیر استفاده می‌شود:

– در هر سری، هر انتشار رقم آخرش نسبت به انتشار قبلی یکی بیشتر می‌شود.

– در هر سال یک یا دو مرتبه نسخه اصلی تولید می‌شود.

– رفع باگ برای هر انتشار در حدود ۴ الی ۸ هفته به طول می‌انجامد.

– در بعضی بسترها برای ویرایشهای اصلی توزیع باینری تولید می‌شود.

– در صورت که به هر دلیلی باگی بزرگ در یک ویرایش پیدا شود، ویرایشی جدید در حداقل زمان ممکن انتشار خواهد یافت.

برای دریافت MySQL به پایگاه http://dev.mysql.com/downloads مراجعه شود. بعد از دانلود به منظور اطمینان از صحت دانلود قبل از نصب می‌توان از md5 یا gnupg استفاده کرد. برای استفاده از md5 بصورت زیر عمل شود:

```
shell> md5sum package_name
```

مثال:

```
shell> md5sum mysql-standard-4.0.17-pc-linux-i686.tar.gz
60f5fe969d61c8f82e4f7f62657e1f06  mysql-standard-4.0.17-
pc-linux-i686.tar.gz
```

نتیجه دستور باید با مورد مشخص شده در صفحه دانلود مطابقت داشته باشد.

بررسی امضا با استفاده از gnupg مناسب‌تر است ولی نیاز به کار بیشتری دارد که از توضیح آن صرف نظر می‌کنیم.

شاخه‌هایی که با نصب یک توزیع متن در یونیکس در مسیر انتخابی شما ایجاد می‌شوند، عبارتند از:

محتوای شاخه	شاخه
برنامه‌های کارخواه و اسکریپتها	`bin'
شامل هدر فایلها	`include/mysql'
مستندات با فرمت info	`info'

کتابخانه‌ها	`lib/mysql'
کارساز mysqld	`libexec'
فایل پیغام خطا	`share/mysql'
تست و محک crash-me	`sql-bench'
فایلهای پایگاه داده و ثبت شده	`var'

جدول (۲-۱) شاخه‌های ایجاد شده با نصب MySQL

۲-۲- نصب پایگاه داده MySQL در لینوکس

پیشنهاد می‌شود برای نصب MySQL از بسته‌های RPM استفاده شود. بسته‌های RPM حاضر در Suse Linux 7.3 ایجاد شده‌اند، ولی بر روی اکثر ویرایشهای لینوکس که rpm را پشتیبانی و از glibc استفاده می‌کنند، قابل استفاده هستند.

برای مشاهده فایلهای یک بسته RPM (بطور مثال یک MySQL-server RPM) دستور زیر را اجرا نمایید:

```
shell> rpm -qpl MySQL-server-VERSION.i386.rpm
```

برای نصب در حالت استاندارد، از دستورات زیر استفاده نمایید:

```
shell> rpm -i MySQL-server-VERSION.i386.rpm
shell> rpm -i MySQL-client-VERSION.i386.rpm
```

برای نصب تنها در سمت کارخواه بصورت زیر رفتار نمایید:

```
shell> rpm -i MySQL-client-VERSION.i386.rpm
```

۲-۳- نصب پایگاه داده MySQL با استفاده از توزیع متن

قبل از اینکه MySQL را با استفاده از متن نصب کنید از مطابقت آن با سیستم عامل مطمئن شوید.

برای نصب MySQL با استفاده از متن به ابزارهای زیر نیاز دارید:

- GNU gunzip : برای غیر فشرده کردن توزیع مربوطه.
- ابزار tar به منظور بازگشایی توزیع مورد نظر.
- یک کامپایلر C++ ، gcc 2.95.2 یا بعد از آن، egcs 1.0.2 یا بعد از آن
- یک برنامه make مناسب مانند GNU make

فرامین اصلی برای نصب MySQL با استفاده از توزیع متن، عبارتند از:

```
shell> groupadd mysql
shell> useradd -g mysql mysql
shell> gunzip < mysql-VERSION.tar.gz | tar -xvf -
shell> cd mysql-VERSION
shell> ./configure --prefix=/usr/local/mysql
shell> make
shell> make install
shell> cp support-files/my-medium.cnf /etc/my.cnf
shell> cd /usr/local/mysql
shell> bin/mysql_install_db --user=mysql
shell> chown -R root  .
```

```
shell> chown -R mysql var
shell> chgrp -R mysql .
shell> bin/mysqld_safe --user=mysql &
```

در ویرایشهای قبل از 4.0 در خط آخر بجای bin/mysqld_safe از bin/safe_mysqld استفاده می‌شود.

جزئیات بیشتر در مورد خطوط بالا در زیر مشاهده می‌شود:

۱- ایجاد یک گروه و کاربر برای اجرای mysqld

```
shell> groupadd mysql
shell> useradd -g mysql mysql
```

۲- وارد شاخه‌ای شوید که می‌خواهید توزیع مورد نظر در آنجا بازگشایی نمایید.

۳- فایل حاوی متن کد را unzip کنید.

```
shell> gunzip < /path/to/mysql-VERSION.tar.gz | tar xvf
-
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
```

۴- وارد فولدر unzip شده شوید.

```
shell> cd mysql-VERSION
```

۵- دستور پیکربندی اجرا شود.

```
shell> ./configure --prefix=/usr/local/mysql
```

در هنگام پیکربندی شما می توانید چندین گزینه را انتخاب نمایید، برای مشاهده آنها قبل از پیکربندی confidure --help را اجرا نمایید.

۶- اکنون زمان کامپایل است.

```
shell> make
```

۷- مرحله نصب در سیستم به صورت زیر انجام می‌شود:

```
shell> make install
```

برای تنظیم یک فایل اختیاری، از الگوهای موجود در شاخه 'support-files` استفاده نمایید. بطور مثال:

```
1.  shell> cp support-files/my-medium.cnf /etc/my.cnf
```

۸- وارد شاخه نصب شوید.

```
shell> cd /usr/local/mysql
```

۹- اگر MySQL را قبلا نصب نکرده‌اید، باید ابتدا قبل از اجرا جداول مورد استفاده MySQL را نصب نمایید.

```
shell> bin/mysql_install_db --user=mysql
```

۱۰- مالک برنامه را به root و مالک شاخه data را به کاربری که قصد اجرای mysqld را دارد، تغییر داده و گروه را به گروه mysql تغییر دهید. با فرض اینکه موقعیت نصب شاخه (`/usr/local/mysql') باشد. خواهیم داشت:

```
shell> chown -R root  .
shell> chown -R mysql var
shell> chgrp -R mysql .
```

۱۱- برای اجرای MySQL در هنگام بوت شدن سیستم support-files/mysql.server را در فایل startup سیستم کپی نمایید.

گزینه‌های قابل استفاده در هنگام پیکربندی عبارتند از:

```
shell> ./configure -help
```

دستور بالا گزینه‌های قابل استفاده را نمایش می‌دهد.

• برای کامپایل فایلهای کتابخانه‌ای و برنامه‌های client به تنهایی (بدون کارساز):

```
shell> ./configure --without-server
```

• اگر می‌خواهید کتابخانه MySQL (libmysqld.a) در آن جاسازی شود:

```
shell> ./configure --with-embedded-server
```

• اگر می خواهید فایلها و شاخه‌های نصب در مسیر دلخواه شما قرار گیرد(مثلا /usr/local/mysql):

```
shell> ./configure --prefix=/usr/local/mysql
```

با استفاده از prefix مسیر نصب مشخص می‌شود.

• اگر از یونیکس استفاده می‌کنید و می‌خواهید فایل سوکت MySQL در مسیر دلخواه شما قرار گیرد:

```
shell> ./configure \
          --with-unix-socket-
path=/usr/local/mysql/tmp/mysql.sock
```

• اگر قصد کامپایل برنامه‌های لینک شده ثابت را دارید:

```
shell> ./configure --with-client-ldflags=-all-static \
          --with-mysqld-ldflags=-all-static
```

• اگر از gcc استفاده می‌کنید و libg++ یا libstdc++ را نصب نکرده‌اید، می‌توانید به configure بگویید که از gcc به عنوان کامپایلر استفاده نماید:

```
shell> CC=gcc CXX=gcc ./configure
```

• بطور پیش‌فرض از latin1 (ISO-8859-1) به عنوان مجموعه نویسه استفاده می‌شود، برای تغییر آن:

```
shell> ./configure --with-charset=CHARSET
```

و برای تغییر تابع تطبیق پیش فرض (latin1_swedish_ci) خواهیم داشت:

```
shell> ./configure --with-collation=COLLATION
```

• جهت اضافه کردن امکان debug کد.

```
shell> ./configure --with-debug
```

• اگر برنامه‌های client از thread استفاده می‌کنند، شما باید ویرایشی از MySQL که این قابلیت را پشتیبانی می‌کند، کامپابل کنید، که گزینه قابل استفاده در پیکربندی --enable-thread-safe client است.

در صورتی که در هنگام پیکربندی با مشکلی مواجه شدید و خواستید عملیات پیکربندی را دوباره انجام دهید. قبل از آن دو دستور زیر را اجرا نمایید:

```
shell> rm config.cache
shell> make clean
```

– اطلاعات مربوط به پیکربندی قبلی که با مشکل مواجه شده بود، در فایل config.cache ذخیره شده است، لذا برای پیکربندی دوباره این فایل باید پاک شود.
– هر دفعه که پیکربندی می کنید، make را باید دوباره اجرا نمایید، ولی اگر بخواهید فایلهای object ایجاد شده قبلی را پاک نمایید، برای این کار از دستور دوم (make clean) استفاده می‌شود.

برای تنظیم کلمه عبور به منظور افزایش امنیت در لینوکس می توان بصورت زیر عمل کرد:

```
shell> mysql -u root
mysql> SET PASSWORD FOR ''@'localhost' =
PASSWORD('newpwd');
mysql> SET PASSWORD FOR ''@'host_name' =
PASSWORD('newpwd');
```

در دستور دوم بجای *host_name* نام کارساز میزبان را قرار دهید.
با استفاده از دستور UPDATE نیز می‌توان کلمه عبور را تنظیم کرد:

```
shell> mysql -u root
mysql> UPDATE mysql.user SET Password =
PASSWORD('newpwd')
    ->     WHERE User = '';
mysql> FLUSH PRIVILEGES;
```

۲-۳-۱- مشکلات اجرای mysql_install_db

هدف از اجرای mysql_install_db ایجاد جداول مورد استفاده MySQL است. در صورت وجود این جداول آنها را دوباره نویسی نمی‌کند و بر روی داده‌های موجود تاثیر نمی‌گذارد. اگر مایل به ایجاد دوباره این جداول هستید، ابتدا اگر کارساز mysqld در حالت اجرا است، آنرا متوقف کرده و سپس شاخه mysql موجود در شاخه data را تغییر نام دهید و سپس mysql_install_db را اجرا نمایید.
به طور مثال:

```
shell> mv mysql-data-directory/mysql mysql-data-
directory/mysql-old
shell> mysql_install_db --user=mysql
```

مشکلاتی که ممکن است در هنگام اجرای mysql_install_db رخ دهند عبارتند از:

- **mysql_install_db doesn't install the grant tables**

mysql_install_db ممکن است برای ایجاد جداول جدید با مشکل مواجه شود و با نمایش پیام زیر پایان یابد:

```
Starting mysqld daemon with databases from XXXXXX
mysqld ended
```

در این حالت باید فایل log به طور دقیق بررسی شود. اگر متوجه دلیل خطا نشدید، می‌توانید آنرا به عنوان یک اشکال برای گردانندگان MySQL ارسال نمایید. برای این کار به فهرستهای پستی مشخص شده در ۱-۳ مراجعه نمایید.

- **There is already a mysqld process running**

این پیغام زمانی نمایش داده می‌شود، که کارساز mysqld در حالت اجرا است و در این حالت نیازی به اجرای mysql_install_db ندارید، نیازی نیست این دستور را هر دفعه اجرا بلکه، بلکه اجرای آن همان دفعه اول بعد از نصب کفایت می کند.

- **Installing a second mysqld server doesn't work when one server is running**

زمانی اتفاق می‌افتد که یک MySQL نصب شده وجود دارد، ولی می‌خواهید آنرا در یک موقعیت جدید (مثلا برای تست) نیز نصب کنید. برای اجرای چندین کارساز به بخش مربوطه مراجعه شود.

- **You don't have write access to `/tmp'**

از آنجایی که فایل سوکت به طور پیش فرض در این مسیر(/tmp) نصب می‌شود، لذا در صورت عدم مجوز دسترسی شما به این فایل، با این پیام مواجه خواهید شد. برای رفع این مشکل می‌توانید از شاخه‌ای دیگر برای ذخیره فایل سوکت استفاده کنید، برای این کار قبل از اجرای mysql_install_db و یا mysqld ، فرامین زیر را اجرا نمایید:

```
shell> TMPDIR=/some_tmp_dir/
shell> MYSQL_UNIX_PORT=/some_tmp_dir/mysql.sock
shell> export TMPDIR MYSQL_UNIX_PORT
```

بعد از اجرا فرامین فوق شما باید بتوانید با استفاده از فرامین زیر mysql_install_db و کارساز را اجرا کنید:

```
shell> bin/mysql_install_db --user=mysql
shell> bin/mysqld_safe --user=mysql &
```

mysql_install_db را از شاخه script نیز می‌توانید اجرا کرد.

برای اجرا و متوقف ساختن mysqld می‌توان به روش زیر عمل کرد:

```
shell> mysql.server start
shell> mysql.server stop
```

تعدادی از تنظیمات مربوط به mysql.server را می توانید در فایل /etc/my.cnf مشاهده کرده و مواردی را نیز می‌توانید به آن اضافه کنید. از جمله محتویات این فایل می‌توان به موارد زیر اشاره کرد:

```
[mysqld]
datadir=/usr/local/mysql/var
socket=/var/tmp/mysql.sock
port=3306
user=mysql

[mysql.server]
basedir=/usr/local/mysql
```

بخش سوم
آموزش MySQL

در این بخش کاربران پایگاه داده MySQL با نحوه ایجاد و استفاده از پایگاه داده MySQL آشنا می‌شوند.

فرض کرده‌ایم که پایگاه داده MySQL از قبل نصب شده و آماده جهت استفاده است.

۳-۱- ارتباط و قطع ارتباط با کارساز

معمولا برای ارتباط با پایگاه داده MySQL نیاز به یک نام کاربری و کلمه عبور دارید و اگر کارساز آن بر روی سیستمی دیگر در حال اجرا است، اطلاع از نام میزبان نیز ضروری است. جهت آگاهی از نام کاربری، کلمه عبور، و نام میزبان با مدیر سیستم ارتباط برقرار نمایید. بعد از آن بصورت زیر می‌توانید با کارساز ارتباط برقرار کنید:

```
shell> mysql -h host -u user -p
Enter password: ********
```

host نام میزبانی است که کارساز MySQL در آن در حالت اجرا است و user نام کاربری شما جهت استفاده از MySQL است. و ******** کلمه عبوری است که جهت ارتباط با کارساز MySQL وارد می‌کنید.

در صورت صحت اطلاعات ورودی پیغامی به صورت زیر مشاهده خواهید کرد:

```
Welcome to the MySQL monitor.  Commands end with ; or
\g.
Your MySQL connection id is 25338 to server version:
4.0.14-log

Type 'help;' or '\h' for help. Type '\c' to clear the
buffer.

mysql>
```

mysql> نشان دهنده آماده بودن جهت وارد کردن فرامین است.

بعضی از MySQLهای نصب شده به کاربران اجازه ارتباط به صورت دورست(ارتباط با کارساز MySQL از طریق سیستمهای دیگر) را می‌دهند، در این حالت می‌توان mysql را بدون هیچ گزینه‌ای اجرا کرد.

```
shell> mysql
```

بعد از ارتباط موفقیت آمیز، برای قطع ارتباط می‌توان از QUIT (یا q/) استفاده کرد:

```
mysql> QUIT
Bye
```

در لینوکس می‌توان ارتباط را با فشردن دکمه‌های Control-D نیز قطع کرد.

۳-۲- وارد کردن پرس و جوها

بعد از اطمینان از ارتباط با کارساز(با روش گفته شده در قسمت قبل)، می‌توان از فرامین مختلف که چندین نمونه از آن در پایین مشاهده می‌شود، استفاده کرد:

```
mysql> SELECT VERSION(), CURRENT_DATE;
+--------------+--------------+
| VERSION()    | CURRENT_DATE |
+--------------+--------------+
| 3.22.20a-log | 1999-03-19   |
+--------------+--------------+
1 row in set (0.01 sec)
mysql>
```

فرمان بالا بیان کننده موارد زیر در MySQL است:

۱- فرامین به طور نرمال با سمی‌کولون پایان می یابند.

۲- با وارد کردن یک فرمان، mysql آن‌را برای اجرا شدن به کارساز ارسال می‌کند و نتیجه را نمایش می‌دهد.

۳- خروجی در جداولی از سطر و ستون نمایش داده می‌شود. ردیف اول برچسب ستون‌ها است و سایر ردیفها خروجی دستور وارد شده هستند.

۴- mysql تعداد سطرهای خروجی و زمان اجرای فرمان را نشان می‌دهد، که با استفاده از آن کارایی کارساز مشخص می‌شود.

کلمه‌های کلیدی را می‌توان به هر حالتی (کوچک یا بزرگ) نوشت:

```
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
```

از mysql می‌توان به عنوان یک ماشین حساب ساده استفاده کرد:

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+------------+---------+
| SIN(PI()/4) | (4+1)*5 |
+-------------+---------+
|    0.707107 |      25 |
+-------------+---------+
```

می‌توان چندین عبارت را نیز در یک خط وارد کرد که در آخر هر عبارت یک سمی‌کولون باید گذاشته شود:

```
mysql> SELECT VERSION(); SELECT NOW();
+--------------+
| VERSION()    |
+--------------+
| 3.22.20a-log |
+--------------+

+---------------------+
| NOW()               |
+---------------------+
| 1999-03-19 00:15:33 |
+---------------------+
```

لزومی ندارد دستور در یک خط نوشته شود، پایان دستور جایی است که سمی‌کولون گذاشته شده است:

```
mysql> SELECT
    -> USER()
    -> ,
    -> CURRENT_DATE;
+--------------------+--------------+
| USER()             | CURRENT_DATE |
+--------------------+--------------+
| joesmith@localhost | 1999-03-18   |
+--------------------+--------------+
```

بعد از اولین خط ، خط فرمان از <mysql به < تغییر حالت می‌دهد.

اگر در حین وارد کردن یک دستور، از اجرای آن منصرف شدید، از :c\ استفاده نمایید:

```
mysql> SELECT
    -> USER()
```

```
    -> \c
mysql>
```

جدول زیر حاوی خط فرمانهایی است که ممکن است در mysql مشاهده نمایید:

خط فرمان	معنی آن
mysql>	آماده برای فرمان جدید
->	در یک فرمان چند خطی، منتظر برای خط بعدی
'>	شروع شده (`'') منتظر برای خط بعدی، برای دستوری کـه رشـته در آن با علامت نقل قول واحد
">	شروع شده(`"') منتظر برای خط بعدی، برای دستوری که رشته در آن با علامت نقل قول دوتایی
`>	شروع شده (``') منتظر برای خط بعدی، در دستوری که تعریف آن با یک بک‌تیک

جدول (۳-۱) خط فرمانهای MySQL

اگر فراموش کردید که سمی‌کولون را در پایان دستور بیاورید، می‌توان آنرا در خط بعدی آورد:

```
mysql> SELECT USER()
    -> ;
+--------------------+
| USER()             |
+--------------------+
| joesmith@localhost |
+--------------------+
```

۳-۳- ایجاد و استفاده از پایگاه داده

این قسمت شامل موارد زیر است:

- ایجاد پایگاه داده
- ایجاد جدول
- بار کردن داده به جدول
- بازیافت داده از جدول به روشهای مختلف
- استفاده از چندین جدول

برای مشاهده پایگاه داده‌های موجود از فرمان SHOW استفاده نمایید:

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
| tmp      |
+----------+
```

برای استفاده از یک پایگاه داده خاص از فرمان USE استفاده نمایید:

```
mysql> USE test
Database changed
```

توجه شود که USE مانند QUIT نیازی به سمی‌کولون در پایان دستور ندارد.

۳-۳-۱- ایجاد و انتخاب پایگاه داده

برای ایجاد پایگاه داده‌ای با نام menagerie بصورت زیر عمل کنید:

```
mysql> CREATE DATABASE menagerie;
```

در یونیکس، نام پایگاه داده بر خلاف کلمات کلیدی mysql، حساس به حالت حروف است. لذا پایگاه داده با نامهای menagerie ، Managerie و MANAGERIE با هم متفاوت هستند.

برای انتخاب پایگاه داده ایجاد شده به صورت زیر رفتار شود:

```
mysql> USE menagerie
Database changed
```

پایگاه داده تنها یکبار ایجاد شده، و به دفعات می‌توان آنرا انتخاب کرد. می‌توان پایگاه داده مورد نظر را همزمان با برقراری ارتباط با کارساز مشخص کرد:

```
shell> mysql -h host -u user -p menagerie
Enter password: ********
```

توجه شود که menagerie کلمه عبور نیست بلکه نام پایگاه داده‌ای است که قصد ارتباط با آنرا داریم.

۳-۳-۲- ایجاد جدول

بعد از ایجاد پایگاه داده و انتخاب آن، برای مشاهده جداول موجود در آن بصورت زیر رفتار نمایید:

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

برای ایجاد جدول از دستور CREATE TABLE استفاده شود، به طور مثال:

```
mysql> CREATE TABLE pet (name VARCHAR(20), owner
VARCHAR(20),
    -> species VARCHAR(20), sex CHAR(1), birth DATE,
death DATE);
```

فرمان بالا جدولی با نام pet با تعدادی ستون ایجاد می‌کند. اکنون اگر فرمان SHOW TABLES را بار دیگر اجرا کنیم، خواهیم داشت:

```
mysql> SHOW TABLES;
+---------------------+
| Tables in menagerie |
+---------------------+
| pet                 |
+---------------------+
```

برای بازبینی جدول ایجاد شده از دستور DESCRIBE استفاده شود:

```
mysql> DESCRIBE pet;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(20) | YES  |     | NULL    |       |
| owner   | varchar(20) | YES  |     | NULL    |       |
| species | varchar(20) | YES  |     | NULL    |       |
| sex     | char(1)     | YES  |     | NULL    |       |
| birth   | date        | YES  |     | NULL    |       |
| death   | date        | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```

۳-۳-۳- بار کردن داده به درون جدول

بعد از ایجاد جدول برای قرار دادن داده در آن از LOAD DATA و INSERT می‌توان استفاده کرد. فرض کنید رکوردهای جدول pet بصورت زیر باشند:

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	
Claws	Gwen	cat	m	1994-03-17	
Buffy	Harold	dog	f	1989-05-13	
Fang	Benny	dog	m	1990-08-27	
Bowser	Diane	dog	m	1979-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	
Whistler	Gwen	bird		1997-12-09	
Slim	Benny	snake	m	1996-04-29	

برای این‌کار یک فایل متنی (pet.txt) ایجاد کرده، و در هر خط از این فایل یک رکورد از این جدول را قرار دهید. ستونهای مربوط به هر رکورد را با استفاده از tab از هم جدا کرده و بعد از آن برای انتقال محتوای فایل مورد نظر به درون جدول به صورت زیر عمل کنید:

```
mysql> LOAD DATA LOCAL INFILE '/path/pet.txt' INTO TABLE
pet;
```

در صورتی که برای ایجاد فایل از ویرایشگرهای ویندوزی استفاده شود، در پایان خط از عبارت \r\n استفاده کنید.

```
mysql> LOAD DATA LOCAL INFILE '/path/pet.txt' INTO TABLE
pet
    -> LINES TERMINATED BY '\r\n';
```

برای اضافه کردن رکورد جدید به جدول ازINSERT استفاده شود:

```
mysql> INSERT INTO pet
    -> VALUES ('Puffball','Diane','hamster','f','1999-
03-30',NULL);
```

همان‌طور که مشاهده می‌شود، دستور LOAD DATA عمل چندینINSERT را انجام می‌دهد.

۳-۳-۴- بازیافت داده‌ها از جدول

از عبارت SELECT برای مشاهده داده‌های مورد نظر جدول می‌توان استفاده کرد:

```
SELECT what_to_select
FROM which_table
WHERE conditions_to_satisfy;
```

what_to_select چیزی است که قرار است دیده شود که می‌تواند تعدادی از ستونها باشد و یا همه ستونها (*) باشد. which_table نام جدولی است که قرار است از داده‌های آن استفاده

شود و conditions_to_satisfy شرایط داده‌هایی است که قرار است نمایش داده شوند. به طور مثال برای انتخاب همه رکوردهای جدول pet خواهیم داشت:

```
mysql> SELECT * FROM pet;
+----------+--------+---------+------+------------+-----
-------+
| name     | owner  | species | sex  | birth      |
death      |
+----------+--------+---------+------+------------+-----
-------+
| Fluffy   | Harold | cat     | f    | 1993-02-04 | NULL
|
| Claws    | Gwen   | cat     | m    | 1994-03-17 | NULL
|
| Buffy    | Harold | dog     | f    | 1989-05-13 | NULL
|
| Fang     | Benny  | dog     | m    | 1990-08-27 | NULL
|
| Bowser   | Diane  | dog     | m    | 1979-08-31 |
1995-07-29 |
| Chirpy   | Gwen   | bird    | f    | 1998-09-11 | NULL
|
| Whistler | Gwen   | bird    | NULL | 1997-12-09 | NULL
|
| Slim     | Benny  | snake   | m    | 1996-04-29 | NULL
|
| Puffball | Diane  | hamster | f    | 1999-03-30 | NULL
|
+----------+--------+---------+------+------------+-----
-------+
```

برای تغییر تاریخ تولد شخصی خاص به صورت زیر رفتار می‌شود:

```
mysql> UPDATE pet SET birth = '1989-08-31' WHERE name
= 'Bowser';
```

و یا برای مشاهده رکوردهایی خاص خواهیم داشت:

```
mysql> SELECT * FROM pet WHERE name = 'Bowser';
+--------+-------+---------+------+------------+--------
----+
| name   | owner | species | sex  | birth      | death
|
+--------+-------+---------+------+------------+--------
----+
| Bowser | Diane | dog     | m    | 1989-08-31 | 1995-
07-29 |
+--------+-------+---------+------+------------+--------
----+
```

دستور بالا رکوردهایی از جدول pet که فیلد نام در آنها Bowser است را نمایش می‌دهد.

برای مشاهده رکوردها با تاریخ تولد مشخص، خواهیم داشت:

```
mysql> SELECT * FROM pet WHERE birth >= '1998-1-1';
+----------+-------+---------+------+------------+------
-+
| name     | owner | species | sex  | birth      | death
|
+----------+-------+---------+------+------------+------
-+
```

```
| Chirpy   | Gwen  | bird    | f    | 1998-09-11 | NULL
|
| Puffball | Diane | hamster | f    | 1999-03-30 | NULL
|
+----------+-------+---------+------+------------+------
-+
```

می توان چندین شرط را با هم ترکیب کرد. مثلا:

```
mysql> SELECT * FROM pet WHERE species = 'dog' AND sex =
'f';
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
```

بجای استفاده از عملگر منطقی AND می‌توان از OR نیز استفاده کرد:

```
mysql> SELECT * FROM pet WHERE species = 'snake' OR
species = 'bird';
+----------+-------+---------+------+------------+------
-+
| name     | owner | species | sex  | birth      | death
|
+----------+-------+---------+------+------------+------
-+
| Chirpy   | Gwen  | bird    | f    | 1998-09-11 | NULL
|
| Whistler | Gwen  | bird    | NULL | 1997-12-09 | NULL
|
| Slim     | Benny | snake   | m    | 1996-04-29 | NULL
|
+----------+-------+---------+------+------------+------
-+
```

AND و OR را با هم نیز می‌توان استفاده کرد:

```
mysql> SELECT * FROM pet WHERE (species = 'cat' AND sex
= 'm')
    -> OR (species = 'dog' AND sex = 'f');
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Claws | Gwen   | cat     | m    | 1994-03-17 | NULL  |
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
```

برای انتخاب ستونهای خاص به صورت زیر رفتار شود:

```
mysql> SELECT name, birth FROM pet;
+----------+------------+
| name     | birth      |
+----------+------------+
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Buffy    | 1989-05-13 |
```

```
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Puffball | 1999-03-30 |
+----------+------------+
```

برای عدم نمایش ردیفهای تکراری از عبارت DISTINCT استفاده شود:

```
mysql> SELECT DISTINCT owner FROM pet;
+--------+
| owner  |
+--------+
| Benny  |
| Diane  |
| Gwen   |
| Harold |
+--------+
```

برای مرتب‌سازی داده‌ها از عبارت ORDER BY استفاده شود:

```
mysql> SELECT name, birth FROM pet ORDER BY birth;
+----------+------------+
| name     | birth      |
+----------+------------+
| Buffy    | 1989-05-13 |
| Bowser   | 1989-08-31 |
| Fang     | 1990-08-27 |
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Slim     | 1996-04-29 |
| Whistler | 1997-12-09 |
| Chirpy   | 1998-09-11 |
| Puffball | 1999-03-30 |
+----------+------------+
```

برای مرتب سازی به صورت برعکس از عبارت DESC استفاده شود:

```
mysql> SELECT name, birth FROM pet ORDER BY birth DESC;
+----------+------------+
| name     | birth      |
+----------+------------+
| Puffball | 1999-03-30 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Claws    | 1994-03-17 |
| Fluffy   | 1993-02-04 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Buffy    | 1989-05-13 |
+----------+------------+
```

عملیات مرتب‌سازی را می‌توان بر اساس چندین ستون نیز انجام داد:

```
mysql> SELECT name, species, birth FROM pet
    -> ORDER BY species, birth DESC;
+----------+---------+------------+
| name     | species | birth      |
+----------+---------+------------+
| Chirpy   | bird    | 1998-09-11 |
| Whistler | bird    | 1997-12-09 |
| Claws    | cat     | 1994-03-17 |
| Fluffy   | cat     | 1993-02-04 |
```

```
| Fang     | dog     | 1990-08-27 |
| Bowser   | dog     | 1989-08-31 |
| Buffy    | dog     | 1989-05-13 |
| Puffball | hamster | 1999-03-30 |
| Slim     | snake   | 1996-04-29 |
+----------+---------+------------+
```

پایگاه داده MySQL توابع مختلفی را برای استفاده از تاریخ در اختیار شما می‌گذارد. به طور مثال دستور زیر تاریخ تولد، تاریخ جاری، و سن را برای هر نام مشخص می‌کند:

```
mysql> SELECT name, birth, CURDATE(),
    -> (YEAR(CURDATE())-YEAR(birth))
    -> - (RIGHT(CURDATE(),5)<RIGHT(birth,5))
    -> AS age
    -> FROM pet;
+----------+------------+------------+------+
| name     | birth      | CURDATE()  | age  |
+----------+------------+------------+------+
| Fluffy   | 1993-02-04 | 2003-08-19 |   10 |
| Claws    | 1994-03-17 | 2003-08-19 |    9 |
| Buffy    | 1989-05-13 | 2003-08-19 |   14 |
| Fang     | 1990-08-27 | 2003-08-19 |   12 |
| Bowser   | 1989-08-31 | 2003-08-19 |   13 |
| Chirpy   | 1998-09-11 | 2003-08-19 |    4 |
| Whistler | 1997-12-09 | 2003-08-19 |    5 |
| Slim     | 1996-04-29 | 2003-08-19 |    7 |
| Puffball | 1999-03-30 | 2003-08-19 |    4 |
+----------+------------+------------+------+
```

MySQL به شما اجازه استفاده از متغیر NULL را نیز می‌دهد:

```
mysql> SELECT 1 = NULL, 1 <> NULL, 1 < NULL, 1 > NULL;
+----------+-----------+----------+----------+
| 1 = NULL | 1 <> NULL | 1 < NULL | 1 > NULL |
+----------+-----------+----------+----------+
|     NULL |      NULL |     NULL |     NULL |
+----------+-----------+----------+----------+
```

از علائم زیر در پرس‌وجوها می‌توان استفاده کرد:

'_` : بجای هر نویسه

'%` : بجای تعدادی نویسه

'[...]` : یک مجموعه از نویسه‌ها مثلا [abc]

'[a-z]` : شامل نویسه‌های بین دو نویسه مشخص شده

'^` : آغاز یک رشته

'$` : پایان یک رشته

مثال۱

```
mysql> SELECT * FROM pet WHERE name LIKE '%fy';
+--------+--------+---------+------+------------+-------
+
| name   | owner  | species | sex  | birth      | death
|
+--------+--------+---------+------+------------+-------
+
| Fluffy | Harold | cat     | f    | 1993-02-04 | NULL
|
```

```
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL
|
+--------+--------+---------+------+------------+-------
+
```

مثال۲

```
mysql> SELECT * FROM pet WHERE name LIKE '_____';
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Claws | Gwen   | cat     | m    | 1994-03-17 | NULL  |
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
```

مثال ۳

```
mysql> SELECT * FROM pet WHERE name REGEXP '^b';
+--------+--------+---------+------+------------+-------
-----+
| name   | owner  | species | sex  | birth      | death
|
+--------+--------+---------+------+------------+-------
-----+
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL
|
| Bowser | Diane  | dog     | m    | 1989-08-31 | 1995-
07-29 |
+--------+--------+---------+------+------------+-------
-----+
```

مثال۴

```
mysql> SELECT * FROM pet WHERE name REGEXP 'fy$';
+--------+--------+---------+------+------------+-------
+
| name   | owner  | species | sex  | birth      | death
|
+--------+--------+---------+------+------------+-------
+
| Fluffy | Harold | cat     | f    | 1993-02-04 | NULL
|
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL
|
+--------+--------+---------+------+------------+-------
+
```

جهت شمارش تعداد ردیفها، می‌توان از COINT(*) استفاده کرد:

```
mysql> SELECT COUNT(*) FROM pet;
+----------+
| COUNT(*) |
+----------+
|        9 |
+----------+
```

از GROUP BY برای گروه بندی مجموعه‌ای از رکوردها می‌توان استفاده کرد:

```
mysql> SELECT owner, COUNT(*) FROM pet GROUP BY owner;
+--------+----------+
| owner  | COUNT(*) |
+--------+----------+
```

```
| Benny  |        2 |
| Diane  |        2 |
| Gwen   |        3 |
| Harold |        2 |
+--------+----------+
```

در پرس و جوها می‌توان از بیش از یک جدول استفاده کرد:

```
mysql> SELECT pet.name,
    -> (YEAR(date)-YEAR(birth)) -
(RIGHT(date,5)<RIGHT(birth,5)) AS age,
    -> remark
    -> FROM pet, event
    -> WHERE pet.name = event.name AND type = 'litter';
+--------+------+----------------------------+
| name   | age  | remark                     |
+--------+------+----------------------------+
| Fluffy |    2 | 4 kittens, 3 female, 1 male |
| Buffy  |    4 | 5 puppies, 2 female, 3 male |
| Buffy  |    5 | 3 puppies, 3 female        |
+--------+------+----------------------------+
```

در مثال بالا از دو جدول pet و event در یک پرس و جو استفاده شده است.

۳-۴- دریافت اطلاعات در مورد پایگاه داده و جداول

برای مشاهده پایگاه داده‌ای که در حال استفاده از آن هستید، بصورت زیر رفتار کنید:

```
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| menagerie  |
+------------+
```

اگر هیچ پایگاه داده‌ای انتخاب نشده باشد، خروجی NULL خواهد بود.

برای مشاهده جداول موجود در پایگاه داده جاری از دستور زیر استفاده شود:

```
mysql> SHOW TABLES;
+---------------------+
| Tables in menagerie |
+---------------------+
| event               |
| pet                 |
+---------------------+
```

و برای مشاهده ساختار هر جدول از عبارت DESCRIBE استفاده شود:

```
mysql> DESCRIBE pet;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(20) | YES  |     | NULL    |       |
| owner   | varchar(20) | YES  |     | NULL    |       |
| species | varchar(20) | YES  |     | NULL    |       |
| sex     | char(1)     | YES  |     | NULL    |       |
| birth   | date        | YES  |     | NULL    |       |
| death   | date        | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```

۳-۴-۱- استفاده از mysql در Batch Mode

می‌توان فرامین مورد نظر جهت اجرا در MySQL را در فایلی ذخیره کرد و سپس آن فایل را با استفاده از mysql اجرا کرد.

```
shell> mysql < batch-file
```

اگر جهت برقراری ارتباط نیاز به نام کاربری ، کلمه عبور و ... دارید، دستور بالا بصورت زیر خواهد بود:

```
shell> mysql -h host -u user -p < batch-file
Enter password: ********
```

می‌توان خروجی حاصل از اجرای دستورات موجود در فایل را در فایلی دیگر ذخیره کرد:

```
shell> mysql < batch-file > mysql.out
```

۳-۵- چندین مثال از پرس و جوهای رایج

- اجرای mysql به همراه انتخاب پایگاه داده

```
shell> mysql your-database-name
```

- ایجاد و ذخیره داده‌ها در جدول

```
mysql> CREATE TABLE shop (
    -> article INT(4) UNSIGNED ZEROFILL DEFAULT '0000'
NOT NULL,
    -> dealer  CHAR(20)                 DEFAULT ''
NOT NULL,
    -> price   DOUBLE(16,2)             DEFAULT '0.00'
NOT NULL,
    -> PRIMARY KEY(article, dealer));
mysql> INSERT INTO shop VALUES
    ->
(1,'A',3.45),(1,'B',3.99),(2,'A',10.99),(3,'B',1.45),
    -> (3,'C',1.69),(3,'D',1.25),(4,'D',19.95);
```

- مشاهده محتویات یک جدول

```
mysql> SELECT * FROM shop;
+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0001 | A      |  3.45 |
|    0001 | B      |  3.99 |
|    0002 | A      | 10.99 |
|    0003 | B      |  1.45 |
|    0003 | C      |  1.69 |
|    0003 | D      |  1.25 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

- بزرگترین مقدار یک ستون

```
SELECT MAX(article) AS article FROM shop;

+---------+
| article |
+---------+
|       4 |
```

– حداکثر ستون برای گروه

```
SELECT article, MAX(price) AS price
FROM   shop
GROUP BY article

+---------+-------+
| article | price |
+---------+-------+
|    0001 |  3.99 |
|    0002 | 10.99 |
|    0003 |  1.69 |
|    0004 | 19.95 |
+---------+-------+
```

– استفاده از متغیرهای کاربر

```
mysql> SELECT
@min_price:=MIN(price),@max_price:=MAX(price) FROM shop;
mysql> SELECT * FROM shop WHERE price=@min_price OR
price=@max_price;
+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0003 | D      |  1.25 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

– استفاده از کلیدهای خارجی

```
CREATE TABLE person (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    name CHAR(60) NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE shirt (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    style ENUM('t-shirt', 'polo', 'dress') NOT NULL,
    color ENUM('red', 'blue', 'orange', 'white',
'black') NOT NULL,
    owner SMALLINT UNSIGNED NOT NULL REFERENCES
person(id),
    PRIMARY KEY (id)
);

INSERT INTO person VALUES (NULL, 'Antonio Paz');

SELECT @last := LAST_INSERT_ID();

INSERT INTO shirt VALUES
(NULL, 'polo', 'blue', @last),
(NULL, 'dress', 'white', @last),
(NULL, 't-shirt', 'blue', @last);

INSERT INTO person VALUES (NULL, 'Lilliana Angelovska');

SELECT @last := LAST_INSERT_ID();

INSERT INTO shirt VALUES
(NULL, 'dress', 'orange', @last),
(NULL, 'polo', 'red', @last),
(NULL, 'dress', 'blue', @last),
```

```
(NULL, 't-shirt', 'white', @last);

SELECT * FROM person;
+----+--------------------+
| id | name               |
+----+--------------------+
|  1 | Antonio Paz        |
|  2 | Lilliana Angelovska |
+----+--------------------+

SELECT * FROM shirt;
+----+---------+--------+-------+
| id | style   | color  | owner |
+----+---------+--------+-------+
|  1 | polo    | blue   |     1 |
|  2 | dress   | white  |     1 |
|  3 | t-shirt | blue   |     1 |
|  4 | dress   | orange |     2 |
|  5 | polo    | red    |     2 |
|  6 | dress   | blue   |     2 |
|  7 | t-shirt | white  |     2 |
+----+---------+--------+-------+

SELECT s.* FROM person p, shirt s
 WHERE p.name LIKE 'Lilliana%'
   AND s.owner = p.id
   AND s.color <> 'white';

+----+-------+--------+-------+
| id | style | color  | owner |
+----+-------+--------+-------+
|  4 | dress | orange |     2 |
|  5 | polo  | red    |     2 |
|  6 | dress | blue   |     2 |
+----+-------+--------+-------+
```

– جستجو روی دو کلید

```
SELECT field1_index, field2_index FROM test_table
WHERE field1_index = '1' OR field2_index = '1'
```

از UNION نیز برای ترکیب خروجی دو select در MySQL 4.0 و بالاتر می‌توان استفاده کرد:

```
SELECT field1_index, field2_index
    FROM test_table WHERE field1_index = '1'
UNION
SELECT field1_index, field2_index
    FROM test_table WHERE field2_index = '1';
```

– استفاده از AUTO_INCREMENT

با استفاده از AUTO_INCREMENT ردیفهای جدید به صورت واحد تعریف می‌شوند و هیچ ردیف تکراری نخواهیم داشت:

```
CREATE TABLE animals (
             id MEDIUMINT NOT NULL AUTO_INCREMENT,
             name CHAR(30) NOT NULL,
             PRIMARY KEY (id)
             );
```

```
INSERT INTO animals (name) VALUES
('dog'),('cat'),('penguin'),

('lax'),('whale'),('ostrich');
```

با توجه به عبارات بالا، خواهیم داشت:

```
SELECT * FROM animals;
+----+---------+
| id | name    |
+----+---------+
|  1 | dog     |
|  2 | cat     |
|  3 | penguin |
|  4 | lax     |
|  5 | whale   |
|  6 | ostrich |
+----+---------+
```

بخش چهارم
استفاده از برنامه‌های MySQL

این بخش شامل برنامه‌های ایجاد شده توسط MySQL AB و نحوه اجرای آنها است.

۴-۱- دید کلی از برنامه‌های MySQL

* کارساز MySQL و اسکریپتهای شروع کارساز:

- mysqld : کارساز MySQL است.
- mysqld_safe، mysql.server و mysqld_multi: اسکریپتهای شروع کارساز هستند.
- mysql_install_db : شاخه data و پایگاه داده اولیه

* برنامه‌های کارخواه که به کارساز دسترسی دارند:

- mysql : یک کارخواه خط فرمان برای اجرای فرامین SQL در حالت batch mode استفاده می‌شود.
- mysqlcc[1] : ابزاری گرافیکی برای اجرای عبارات SQL و مدیریت.
- mysqladmin : یک کارخواه با مجوز مدیر
- mysqlcheck : عملگرهای نگهداری جداول را اجرا می‌کند.
- mysqldump و mysqlhotcopy : جهت گرفتن پشتیبان از پایگاه داده استفاده می‌شود.
- mysqlimport : فایلهای داده را وارد می‌کند.
- mysqlshow : اطلاعاتی در مورد پایگاه داده و جداولش نمایش می‌دهد.

* برنامه‌های مستقل از کارساز

- myisamchk : عملگرهای نگهداری جداول را اجرا می‌کند.
- myisampack : جداول فقط خواندنی فشرده شده تولید می‌کند.
- mysqlbinlog : ابزاری جهت پردازش فایلهای باینری ثبت شده.
- perror : معنی کدهای خطا را نمایش می‌دهد.

۴-۲- مشخص کردن گزینه‌های برنامه

* به منظور غیر فعال کردن نام ستونها می‌توان از گزینه‌های زیر استفاده کرد:

```
--disable-column-names
--skip-column-names
--column-names=0
```

و برای فعال کردن آنها می‌توان از گزینه‌های زیر استفاده کرد:

```
--column-names
--enable-column-names
--column-names=1
```

* در یونیکس فایلهایی که در هنگام شروع MySQLخوانده می‌شوند، عبارتند از:

- /etc/my.cnf : شامل گزینه‌های عمومی
- DATADIR/my.cnf : گزینه‌های مخصوص کارساز
- defaults-extra-file : فایلی که مسیرش با defaults-extra-file=path-- مشخص شده است.

---

[1]) MySQL Control Center

– ~/.my.cnf : گزینه‌های مربوط به یک کاربر خاص.
DATADIR موقعیت شاخه data از MySQL است.

نمونه‌ای از گزینه‌های عمومی موجود در این فایلها بصورت زیر است:

```
[client]
port=3306
socket=/tmp/mysql.sock

[mysqld]
port=3306
socket=/tmp/mysql.sock
key_buffer_size=16M
max_allowed_packet=8M
```

از گزینه‌ها برای تنظیم متغیرهای برنامه می‌توان استفاده کرد:

```
shell> mysql --max_allowed_packet=16777216
shell> mysql --max_allowed_packet=16M
```

متغیر max_allowed_packet حداکثر اندازه فضای ارتباط را کنترل می‌کند.

بخش پنجم
مدیریت پایگاه داده
۵-۱- پیشگیری از حوادث بد و بازیافت
در این بخش با نحوه ایجاد پشتیبان برای پایگاه داده و نگهداری جداول آشنا می‌شویم.
۵-۱-۱- پشتیبان گرفتن از پایگاه داده
از آنجایی که جداول MySQL در فایل ذخیره می‌شوند، لذا گرفتن پشتیبان از آنها به راحتی انجام می‌گیرد.

اگر می‌خواهید از جداول مورد نظر با استفاده از دستورات SQL پشتیبان بگیرید می‌توان از SELECT INTO ... OUTFILE و یا BACKUP TABLE استفاده کرد.

روش دیگر برای گرفتن پشتیبان از پایگاه داده استفاده از برنامه mysqldump و یا mysqlhotcopy script است.

✧ پشتیبان گرفتن از پایگاه داده با استفاده از mysqldump

از mysqldump به منظور روبرداری از یک پایگاه داده و یا مجموعه‌ای از پایگاه داده‌ها به منظور انتقال به یک کارساز SQL دیگر استفاده می‌شود. در زیر چندین روش استفاده از mysqldump را مشاهده می‌کنید:

```
shell> mysqldump [options] db_name [tables]
shell> mysqldump [options] --databases DB1 [DB2 DB3...]
shell> mysqldump [options] --all-databases
```

اگر نام جداول را نمی‌دانید می‌توانید از databases-- و یا all-databases-- استفاده کنید.

برای مشاهده گرینه‌های پشتیبانی شونده توسط mysqldump می‌توان mysqldump –help را اجرا کرد.

گزینه‌های پشتیبانی شونده توسط mysqldump عبارتند از:

- --help, -? پیام کمک را نمایش می‌دهد.
- --add-drop-table یک عبارت DROP TABLE قبل از هر عبارت CREATE TABLE قرار می‌دهد.
- --all-databases, -A روبرداری از همه جداول
- --compress, -C اطلاعات ارسال شده بین کارساز و کارخواه را فشرده می‌کند.
- --databases, -B برای روبرداری از چندین پایگاه داده استفاده می‌شود.
- --default-character-set=charset برای تنظیم مجموعه نویسه پیش فرض
- --force, -f حتی در صورت بروز خطا، روبرداری ادامه می‌یابد.
- --host=host_name, -h host_name روبرداری داده از کارساز MySQL روی میزبان داده شده
- --password[=password], -p[password] کلمه عبور که برای ارتباط با کارساز استفاده می‌شود.
- --port=port_num, -P port_num شماره پورت TCP/IP که برای ارتباط استفاده می‌شود.
- --quick, -q برای روبرداری از جداول بزرگ مفید است.

- --result-file=file, -r file برای مشخص کردن فایل خروجی استفاده می‌شود.
- --socket=path, -S path فایل سوکت استفاده شده جهت ارتباط با localhost
- --user=user_name, -u user_name نام کاربری که MySQL برای ارتباط با کارساز استفاده می‌کند.
- --where='where-condition', -w 'where-condition' تنها رکوردهای مشخص شده توسط where را روبرداری می‌کند.

سایر گزینه‌های قابل استفاده عبارتند از:

```
--add-locks
--allow-keywords
--comments[={0|1}]
--compatible=name
--complete-insert, -c
--create-options
--debug[=debug_options]
--delayed
--delete-master-logs
--disable-keys, -K
--extended-insert, -e
--fields-terminated-by=...
--fields-enclosed-by=...
--fields-optionally-enclosed-by=...
--fields-escaped-by=...
--lines-terminated-by=...
--first-slave, -x
--flush-logs, -F
--hex-blob
--lock-all-tables, -x
--lock-tables, -l
--master-data[=value]
--no-create-db, -n
--no-create-info, -t
--no-data, -d
--opt
--protocol={TCP | SOCKET | PIPE | MEMORY}
--quote-names, -Q
--set-charset
--single-transaction
--skip-comments
--tab=path, -T path
--tables
--verbose, -v
--version, -V
--xml, -X
```

برنامه mysqldump جهت گرفتن پشتیبان از پایگاه داده بیشتر به صورت زیر بکار می‌رود:

```
shell> mysqldump --opt db_name > backup-file.sql
```

جهت قرار دادن فایل پشتیبان به درون کارساز:

```
shell> mysql db_name < backup-file.sql
```

اگر mysqldump را بدون quick– و یا opt– اجرا کنید، mysqldump قبل از روبرداری از نتایج، آنها را در حافظه بار می‌کند.

✧ پشتیبان گرفتن از پایگاه داده با استفاده از mysqlhotcopy

Mysqlhotcopy یک اسکریپت perl است که برای پشتیبان گرفتن سریع از پایگاه داده از LOCK TABLES, FLUSH TABLES، و cp یا scp استفاده می‌کند. و سریعترین روش برای گرفتن پشتیبان از پایگاه داده و جداول واحد است، ولی تنها بر روی همان سیستمی که فایلهای پایگاه داده مورد نظر قرار دارند قابل استفاده است. Mysqlhotcopy تنها برای گرفتن پشتیبان از جداول MyISAM و ISAM استفاده می‌شود. و تنها در یونیکس قابل اجرا است. البته برای MySQL-4.0.18 در NetWare نیز قابل اجرا است.

```
shell> mysqlhotcopy db_name [/path/to/new_directory]
shell> mysqlhotcopy db_name_1 ... db_name_n
/path/to/new_directory
shell> mysqlhotcopy db_name./regex/
```

برای مشاهده گزینه‌های مربوط به آن، از گزینه ? , help-- استفاده نمایید.

۵-۲- محلی‌سازی و کاربردهای بین‌المللی از MySQL

در این بخش با نحوه پیکربندی کارساز جهت استفاده از مجموعه نویسه‌های مختلف آشنا می‌شویم.

۵-۲-۱- مجموعه نویسه استفاده شده برای داده و مرتب سازی

MySQL به طور پیش فرض از مجموعه نویسه ISO-8859-1 (Latin1) با قوانین مرتب سازی Swedish/Finnish استفاده می‌کند. که این فرضیات برای ایالات متحده و اروپای غربی مناسب است.

برای اضافه کردن مجموعه نویسه‌های دیگر در کنار مجموعه نویسه پیش فرض، می‌توان از گزینه

--with-extra-charsets=complex استفاده کرد. مجموعه نویسه مشخص می‌کند چه نویسه‌هایی را می‌توان در نامها استفاده کرد. همچنین نحوه مرتب سازی در ORDER BY و GROUP BY در عبارت SELECT را مشخص می‌کند. مجموعه نویسه را می‌توان با گزینه --default-character-set در هنگام اجرای کارساز تغییر داد. مانند مجموعه نویسه تابع تطبیق را نیز می‌توان با گزینه default-collation-- تغییر داد. تابع تطبیق نحوه مرتب سازی را در مجموعه نویسه داده شده مشخص می‌کند. مجموعه نویسه مربوط به داده‌های فارسی utf8 و تابع تطبیق مربوط به داده‌های فارسی utf8_persian_ci است.

لذا کاربران فارسی زبان برای تنظیم مجموعه نویسه و تابع تطبیق در زمان اجرای کارساز mysql می‌توانند بصورت زیر رفتار کنند:

```
shell> mysqld --default-character-set=utf8 --default-
collation=utf8_persian_ci
```

اگر مایل به تنظیم مجموعه نویسه و تابع تطبیق در زمان نصب MySQL هستید، در زمان پیکربندی بصورت زیر رفتار کنید:

```
shell> ./configure --with-charset=utf8 --with-
collation=utf8_persian_ci
```

جهت آشنایی بیشتر با این موضوع، به بخش ۷ مراجعه شود.

۵-۳- اجرای چندین کارساز MySQL در یونیکس

آسانترین روش برای اجرای چندین کارساز در یونیکس، استفاده از پورتهای TCP/IP و فایلهای سوکت مختلف که هر کدام به واسطهای مختلف شبکه گوش می‌دهند، است. البته هر نصب جدید باید در مسیری جدید صورت گیرد.

شماره پورت TCP/IP پیش فرض ۳۳۰۶ و فایل سوکت یونیکسی پیش فرض /tmp/mysql.sock است. برای پیکربندی یک کارساز جدید که دارای پارامترهای عملگر مختلفی است، دستور configure را بصورت زیر اجرا کنید:

```
shell> ./configure --with-tcp-port=port_number \
            --with-unix-socket-path=file_name \
            --prefix=/usr/local/mysql-4.0.17
```

port_number و file_path باید با شماره و نام فایل سوکت پیش‌فرض که در نصب کارسازهای قبلی استفاده شده است، فرق کند، همچنین مسیر مشخص شده در جلوی --prefix باید با مسیر مشخص شده که برای نصب در دفعات قبل استفاده شده است، متفاوت باشد.

در محیطهایی که چندین کارساز MySQL وجود دارد، برنامه کارخواه جهت ارتباط با کارساز مورد نظر به روشهای زیر می‌تواند رفتار نماید:

- برنامه کارخواه با گزینه --host=host_name --port=port_number می‌تواند از طریق TCP/IP با یک کارساز دوردست ارتباط برقرار نماید.
- در محیط یونیکس قبل از اجرای برنامه کارخواه متغیرهای محلی MYSQL_UNIX_PORT و MYSQL_TCP_PORT که اشاره به فایل یونیکس و شماره پورت TCP/IP دارند، تنظیم شوند.
- تنظیم فایل سوکت یونیکس و شماره پورت TCP/IP پیش فرض در فایل گزینه (my.cnf).
- در برنامه‌های نوشته شده با زبان C ، می‌توان فایل سوکت و شماره پورت TCP/IP را در تابع mysql_real_connect() که مخصوص ارتباط با پایگاه داده است، مشخص کرد.

بخش ششم
ساختار زبان

این بخش شامل قوانین SQL استفاده شده در MySQL، است. که عبارتند از:

- متغیرهای حرفی مانند رشته‌ها و اعداد
- شناسه‌ها مانند نام جداول و ستونها
- متغیرهای سیستمی و کاربری
- توضیحات
- کلمات رزرو شده

۶-۱- متغیرهای حرفی

این بخش نحوه نوشتن متغیرهای حرفی در MySQL را نشان می‌دهد. که شامل رشته‌ها، اعداد، متغیرهای هگزادسیمال، متغیرهای بولین وNULL است.

۶-۱-۱- رشته‌ها

یک رشته ترتیبی از نویسه‌ها است که توسط نویسه (') و یا (") احاطه شده است. مانند:

```
'a string'
"another string"
```

درون یک رشته، توالی‌های معین دارای معانی خاص هستند. از جمله موارد زیر که با نویسه (\ ) شروع می‌شوند و با توجه به نویسه بعد از آن معنی خاصی دارند:

\0	نویسه NULL
\'	نویسه (`'')
\"	نویسه (`"')
\b	نویسه backspace
\n	نویسه خط جدید
\r	نویسه بازگردان carriage
\t	نویسه tab
\Z	در ویندوز به معنی پایان فایل
\\	نویسه (`\')
\%	نویسه `%'
_	نویسه `_'

جدول(۶-۱) رشته‌های بامعنی در MySQL

توجه شود که حالت حروف مهم است، مثلا منظور از b/ نویسه backspace و منظور از B/ نویسه B است.

در مثالهای زیر کاربرد موارد بالا را مشاهده می‌شود:

```
mysql> SELECT 'hello', '"hello"', '""hello""',
'hel''lo', '\'hello';
```

```
+-------+---------+-----------+--------+--------+
| hello | "hello" | ""hello"" | hel'lo | 'hello |
+-------+---------+-----------+--------+--------+

mysql> SELECT "hello", "'hello'", "''hello''",
"hel""lo", "\"hello";
+-------+---------+-----------+--------+--------+
| hello | 'hello' | ''hello'' | hel"lo | "hello |
+-------+---------+-----------+--------+--------+

mysql> SELECT 'This\nIs\nFour\nLines';
+--------------------+
| This
Is
Four
Lines |
+--------------------+

mysql> SELECT 'disappearing\ backslash';
+------------------------+
| disappearing backslash |
+------------------------+
```

۶-۱-۲- اعداد

Integer توالی اعداد است، float اعدادی هستند که در آنها از '.' استفاده شده است و اعداد منفی با '-' شروع می‌شوند. چند نمونه از اعداد integer بصورت زیر هستند:

```
1221
0
-32
```

چند نمونه از اعداد float بصورت زیر هستند:

```
294.42
-32032.6809e+10
148.00
```

۶-۱-۳- متغیرهای هگزا دسیمال

MySQL متغیرهای هگزا دسیمال را نیز پشتیبانی می‌کند. بطور مثال:

```
mysql> SELECT x'4D7953514C';
        -> 'MySQL'
mysql> SELECT 0xa+0;
        -> 10
mysql> SELECT 0x5061756c;
        -> 'Paul'
```

رشته‌های هگزا دسیمال اغلب توسط ODBC برای پشتیبانی از ستون BLOB استفاده می‌شوند.

۶-۱-۴- متغیرهای بولین

با MySQL 4.1 شروع شد. ثابت TRUE با 1 و ثابت FALSE با 0 ارزیابی می‌شود. بطور مثال:

```
mysql> SELECT TRUE, true, FALSE, false;
        -> 1, 1, 0, 0
```

۶-۱-۵- متغیر NULL

متغیر NULL به معنی عدم وجود داده است. و به هر حالتی می تواند نوشته شود. توجه شود که این متغیر با متغیر 0 برای اعداد و یا رشته خالی برای رشته‌ها، متفاوت است.

۶-۲- نام آلیاس، ستون، ایندکس، جدول و پایگاه داده

نامهای پایگاه داده، جدول، ایندکس، ستون و آلیاس، شناسه هستند. این بخش شامل سینتکس قابل قبول برای شناسه‌ها در MySQL است.

جدول زیر شامل حداکثر طول قابل قبول و نویسه‌های مجاز در شناسه‌ها است:

شناسه	حداکثر طول (بایت)	نویسه‌های مجاز
پایگاه داده	64	همه نویسه‌هایی که می‌توان در نام یک دایرکتوری قرار داد. موارد استثنا: `'/', `'\', `'.'
جدول	64	همه نویسه‌های مجاز در نام فایل. موارد استثنا: `'/', `'\', `'.'
ستون	64	همه نویسه‌ها
ایندکس	64	همه نویسه‌ها
آلیاس	255	همه نویسه‌ها

جدول(۶-۲) حداکثر طول قابل قبول و نویسه‌های مجاز در شناسه‌ها

۶-۲-۱- صفات شناسه

در MySQL هر نام می‌تواند دارای یک شناسه و یا چندین شناسه باشد. در MySQL برای دسترسی به یک ستون هر یک از روشهای زیر را می‌توان اختیار کرد:

مرجع ستون	معنی
col_name	ستون col_name از هر جدولی که در پرس و جو استفاده می‌شود
tbl_name.col_name	ستون col_name از جدول tbl_name از یک پایگاه داده پیش‌فرض
db_name.tbl_name.col_name	ستون col_name از جدول tbl_name از پایگاه داده db_name

جدول(۶-۳) روشهای دسترسی به یک ستون

۶-۳- متغیرهای کاربری

متغیرهای کاربری در MySQL از ویرایش 3.23.6 به بعد پشتیبانی می‌شوند. شما می‌توانید متغیری را برای استفاده‌های بعدی در یک متغیر کاربری ذخیره نمایید. متغیرهای کاربری ایجاد شده توسط هر کاربر قابل استفاده توسط سایر کاربران نمی‌باشد. همه متغیرهای کاربری برای همه کاربران موجود، آزاد و قابل استفاده است.

متغیرهای کاربری به شکل @var_name نوشته می شوند. var_name نام متغیر است که می‌تواند حاوی مجموعه‌ای از نویسه‌ها باشد. نام متغیرهای کاربری حساس به حالت حروف نیست.

یک روش برای تنظیم متغیرهای کاربری استفاده از عبارت SET است:

```
SET @var_name = expr [, @var_name = expr] ...
```

عملگر نسبت دادن در SET می‌تواند ' = ' و یا ' := ' باشد. یک متغیر تنظیم شده می‌تواند با متغیرهایی از نوع integer ، real، رشته و یا NULL ارزیابی شود.

بجای استفاده از عبارت SET برای نسبت دادن یک متغیر به یک متغیر کاربری می‌توانید از عبارات نیز استفاده نمایید، ولی در این حالت تنها مجاز به استفاده از ' := ' هستید و نمی توان از ' = ' استفاده کرد. مثال:

```
mysql> SET @t1=0, @t2=0, @t3=0;
mysql> SELECT @t1:=(@t2:=1)+@t3:=4,@t1,@t2,@t3;
+----------------------+------+------+------+
| @t1:=(@t2:=1)+@t3:=4 | @t1  | @t2  | @t3  |
+----------------------+------+------+------+
|                    5 |    5 |    1 |    4 |
+----------------------+------+------+------+
```

۶-۴- متغیرهای سیستمی

در MySQL 4.0.3 و بعد از آن قابل استفاده هستند که امکان دسترسی‌های بهتر به متغیرهای سیستمی و ارتباطی را فراهم می‌کنند. امکان تغییر متغیرها به صورت پویا در هنگام اجرای کارساز فراهم شد. با این قابلیت برای تنظیم عملگرهای کارساز نیازی به متوقف کردن و راه اندازی دوباره کارساز نیست.

دایمون mysqld حاوی دو نوع متغیر است. یکی متغیرهای عمومی که بر عملگرهای کلی سیستم تاثیر گذار هستند و دیگری متغیرهای جلسه‌ای که بصورت خاص بر ارتباط کارخواه‌ها تاثیرگذار است. با اجرا شدن کارساز همه متغیرهای عمومی مقدار پیش فرض خود را می‌گیرند. که این مقدار پیش فرض را می‌توان با تغییر فایلهای مربوطه و یا از طریق خط فرمان تغییر داد. برای تغییر این متغیرهای عمومی از طریق خط فرمان می‌توان از عبارت

SET GLOBAL var_name استفاده کرد. که برای این‌کار باید میزان دسترسی شما بالا باشد. کارساز همچنین شامل تعدادی متغیرهای جلسه‌ای خاص هر کاربر است، که کاربران برای تغییر این متغیرها می‌توانند از عبارت

SET SESSION var_name استفاده نمایند. توجه شود که هر کارخواه تنها قادر به تغییر متغیرهای جلسه‌ای مربوط به خود است و متغیرهای جلسه‌ای مربوط به سایر کاربران را نمی‌تواند تغییر دهد.

برای تغییر متغیر عمومی(GLOBAL) از یکی از دو روش زیر استفاده نمایید:

```
mysql> SET GLOBAL sort_buffer_size=value;
mysql> SET @@global.sort_buffer_size=value;
```

برای تنظیم متغیر جلسه‌ای(SESSION) به یکی از روشهای زیر رفتار نمایید:

```
mysql> SET SESSION sort_buffer_size=value;
mysql> SET @@session.sort_buffer_size=value;
mysql> SET sort_buffer_size=value;
```

برای بدست آوردن متغیر عمومی به یکی از دو روش زیر رفتار کنید:

```
mysql> SELECT @@global.sort_buffer_size;
mysql> SHOW GLOBAL VARIABLES like 'sort_buffer_size';
```

و برای بدست آوردن متغیر جلسه‌ای به یکی از روشهای زیر عمل نمایید:

```
mysql> SELECT @@sort_buffer_size;
```

```
mysql> SELECT @@session.sort_buffer_size;
mysql> SHOW SESSION VARIABLES like 'sort_buffer_size';
```

توجه شود که LOCAL معادل SESSION است. و از آن نیز می‌توان استفاده کرد.

در دستور SHOW VARIABLES اگر GLOBAL, SESSION و یا LOCAL مشخص نشود، گزینه پیش فرض SESSION است.

۶-۵- ساختار توضیحات (Comment)

کارساز MySQL سه حالت را برای توضیحات می‌پذیرد:

- نویسه '#': تا پایان خط را تبدیل به توضیح می‌کند.
- ترتیب ' --` : تا پایان خط را تبدیل به توضیح می‌کند.
- ترتیب '/*` : تا عبارت '*/` را تبدیل به توضیح می کند.

در زیر هر سه حالت توضیحات را مشاهده می‌نمایید:

```
mysql> SELECT 1+1;     # This comment continues to the
end of line
mysql> SELECT 1+1;     -- This comment continues to the
end of line
mysql> SELECT 1 /* this is an in-line comment */ + 1;
mysql> SELECT 1+
/*
this is a
multiple-line comment
*/
1;
```

۶-۶- کلمات رزرو شده در MySQL

جدول زیر حاوی کلمات رزرو شده در MySQL است، بسیاری از این کلمات خاص MySQL بوده و در SQL استاندارد نمی‌توان از آنها استفاده کرد. و تعدادی از این کلمات مخصوص ابزار تجزیه yacc است.

کلمه	کلمه	کلمه
ADD	ALL	ALTER
ANALYZE	AND	AS
ASC	ASENSITIVE	BEFORE
BETWEEN	BIGINT	BINARY
BLOB	BOTH	BY
CALL	CASCADE	CASE
CHANGE	CHAR	CHARACTER
CHECK	COLLATE	COLUMN
COLUMNS	CONDITION	CONNECTION
CONSTRAINT	CONTINUE	CONVERT
CREATE	CROSS	CURRENT_DATE
CURRENT_TIME	CURRENT_TIMESTAMP	CURRENT_USER
CURSOR	DATABASE	DATABASES

DAY_HOUR	DAY_MICROSECOND	DAY_MINUTE
DAY_SECOND	DEC	DECIMAL
DECLARE	DEFAULT	DELAYED
DELETE	DESC	DESCRIBE
DETERMINISTIC	DISTINCT	DISTINCTROW
DIV	DOUBLE	DROP
DUAL	EACH	ELSE
ELSEIF	ENCLOSED	ESCAPED
EXISTS	EXIT	EXPLAIN
FALSE	FETCH	FIELDS
FLOAT	FOR	FORCE
FOREIGN	FOUND	FROM
FULLTEXT	GOTO	GRANT
GROUP	HAVING	HIGH_PRIORITY
HOUR_MICROSECOND	HOUR_MINUTE	HOUR_SECOND
IF	IGNORE	IN
INDEX	INFILE	INNER
INOUT	INSENSITIVE	INSERT
INT	INTEGER	INTERVAL
INTO	IS	ITERATE
JOIN	KEY	KEYS
KILL	LEADING	LEAVE
LEFT	LIKE	LIMIT
LINES	LOAD	LOCALTIME
LOCALTIMESTAMP	LOCK	LONG
LONGBLOB	LONGTEXT	LOOP
LOW_PRIORITY	MATCH	MEDIUMBLOB
MEDIUMINT	MEDIUMTEXT	MIDDLEINT
MINUTE_MICROSECOND	MINUTE_SECOND	MOD
MODIFIES	NATURAL	NOT
NO_WRITE_TO_BINLOG	NULL	NUMERIC
ON	OPTIMIZE	OPTION
OPTIONALLY	OR	ORDER
OUT	OUTER	OUTFILE
PRECISION	PRIMARY	PRIVILEGES
PROCEDURE	PURGE	READ
READS	REAL	REFERENCES
REGEXP	RENAME	REPEAT
REPLACE	REQUIRE	RESTRICT
RETURN	REVOKE	RIGHT

RLIKE	SCHEMA	SCHEMAS
SECOND_MICROSECOND	SELECT	SENSITIVE
SEPARATOR	SET	SHOW
SMALLINT	SONAME	SPATIAL
SPECIFIC	SQL	SQLEXCEPTION
SQLSTATE	SQLWARNING	SQL_BIG_RESULT
SQL_CALC_FOUND_ROWS	SQL_SMALL_RESULT	SSL
STARTING	STRAIGHT_JOIN	TABLE
TABLES	TERMINATED	THEN
TINYBLOB	TINYINT	TINYTEXT
TO	TRAILING	TRIGGER
TRUE	UNDO	UNION
UNIQUE	UNLOCK	UNSIGNED
UPDATE	USAGE	USE
USING	UTC_DATE	UTC_TIME
UTC_TIMESTAMP	VALUES	VARBINARY
VARCHAR	VARCHARACTER	VARYING
WHEN	WHERE	WHILE
WITH	WRITE	XOR
YEAR_MONTH	ZEROFILL	

جدول(۶-۳) کلمات رزرو شده MySQL

بخش هفتم
پشتیبانی از مجموعه نویسه

پشتیبانی از مجموعه نویسه در ویرایش 4.1 از MySQL پیشرفت زیادی کرد. در این بخش با مفاهیم زیر آشنا می‌شویم:

- مفهوم مجموعه نویسه و تابع تطبیق
- سیستم پیش فرض چند سطحی
- ساختار جدید در MySQL 4.1
- عملگرها و توابع موثر
- پشتیبانی از یونی‌کد
- معنی هر تابع تطبیق و مجموعه نویسه خاص

۷-۱- مجموعه نویسه و تابع تطبیق

مجموعه نویسه۱ حاوی یک مجموعه از نمادها۲ و علائم رمزگذاری است. و تابع تطبیق۳ حاوی یک مجموعه قوانین برای مقایسه نویسه‌ها است. با چندین مثال این مفاهیم را روشن می‌کنیم.

فرض کنید الفبایی با چهار حرف داریم: `A', `B', `a', `b'. به هر یک ار این حروف یک عدد نسبت می‌دهیم `A' = 0, `B' = 1, `a' = 2, `c' = 3. حرف A یک نماد است و عدد 0 رمز آن است. به ترکیب این حروف و رمزهای آنها، مجموعه نویسه گفته می‌شود.

حال فرض کنید می‌خواهیم دو متغیر رشته‌ای A و B را با هم مقایسه کنیم. ساده‌ترین روش بررسی رمز مربوط به آنها است. 0 برای A و 1 برای B . از آنجایی که 0 کوچکتر از 1 است به همین دلیل A کوچکتر از B است. تابع تطبیق شامل مجموعه‌ای از قوانین این چنینی است که در اینجا تنها یک مورد را بیان کردیم. به این روش ساده قابل استفاده در تابع تطبیق، تابع تطبیق binary گفته می‌شود.

حال فرض کنید حالتی را داریم که در آن حالت حروف کرچک و حروف بزرگ هم ارز هستند. در این حالت نیاز به دو قانون داریم. که قانون اول عبارت است از معادل بودن حروف کوچک a و b با حروف بزرگ A و B است و قانون دوم مقایسه رمز هر حرف است. که معروف به تابع تطبیق case-insensitive است. که مقداری پیچیده‌تر از حالت اول است. اکثر توابع تطبیق دارای قوانین دیگری علاوه بر موارد فوق نیز هستند مانند قوانینی جهت بررسی لهجه (به طور مثال در زبان فارسی فتحه، ضمه، و کسره لهجه هستند).

MySQL 4.1 می‌تواند موارد زیر را پشتیبانی کند:

- ذخیره رشته با مجموعه نویسه‌های مختلف
- امکان مقایسه رشته‌هایی که از توابع تطبیق مختلف استفاده می‌کنند.
- امکان ترکیب رشته‌هایی با مجموعه نویسه و تابع تطبیق مختلف در یک کارساز، یک پایگاه داده و حتی یک جدول

---

[1] Character Set

[2] Symbols

[3] Collation

– اجازه مشخص کردن مجموعه نویسه و تابع تطبیق را در هر سطحی می‌دهد.

۷-۲- مجموعه نویسه و تابع تطبیق در MySQL

کارساز MySQL توانایی پشتیبانی از چندین مجموعه نویسه را دارد. برای مشاهده مجموعه نویسه‌های موجود از عبارت SHOW CHARACTER SET استفاده نمایید.

```
mysql> SHOW CHARACTER SET;
+----------+-----------------------------+-------------
-------+
| Charset  | Description                 | Default
collation   |
+----------+-----------------------------+-------------
-------+
| big5     | Big5 Traditional Chinese    |
big5_chinese_ci     |
| dec8     | DEC West European           |
dec8_swedish_ci     |
| cp850    | DOS West European           |
cp850_general_ci    |
| hp8      | HP West European            |
hp8_english_ci      |
| koi8r    | KOI8-R Relcom Russian       |
koi8r_general_ci    |
| latin1   | ISO 8859-1 West European    |
latin1_swedish_ci   |
| latin2   | ISO 8859-2 Central European |
latin2_general_ci   |
...
```

هر مجموعه نویسه حداقل دارای یک تابع تطبیق است، و می‌تواند دارای چندین تابع تطبیق نیز باشد.

برای مشاهده توابع تطبیق پشتیبانی شونده توسط MySQL از عبارت SHOW COLLATION استفاده نمایید. برای مثال برای مشاهده توابع تطبیق مربوط به latin1، بصورت زیر رفتار نمایید:

```
mysql> SHOW COLLATION LIKE 'latin1%';
+-------------------+---------+----+---------+----------
+---------+
| Collation         | Charset | Id | Default | Compiled
| Sortlen |
+-------------------+---------+----+---------+----------
+---------+
| latin1_german1_ci | latin1  |  5 |         |
|       0 |
| latin1_swedish_ci | latin1  |  8 | Yes     | Yes
|       1 |
| latin1_danish_ci  | latin1  | 15 |         |
|       0 |
| latin1_german2_ci | latin1  | 31 |         | Yes
|       2 |
| latin1_bin        | latin1  | 47 |         | Yes
|       1 |
| latin1_general_ci | latin1  | 48 |         |
|       0 |
| latin1_general_cs | latin1  | 49 |         |
|       0 |
```

```
| latin1_spanish_ci | latin1  | 94 |         |
|       0 |
+-------------------+---------+----+---------+----------
+---------+
```

توابع تطبیق latin1 دارای معانی زیر هستند:

تابع تطبیق	معنی
latin1_bin	Binary according to latin1 encoding
latin1_danish_ci	Danish/Norwegian
latin1_general_ci	Multilingual
latin1_general_cs	Multilingual, case sensitive
latin1_german1_ci	German DIN-1
latin1_german2_ci	German DIN-2
latin1_spanish_ci	Modern Spanish
latin1_swedish_ci	Swedish/Finnish

جدول(۶-۴) توابع تطبیق latin1

تابع تطبیق دارای مشخصات کلی زیر است:

– دو مجموعه نویسه مختلف نمی‌توانند دارای تابع تطبیق یکسان باشند.

– هر مجموعه نویسه دارای یک تابع تطبیق پیش‌فرض است. به طور مثال تابع تطبیق پیش‌فرض برای latin1، latin1_swedish_ci است.

– قراردادی برای نام تابع تطبیق وجود دارد: همه آنها با نام مجموعه نویسه مربوطه به همراه موارد پیوستی مانند نام زبان مربوطه شروع شده و به _ci (case insensitive), _cs (case sensitive) و یا .(binary) bin_ ختم می‌شوند.

۷-۳- تعیین مجموعه نویسه و تابع تطبیق پیش‌فرض

تنظیمات پیش فرض برای مجموعه نویسه و تابع تطبیق در چهار سطح امکان پذیر است: کارساز، پایگاه داده، جدول و ارتباط. در زیر به توضیح هر یک از این چهار روش می‌پردازیم:

۷-۳-۱- مجموعه نویسه و تابع تطبیق کارساز

کارساز MySQL دارای یک مجموعه نویسه کارساز و یک تابع تطبیق کارساز است که نمی‌توانند تهی باشند.

مجموعه نویسه و تابع تطبیق کارساز را به دو روش زیر می‌توان تنظیم کرد:

– تنظیم به نحوی که بر اجرای پیش فرض کارساز موثر باشد.

– تنظیم گزینه‌های مربوطه در زمان اجرای کارساز

برای حالت اول در زمان نصب MySQL پیکربندیهای لازمه انجام می‌شود، لذا برای این‌کار در زمان اجرای دستور configure به‌صورت زیر عمل می‌کنیم:

برای تنضیم مجموعه نویسه

```
shell> ./configure --with-charset=charset_name
```

و یا برای تنظیم مجموعه نویسه و تابع تطبیق خواهیم داشت:

```
shell> ./configure --with-charset=charset_name \
           --with-collation=collation_name
```

که دو عبارت بالا برای زبان فارسی به صورت زیر خواهند بود:

```
shell> ./configure --with-charset=utf8

shell> ./configure --with-charset=utf8 \
           --with-collation=utf8_persian_ci
```

اما روش دوم تنظیم مجموعه نویسه و تابع تطبیق در زمان اجرای کارساز است که برای آن بصورت زیر عمل می‌شود:

جهت تنظیم مجموعه نویسه:

```
shell> mysqld --default-character-set=charset_name
```

جهت تنظیم مجموعه نویسه و تابع تطبیق:

```
shell> mysqld --default-character-set=charset_name \
           --default-collation=collation_name
```

دو عبارت بالا برای زبان فارسی بصورت زیر خواهد بود:

```
shell> mysqld --default-character-set=utf8

shell> mysqld --default-character-set=utf8 \
           --default-collation=utf8_persian_ci
```

۷-۳-۲- مجموعه نویسه و تابع تطبیق پایگاه داده

هر پایگاه داده دارای یک مجموعه نویسه و تابع تطبیق مخصوص به خود دارد که نمی‌تواند تهی باشد. با استفاده از CREATE DATABASE و ALTER DATABASEمی‌توان مجموعه نویسه و تابع تطبیق را برای پایگاه داده مشخص کرد:

```
CREATE DATABASE db_name
    [[DEFAULT] CHARACTER SET charset_name]
    [[DEFAULT] COLLATE collation_name]

ALTER DATABASE db_name
    [[DEFAULT] CHARACTER SET charset_name]
    [[DEFAULT] COLLATE collation_name]
```

مثال:

```
CREATE DATABASE db_name
    DEFAULT CHARACTER SET utf8 COLLATE utf8_persian_ci;
```

۷-۳-۳- مجموعه نویسه و تابع تطبیق جدول

هر جدول دارای یک مجموعه نویسه و تابع تطبیق اختصاصی است که نمی‌تواند تهی باشد. عبارات

CREATE TABLE و ALTER TABLE برای تنظیم مجموعه نویسه و تابع تطبیق استفاده می‌شوند.

```
CREATE TABLE tbl_name (column_list)
    [DEFAULT CHARACTER SET charset_name [COLLATE
collation_name]]

ALTER TABLE tbl_name
```

```
    [DEFAULT CHARACTER SET charset_name] [COLLATE
collation_name]
```

مثال

```
CREATE TABLE t1 ( ... )
    DEFAULT CHARACTER SET utf8 COLLATE utf8_persian_ci;
```

۷-۳-۴- مجموعه نویسه و تابع تطبیق ستون

هر نوع ستون رشته‌ای (از نوع char، varchar، و یا text) دارای یک مجموعه نویسه و تابع تطبیق است که نمی‌تواند تهی باشد. و به صورت زیر در هنگام تعریف ستون قابل تنظیم است:

```
col_name {CHAR | VARCHAR | TEXT} (col_length)
    [CHARACTER SET charset_name [COLLATE
collation_name]]
```

مثال:

```
CREATE TABLE Table1
(
    column1 VARCHAR(5) CHARACTER SET utf8 COLLATE
utf8_persian_ci
);
```

۷-۳-۵- چندین مثال از تنظیم مجموعه نویسه و تابع تطبیق

مثال ۱: تعریف جدول + ستون

```
CREATE TABLE t1
(
    c1 CHAR(10) CHARACTER SET latin1 COLLATE
latin1_german1_ci
) DEFAULT CHARACTER SET latin2 COLLATE latin2_bin;
```

در این حالت مجموعه نویسه و تابع تطبیق برای ستون c1 با مجموعه نویسه و تابع تطبیق تنظیم شده برای جدول متفاوت است. برای ستونهایی که مجموعه نویسه و تابع تطبیق در آنها تنظیم نشود، مجموعه نویسه و تابع تطبیق پیش‌فرض آنها همان مجموعه نویسه و تابع تطبیق مربوط به جدول خواهد بود.

مثال ۲: تعریف پایگاه داده + جدول + ستون

```
CREATE DATABASE d1
    DEFAULT CHARACTER SET latin2 COLLATE
latin2_czech_ci;
USE d1;
CREATE TABLE t1
(
    c1 CHAR(10)
);
```

همان‌طور که مشاهده می‌شود مجموعه نویسه و تابع تطبیق برای جدول و ستون مشخص نشده است، در این حالت مجموعه نویسه و تابع تطبیق ستون به یک سطح بالاتر یعنی جدول، و مجموعه نویسه و تابع تطبیق جدول نیز معادل یک سطح بالاتر یعنی پایگاه داده خواهد شد.

۷-۳-۶- مجموعه نویسه و تابع تطبیق ارتباط

چندین مجموعه نویسه و تابع تطبیق با تعاملات کارخواه و کارساز در ارتباط هستند. وقتی کارخواه با کارساز ارتباط برقرار می‌نماید، کارخواه برای کارساز عبارت SQL را ارسال می‌نماید و

کارساز در جواب نتایج را برای کارخواه ارسال می‌نماید، برای هر یک از این تعاملات، مجموعه نویسه باید مشخص شود.

– برای تنظیم مجموعه نویسه جهت ارسال دستورات SQL از جانب کاربر از عبارت character_set_client استفاده می‌شود.

– در کارساز برای ترجمه دستورات SQL که از جانب کاربر آمده است، جهت تنظیم مجموعه نویسه و تابع تطبیق باید از character_set_connection و collation_connection استفاده کرد.

– در کارساز جهت نمایش نتایج به کارخواه و یا نمایش پیام خطا، مجموعه نویسه با استفاده از عبارت character_set_results تنظیم می‌شود.

برای تنظیم هر یک از موارد فوق می‌توان از عبارت SET استفاده کرد:

```
mysql> SET character_set_client = x;
mysql> SET character_set_results = x;
mysql> SET character_set_connection = x;
```

برای مشاهده مجموعه نویسه‌های مختلف از دستور زیر استفاده نمایید:

```
mysql> SHOW VARIABLES LIKE 'character_set_system'
```

۷-۳-۷- استفاده از COLLATE در عبارات SQL

از COLLATE در عبارات مختلف SQL می‌توان استفاده کرد. در زیر تعدادی از مثالهای مربوطه را مشاهده می‌نمایید:

- با ORDER BY

```
SELECT k
FROM t1
ORDER BY k COLLATE latin1_german2_ci;
```

- با AS

```
SELECT k COLLATE latin1_german2_ci AS k1
FROM t1
ORDER BY k1;
```

- با GROUP BY

```
SELECT k
FROM t1
GROUP BY k COLLATE latin1_german2_ci;
```

- با تابع جمع

```
SELECT MAX(k COLLATE latin1_german2_ci)
FROM t1;
```

- با DISTINCT

```
SELECT DISTINCT k COLLATE latin1_german2_ci
FROM t1;
```

- با WHERE

```
SELECT *
FROM t1
WHERE _latin1 'Müller' COLLATE latin1_german2_ci
= k;
```

- با HAVING

```
SELECT k
FROM t1
```

```
  GROUP BY k
  HAVING k = _latin1 'Müller' COLLATE
latin1_german2_ci;
```

۷-۴- عملگرهای موثر در پشتیبانی مجموعه نویسه

۷-۴-۱- رشته‌های نتیجه

MySQL دارای تعدادی عملگر و تابع است که رشته برمی‌گردانند. تابعی که یک رشته را به عنوان ورودی می‌گیرد و یک خروجی رشته‌ای را به عنوان نتیجه بازمی‌گرداند، مجموعه نویسه و تابع تطبیق خروجی آن معادل همان مجموعه نویسه و تابع تطبیق مربوط به ورودی است. بطور مثال رشته خروجی UPPER(X) معادل همان مجموعه نویسه و تابع تطبیق ورودی آن یعنی X است. بعضی دیگر از این نمونه عبارتند از:

```
INSTR(), LCASE(), LOWER(), LTRIM(), MID(), REPEAT(),
REPLACE(), REVERSE(), RIGHT(), RPAD(), RTRIM(), SOUNDEX(),
                        SUBSTRING(), TRIM(), UCASE(),
                                        و UPPER()
```

رشته‌هایی را که COLLATE آنها شبیه است را می‌توان با هم ترکیب کرد ولی رشته‌هایی که COLLATE آنها با هم متفاوت است را نمی‌توان با هم ترکیب کرد.

۷-۴-۲- CONVERT()

CONVERT() امکان برگرداندن داده میان مجموعه نویسه‌های مختلف را فراهم می‌کند. و سینتکس آن بصورت زیر است:

```
CONVERT(expr USING transcoding_name)
```

مثال:

```
SELECT CONVERT( _latin1'Müller' USING utf8);
INSERT INTO utf8table (utf8column)
    SELECT CONVERT(latin1field USING utf8) FROM
latin1table;
```

۷-۴-۳- CAST()

از CAST() نیز می‌توان برای برگرداندن رشته میان مجموعه نویسه‌های مختلف استفاده کرد. ساختار آن بصورت زیر است:

```
CAST(character_string AS character_data_type CHARACTER
SET charset_name)
```

مثال:

```
SELECT CAST(_latin1'test' AS CHAR CHARACTER SET utf8);
```

اگر از CAST() بدون مشخص کردن CHARACTER SET استفاده نمایید، مجموعه نویسه و تابع تطبیق توسط متغیرهای سیستمی character_set_connection وcollation_connection مشخص می‌شود. اگر از CAST() با CHARACTER SET X استفاده نمایید، مجموعه نویسه و تابع تطبیق X خواهند بود(تابع تطبیق پیش‌فرض X)

۷-۴-۴- عبارت SHOW

عبارت SHOW از ویرایش 4.1 به MySQL اضافه شده است و امکان مشاهده اطلاعات اضافی در مورد مجموعه نویسه را فراهم می‌نماید. عبارت SHOW CHARACTER SET همه

مجموعه نویسه‌های موجود را نشان می‌دهد. از عبارت LIKE به منظور مشاهده موارد خاص می‌توان استفاده کرد:

```
mysql> SHOW CHARACTER SET LIKE 'latin%';
+---------+-----------------------------+---------------
----+--------+
| Charset | Description                 | Default
collation | Maxlen |
+---------+-----------------------------+---------------
----+--------+
| latin1  | ISO 8859-1 West European    |
latin1_swedish_ci |      1 |
| latin2  | ISO 8859-2 Central European |
latin2_general_ci |      1 |
| latin5  | ISO 8859-9 Turkish          |
latin5_turkish_ci |      1 |
| latin7  | ISO 8859-13 Baltic          |
latin7_general_ci |      1 |
+---------+-----------------------------+---------------
----+--------+

mysql> SHOW COLLATION LIKE 'latin1%';
+-------------------+---------+----+---------+----------
+---------+
| Collation         | Charset | Id | Default | Compiled
| Sortlen |
+-------------------+---------+----+---------+----------
+---------+
| latin1_german1_ci | latin1  |  5 |         |
|       0 |
| latin1_swedish_ci | latin1  |  8 | Yes     | Yes
|       0 |
| latin1_danish_ci  | latin1  | 15 |         |
|       0 |
| latin1_german2_ci | latin1  | 31 |         | Yes
|       2 |
| latin1_bin        | latin1  | 47 |         | Yes
|       0 |
| latin1_general_ci | latin1  | 48 |         |
|       0 |
| latin1_general_cs | latin1  | 49 |         |
|       0 |
| latin1_spanish_ci | latin1  | 94 |         |
|       0 |
+-------------------+---------+----+---------+----------
+---------+
```

۷-۵- پشتیبانی از یونی‌کد

به MySQL 4.1 دو مجموعه نویسه جدید برای ذخیره داده‌ه یونی‌کد اضافه شده که عبارتند از:

- ucs2
- utf8

در UCS-2 هر نویسه با دو بایت ارائه شده است به طور مثال کد مربوط به حرف لاتین A بزرگ بصورت 0x0041 است که در توالی دو بایت بصورت 0x41 0x00 ذخیره شده است.

یک محدودیت موقتی در مجموعه نویسه UCS-2 وجود دارد که امکان استفاده آن‌را در برنامه‌های کارخواه نمی‌دهد. لذا پیشنهاد می‌شود در ویرایشهای کنونی (MySQL 5.0.2 و قبل از آن) از این مجموعه نویسه استفاده نشود. البته در ویرایشهای بعدی نیز این محدودیت باید مورد بررسی قرار گیرد.

پیشنهاد دیگر برای ذخیره داده یونی‌کد استفاده از UTF8 است، که بر اساس RFC2279 استفاده شده است. در مجموعه نویسه UTF8 که نویسه‌های یونی‌کد مختلفی را در خود جای داده است از توالی بایتها با اندازه‌های مختلف استفاده می‌شود:

– برای حروف اصلی لاتین، ارقام، و علائم نقطه‌گذاری از یک بایت استفاده می‌شود.
– برای بیشتر کشورهای اروپایی و خاور میانه از دو بایت استفاده می‌شود.
– برای کره، چین، و علائم خاص ژاپن از سه بایت استفاده می‌شود.

در حال حاظر MySQL از چهار بایت UTF8 استفاده نمی‌کند.

۷-۶- UTF8 برای فرا داده

فرا داده[1] داده‌ای است در مورد داده. هر چیزی که پایگاه داده را شرح می‌دهد ولی محتوای پایگاه داده نیست، فرا داده است. مانند:نام ستون، نام پایگاه داده، نام کاربری، نام ویرایش، و بیشتر نتایج رشته‌ای SHOW فرا داده هستند.

فرادادە باید موارد زیر را پشتیبانی نماید:

– همه فرادادەها باید دارای مجموعه نویسه یکسان باشند.
– فرداده باید نویسه‌های مربوط به همه زبانها را پشتیبانی کند، در غیر این‌صورت کاربران قادر نخواهند بود نام ستون و جدول را با زبان خود ایجاد کنند.

به منظور پشتیبانی این دو نیازمندی، MySQL برای ذخیره فراداده از مجموعه نویسه UTF8 استفاده می‌کند.

کارساز متغیر سیستمی character_set_system را با نام مجموعه نویسه فرا داده تنظیم می‌کند:

```
mysql> SHOW VARIABLES LIKE 'character_set_system';
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| character_set_system | utf8  |
+----------------------+-------+
```

۷-۷- مجموعه نویسه و توابع تطبیق پشتیبانی شونده توسط MySQL

MySQL بیش از ۷۰ تابع تطبیق و بیش از ۳۰ مجموعه نویسه را پشتیبانی می کند. که برای مشاهده این مجموعه نویسه‌ها از عبارت SHOW CHARACTER SET استفاده می‌شود:

```
mysql> SHOW CHARACTER SET;
```

[1] Metadata

```
+----------+-----------------------------+-------------
-------+
| Charset  | Description                 | Default
collation   |
+----------+-----------------------------+-------------
-------+
| big5     | Big5 Traditional Chinese    |
big5_chinese_ci     |
| dec8     | DEC West European           |
dec8_swedish_ci     |
| cp850    | DOS West European           |
cp850_general_ci    |
| hp8      | HP West European            |
hp8_english_ci      |
| koi8r    | KOI8-R Relcom Russian       |
koi8r_general_ci    |
| latin1   | ISO 8859-1 West European    |
latin1_swedish_ci   |
| latin2   | ISO 8859-2 Central European |
latin2_general_ci   |
| swe7     | 7bit Swedish                |
swe7_swedish_ci     |
| ascii    | US ASCII                    |
ascii_general_ci    |
| ujis     | EUC-JP Japanese             |
ujis_japanese_ci    |
| sjis     | Shift-JIS Japanese          |
sjis_japanese_ci    |
| cp1251   | Windows Cyrillic            |
cp1251_bulgarian_ci |
| hebrew   | ISO 8859-8 Hebrew           |
hebrew_general_ci   |
| tis620   | TIS620 Thai                 |
tis620_thai_ci      |
| euckr    | EUC-KR Korean               |
euckr_korean_ci     |
| koi8u    | KOI8-U Ukrainian            |
koi8u_general_ci    |
| gb2312   | GB2312 Simplified Chinese   |
gb2312_chinese_ci   |
| greek    | ISO 8859-7 Greek            |
greek_general_ci    |
| cp1250   | Windows Central European    |
cp1250_general_ci   |
| gbk      | GBK Simplified Chinese      |
gbk_chinese_ci      |
| latin5   | ISO 8859-9 Turkish          |
latin5_turkish_ci   |
| armscii8 | ARMSCII-8 Armenian          |
armscii8_general_ci |
| utf8     | UTF-8 Unicode               |
utf8_general_ci     |
| ucs2     | UCS-2 Unicode               |
ucs2_general_ci     |
| cp866    | DOS Russian                 |
cp866_general_ci    |
| keybcs2  | DOS Kamenicky Czech-Slovak  |
keybcs2_general_ci  |
| macce    | Mac Central European        |
macce_general_ci    |
```

```
| macroman | Mac West European           |
macroman_general_ci |
| cp852    | DOS Central European        |
cp852_general_ci    |
| latin7   | ISO 8859-13 Baltic          |
latin7_general_ci   |
| cp1256   | Windows Arabic              |
cp1256_general_ci   |
| cp1257   | Windows Baltic              |
cp1257_general_ci   |
| binary   | Binary pseudo charset       | binary
|
| geostd8  | GEOSTD8 Georgian            |
geostd8_general_ci  |
+----------+-----------------------------+-------------
-------+
```

MySQL دارای دو مجموعه نویسه یونی‌کد است. با استفاده از این دو مجموعه نویسه برای ذخیره داده‌ها از ۶۵۰ زبان مختلف می‌توان استفاده کرد. در زیر تعدادی از توابع تطبیق مربوط به این دو مجموعه نویسه مشاهده می‌شود:

- `ucs2` (UCS-2 Unicode) collations:
    - `ucs2_bin`
    - `ucs2_czech_ci`
    - `ucs2_danish_ci`
    - `ucs2_estonian_ci`
    - `ucs2_general_ci` (default)
    - `ucs2_icelandic_ci`
    - `ucs2_latvian_ci`
    - `ucs2_lithuanian_ci`
    - **`ucs2_persian_ci`**
    - `ucs2_polish_ci`
    - `ucs2_roman_ci`
    - `ucs2_romanian_ci`
    - `ucs2_slovak_ci`
    - `ucs2_slovenian_ci`
    - `ucs2_spanish2_ci`
    - `ucs2_spanish_ci`
    - `ucs2_swedish_ci`

    - ucs2_turkish_ci
    - ucs2_unicode_ci
- utf8 (UTF-8 Unicode) collations:
    - utf8_bin
    - utf8_czech_ci
    - utf8_danish_ci
    - utf8_estonian_ci
    - utf8_general_ci (default)
    - utf8_icelandic_ci
    - utf8_latvian_ci
    - utf8_lithuanian_ci
    - **utf8_persian_ci**
    - utf8_polish_ci
    - utf8_roman_ci
    - utf8_romanian_ci
    - utf8_slovak_ci
    - utf8_slovenian_ci
    - utf8_spanish2_ci
    - utf8_spanish_ci
    - utf8_swedish_ci
    - utf8_turkish_ci
    - utf8_unicode_ci

بخش هشتم
انواع ستون

MySQL انواع ستون در مقوله‌های مختلف را پشتیبانی می‌کند: نوع عددی، نوع تاریخ و زمان، و نوع رشته‌ای. در این بخش نگاهی اجمالی به انواع این ستونها داریم.

۸-۱- نظری به انواع ستون عددی

- `TINYINT[(M)] [UNSIGNED] [ZEROFILL]`

داده عددی خیلی کوچک، مابین 127- تا 127 و نوع بی‌علامت[1] آن بین 0 تا 255

- `BIT`
- `BOOL`
- `BOOLEAN`

این موارد معادل TINYINT(1) هستند.

- `SMALLINT[(M)] [UNSIGNED] [ZEROFILL]`

نوع عددی کوچک. بین 32768- تا 32768 و نوع بی‌علامت آن بین 0 تا 65535

- `MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]`

نوع عددی با اندازه متوسط، محدوده این نوع داده عددی ما بین 8388608- و 8388607 است و نوع بی‌علامت آن از 0 تا 16777215 است.

- `INT[(M)] [UNSIGNED] [ZEROFILL]`

نوع عددی نرمال، محدوده آن بین 2147483648- تا 2147483647 است و محدوده بی‌علامت آن بین 0 تا 4294967295 است.

- `INTEGER[(M)] [UNSIGNED] [ZEROFILL]`

معادل همان INT است.

- `BIGINT[(M)] [UNSIGNED] [ZEROFILL]`

نوع عددی بزرگ، محدوده آن بین 9223372036854775808- تا 9223372036854775807 است و محدوده بی‌علامت آن از 0 تا 18446744073709551615 است.

- `FLOAT(p) [UNSIGNED] [ZEROFILL]`

نوع داده عددی شناور، که در آن p نشان دهنده میزان دقت است. P برای داده‌های شناور با دقت واحد[2] بین 0 تا 24 می‌تواند باشد و برای داده‌های شناور با دقت مضاعف[3] ما بین 25 تا 53 است.

- `FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]`

نوع داده عددی شناور (با دقت واحد). که مجوز اعدادی با اندازه 3.402823466E+38- تا 3.402823466E+38- ، 0، و 1.175494351E-38 تا 3.402823466E+38 را می‌دهد.

- `DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]`

نوع داده عددی شناور با اندازه نرمال. که محدوده متغیرهای آن عبارتند از: 1.7976931348623157E+308- تا 2.2250738585072014E-308-، 0، و 2.2250738585072014E-308 تا 1.7976931348623157E+308

---

[1]) unsigned

[2]) single-precision

[3]) double-precision

- DOUBLE PRECISION[(*M*,*D*)] [UNSIGNED] [ZEROFILL]
- REAL[(*M*,*D*)] [UNSIGNED] [ZEROFILL]

این موارد معادل DOUBLE هستند.

- DECIMAL[(*M*[,*D*])] [UNSIGNED] [ZEROFILL]

عددی ثابت که بصورت رشته ذخیره می‌شود، شبیه ستون CHAR رفتار می‌کند. و یک نویسه برای هر عدد در نظر می‌گیرد.

- DEC[(*M*[,*D*])] [UNSIGNED] [ZEROFILL]
- NUMERIC[(*M*[,*D*])] [UNSIGNED] [ZEROFILL]
- FIXED[(*M*[,*D*])] [UNSIGNED] [ZEROFILL]

شبیه نوع داده DECIMAL هستند.

۸-۲- خلاصه‌ای از انواع داده‌ای زمان و تاریخ

- DATE

یک تاریخ، که محدوده '1000-01-01' تا '9999-12-31' را پشتیبانی می‌کند. MySQL از فرمت 'YYYY-MM-DD' برای نمایش استفاده می‌نماید ولی اجازه قراردادن هر رشته یا عددی را در این ستون می‌دهد.

- DATETIME

ترکیب تاریخ و زمان، که محدوده '1000-01-01 00:00:00' تا '9999-12-31 23:59:59' را پشتیبانی می‌کند. MySQL از فرمت 'YYYY-MM-DD HH:MM:SS' برای نمایش استفاده می‌کند. ولی اجازه استفاده از هر رشته یا عددی را می‌دهد.

- TIMESTAMP[(*M*)]

یک مهر زمانی، که محدوده آن از '1970-01-01 00:00:00' تا سال 2037 است. TIMESTAMP برای ثبت تاریخ و زمان ذخیره و به روز رسانی مفید است.

- TIME

یک زمان، محدوده بین '-838:59:59' و '838:59:59' است. MySQL از فرمت 'HH:MM:SS' برای نمایش این نوع داده استفاده می‌کند. ولی اجازه استفاده از هر عدد و رشته‌ای را می‌دهد.

- YEAR[(2|4)]

یک سال دارای فرمت دو رقمی و یا چهار رقمی است. که پیش فرض آن فرمت چهار رقمی است. و محدوده مجاز در این فرمت 1901 تا 2155 و 0000 است.

۸-۳- خلاصه‌ای از انواع رشته‌ای

- [NATIONAL] CHAR(*M*) [BINARY | ASCII | UNICODE]

یک رشته با طول ثابت، M نشان دهنده طول ستون است که بین 0 تا 255 نویسه می‌تواند باشد.

- CHAR

معادل CHAR(1) است.

- [NATIONAL] VARCHAR(*M*) [BINARY]

یک رشته با طول متغیر. M نشان دهنده حداکثر طول ستون است، و بین 0 تا 255 می‌تواند باشد.

- BINARY(*M*)

شبیه نوع داده CHAR است با این تفاوت که داده‌ها را بصورت دودویی ذخیره می‌کند. این نوع داده به MySQL 4.1.2 اضافه شده است.

- VARBINARY(*M*)

شبیه نوع داده VARCHAR است با این تفاوت که داده‌ها را بصورت دودویی ذخیره می‌کند.

- TINYBLOB
- TINYTEXT

یک ستون TEXT یا BLOB با حداکثر طول (1 - 2^8) 255

- BLOB
- TEXT

یک ستون TEXT یا BLOB با حداکثر طول (1- 2^16) 65,535

- MEDIUMBLOB
- MEDIUMTEXT

یک ستون TEXT یا BLOB با حداکثر طول (1 - 2^24) 16,777,215

- LONGBLOB
- LONGTEXT

یک ستون TEXT یا BLOB با حداکثر طول (1 - 2^32) 4,294,967,295 or 4GB

- ENUM('*value1*','*value2*',...)

یک نوع داده عددی شمارشی. رشته تنها متغیرهای مشخص شده (value1, value2, ..., null) می‌تواند باشد.

SET('*value1*','*value2*',...)

یک مجموعه، یک شی رشته‌ای که می‌تواند صفر یا چند متغیر را شامل شود، که هر یک باید از لیست مشخص شده (value1, value2, ...) انتخاب شوند.

بخش نهم
توابع و عملگرها

توابع و عملگرها در عبارات مختلف SQL استفاده می‌شوند. مانند: ORDER BY، HAVING ، یا عبارت SELECT ، و ... عبارات کلیدی را به هر حالتی (حروف کوچک یا بزرگ) می‌توان بکار برد.

توجه: به طور پیش فرض میان یک تابع و پرانتز جلوی آن نباید فضای خالی وجود داشته باشد. اگر می‌خواهید فضای خالی بعد از نام تابع پذیرفته شود، در زمان اجرای کارساز MySQL از گزینه --sql-mode=IGNORE_SPACE استفاده شود.

بیشتر مثالهای این بخش بصورت کوتاه آورده شده‌اند، به طور مثال:

```
mysql> SELECT MOD(29,9);
+-----------+
| mod(29,9) |
+-----------+
|         2 |
+-----------+
1 rows in set (0.00 sec)
```

جهت اختصار عبارت بالا را بصورت زیر می‌آوریم:

```
mysql> SELECT MOD(29,9);
        -> 2
```

۹-۱- عملگرها

۹-۱-۱- اولویت عملگرها

اولویت عملگرها در MySQL بصورت زیر است، مواردی که در یک خط آورده شده‌اند دارای اولویت یکسان هستند:

```
:=
||, OR, XOR
&&, AND
NOT
BETWEEN, CASE, WHEN, THEN, ELSE
=, <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN
|
&
<<, >>
-, +
*, /, DIV, %, MOD
^
- (unary minus), ~ (unary bit inversion)
!
BINARY, COLLATE
```

۹-۱-۲- پرانتزها

از پرانتز برای مشخص کردن تقدم عملیات استفاده نمایید:

```
mysql> SELECT 1+2*3;
        -> 7
mysql> SELECT (1+2)*3;
        -> 9
```

۹-۱-۳- توابع و عملگرهای مقایسه

نتیجه عملگرهای مقایسه ۱ (TRUE)، 0 (FALSE)، یا NULL است. این عملگرها هم برای اعداد و هم برای رشته‌ها قابل استفاده هستند. MySQL با استفاده از قوانین زیر متغیرها را مقایسه می‌کند:

– اگر یک یا هر دو آرگومان NULL باشند، خروجی NULL خواهد بود.

– اگر هر دو آرگومان مورد مقایسه از نوع رشته باشند، مقایسه بر اساس رشته صورت می‌گیرد.

– اگر هر دو آرگومان مورد مقایسه عدد باشند، مقایسه بر اساس عدد صورت می‌گیرد.

– متغیرهای هگزا دسیمال اگر بصورت عدد قابل مقایسه نباشند، بصورت باینری مقایسه می‌شوند.

– اگر یکی از آرگومانها از نوع TIMESTAMP و یا DATETIME باشد و آرگومان دیگر از نوع ثابت باشد، نوع ثابت قبل از مقایسه به timestamp تبدیل می‌شود.

– در حالتهای دیگر مقایسه بر اساس اعداد حقیقی مقایسه می‌شوند.

– در مقایسه رشته‌ها به طور پیش فرض به حالت نویسه‌ها (کوچک/بزرگ) توجهی نمی‌شود.

چندین مثال:

```
mysql> SELECT 1 > '6x';
        -> 0
mysql> SELECT 7 > '6x';
        -> 1
mysql> SELECT 0 > 'x6';
        -> 0
mysql> SELECT 0 = 'x6';
        -> 1
```

```
mysql> SELECT 1 <=> 1, NULL <=> NULL, 1 <=> NULL;
        -> 1, 1, 0
mysql> SELECT 1 = 1, NULL = NULL, 1 = NULL;
        -> 1, NULL, NULL
```

- <>
- != Not equal:

```
mysql> SELECT '.01' <> '0.01';
        -> 1
mysql> SELECT .01 <> '0.01';
        -> 0
mysql> SELECT 'zapp' <> 'zappp';
        -> 1
```

- <= Less than or equal:

```
mysql> SELECT 0.1 <= 2;
        -> 1
```

- <

Less than:

```
mysql> SELECT 2 < 2;
        -> 0
```

- >=

  Greater than or equal:

```
mysql> SELECT 2 >= 2;
        -> 1
```

- > Greater than:

```
mysql> SELECT 2 > 2;
        -> 0
```

- IS *boolean_value*
- IS NOT *boolean_value*

```
mysql> SELECT 1 IS TRUE, 0 IS FALSE, NULL IS
UNKNOWN;
        -> 1, 1, 1
mysql> SELECT 1 IS NOT UNKNOWN, 0 IS NOT UNKNOWN,
NULL IS NOT UNKNOWN;
        -> 1, 1, 0
```

- IS NULL
- IS NOT NULL

```
mysql> SELECT 1 IS NULL, 0 IS NULL, NULL IS NULL;
        -> 0, 0, 1
mysql> SELECT 1 IS NOT NULL, 0 IS NOT NULL, NULL
IS NOT NULL;
        -> 1, 1, 0
```

- *expr* BETWEEN *min* AND *max*

```
mysql> SELECT 1 BETWEEN 2 AND 3;
        -> 0
mysql> SELECT 'b' BETWEEN 'a' AND 'c';
        -> 1
mysql> SELECT 2 BETWEEN 2 AND '3';
        -> 1
mysql> SELECT 2 BETWEEN 2 AND 'x-3';
        -> 0
```

- COALESCE(*value*,...) اولین عبارت غیر تهی را برمی‌گرداند

```
mysql> SELECT COALESCE(NULL,1);
        -> 1
mysql> SELECT COALESCE(NULL,NULL,NULL);
        -> NULL
```

- GREATEST(*value1*,*value2*,...)

  بزرگترین آرگومان را از میان دو یا چند آرگومان برمی‌گرداند.

```
mysql> SELECT GREATEST(2,0);
        -> 2
mysql> SELECT GREATEST(34.0,3.0,5.0,767.0);
        -> 767.0
mysql> SELECT GREATEST('B','A','C');
        -> 'C'
```

- *expr* IN (*value*,...)

```
mysql> SELECT 2 IN (0,3,5,'wefwf');
        -> 0
mysql> SELECT 'wefwf' IN (0,3,5,'wefwf');
        -> 1
```

- ISNULL(*expr*)

```
mysql> SELECT ISNULL(1+1);
        -> 0
mysql> SELECT ISNULL(1/0);
        -> 1
```

- INTERVAL(*N*,*N1*,*N2*,*N3*,...)

اگر N<N1 صفر برمی‌گرداند، اگر N < N2 یک برمی‌گرداند و الی آخر. اگر N تهی باشد -1 برمی‌گرداند.

```
mysql> SELECT INTERVAL(23, 1, 15, 17, 30, 44,
200);
        -> 3
mysql> SELECT INTERVAL(10, 1, 10, 100, 1000);
        -> 2
mysql> SELECT INTERVAL(22, 23, 30, 44, 200);
        -> 0
```

- LEAST(*value1*,*value2*,...)

بین دو یا چند آرگومان کوچکترین را برمی‌گرداند.

```
mysql> SELECT LEAST(2,0);
        -> 0
mysql> SELECT LEAST(34.0,3.0,5.0,767.0);
        -> 3.0
mysql> SELECT LEAST('B','A','C');
        -> 'A'
```

۹-۱-۴- عملگرهای منطقی

در SQL همه عبارات منطقی با TRUE، FALSE، و NULL ارزیابی می‌شوند. در MySQL بجای این موارد از ۱ (TRUE)، 0 (FALSE)، و NULL استفاده می شود. عملگرهای منطقی استفاده شده در MySQLعبارتند از:

- NOT
- !

اگر نتیجه عملی 0 باشد، یک برمی‌گرداند. و اگر غیر صفر باشد، صفر برمی‌گرداند. اگر NULL باشد، غیر تهی. و اگر غیر تهی باشد، NULL برمی‌گرداند.

```
mysql> SELECT NOT 10;
        -> 0
mysql> SELECT NOT 0;
        -> 1
mysql> SELECT NOT NULL;
        -> NULL
mysql> SELECT ! (1+1);
        -> 0
mysql> SELECT ! 1+1;
        -> 1
```

آخرین عبارت معادل 1+(!1) است و به همین دلیل نتیجه ۱ شده است.

- AND
- &&

اگر همه عوامل غیر صفر و غیر تهی باشند، ۱ برمی‌گرداند. اگر یکی از عوامل 0 باشد، 0 برمی‌گرداند و در غیر اینصورت NULL برمی‌گرداند.

```
mysql> SELECT 1 && 1;
        -> 1
mysql> SELECT 1 && 0;
        -> 0
mysql> SELECT 1 && NULL;
        -> NULL
mysql> SELECT 0 && NULL;
        -> 0
mysql> SELECT NULL && 0;
        -> 0
```

- OR
- ||

اگر یکی از عوامل غیر صفر باشد، ۱ برمی‌گرداند. در غیر اینصورت اگر یکی از عوامل NULL باشد، NULL برمی‌گرداند و در غیر اینصورت صفر برمی‌گرداند.

```
mysql> SELECT 1 || 1;
        -> 1
mysql> SELECT 1 || 0;
        -> 1
mysql> SELECT 0 || 0;
        -> 0
mysql> SELECT 0 || NULL;
        -> NULL
mysql> SELECT 1 || NULL;
        -> 1
```

- XOR

اگر هر یک از عوامل NULL باشند، NULL برمی‌گرداند. برای عوامل غیر NULL ، اگر تعداد یکها عددی فرد باشد، ۱ برمی‌گرداند و در غیر اینصورت 0 برمی‌گرداند.

```
mysql> SELECT 1 XOR 1;
        -> 0
mysql> SELECT 1 XOR 0;
        -> 1
mysql> SELECT 1 XOR NULL;
        -> NULL
mysql> SELECT 1 XOR 1 XOR 1;
        -> 1
```

۹-۲- توابع رشته

اگر طول رشته بزرگتر از مقدار متغیر سیستمی max_allowed_packet باشد، مقدار NULL برگردانده می‌شود. در اینجا تعدادی از توابع مرتبط با رشته‌ها را بیان می‌کنیم.

- ASCII(*str*)

متغیر عددی مربوط به سمت چپ‌ترین نویسه رشته را برمی‌گرداند. اگر رشته خالی باشد، 0 برمی‌گرداند و اگر رشته(ASCII. NULL باشد، NULL برمی‌گرداند(برای نویسه‌هایی که متغیر عددی آنها خارج از 0 تا 255 است)

```
mysql> SELECT ASCII('2');
        -> 50
mysql> SELECT ASCII(2);
        -> 50
mysql> SELECT ASCII('dx');
        -> 100
```

- BIN(*N*)

رشته‌ای که معادل مقدار باینری N است را برمی‌گرداند مقدار N معادل یک عدد BIGINT(longlong) می‌تواند باشد. این تابع معادل CONV(N,10,2) است و اگر مقدار N تهی باشد، NULL برمی‌گرداند.

```
mysql> SELECT BIN(12);
        -> '1100'
```

- BIT_LENGTH(*str*)

طول رشته str را به بیت برمی‌گرداند.

```
mysql> SELECT BIT_LENGTH('text');
        -> 32
```

- CHAR(*N*,...)

آرگومانهای مربوط به هر رشته را بدست می‌آورد و با ترکیب آنها یک رشته برمی‌گرداند.

```
mysql> SELECT CHAR(77,121,83,81,'76');
        -> 'MySQL'
mysql> SELECT CHAR(77,77.3,'77.3');
        -> 'MMM'
```

- CHAR_LENGTH(*str*)

طول رشته str را برمی‌گرداند.

- COMPRESS(*string_to_compress*)

رشته را فشرده می‌کند.

```
mysql> SELECT LENGTH(COMPRESS(REPEAT('a',1000)));
        -> 21
mysql> SELECT LENGTH(COMPRESS(''));
        -> 0
mysql> SELECT LENGTH(COMPRESS('a'));
        -> 13
mysql> SELECT LENGTH(COMPRESS(REPEAT('a',16)));
        -> 15
```

- CONCAT(*str1*,*str2*,...)

نتیجه رشته‌ای است که از به هم پیوستن سایر رشته‌ها به وجود آمده است. اگر همه آرگومانها تهی باشند، NULL برمی‌گرداند.

```
mysql> SELECT CONCAT('My', 'S', 'QL');
        -> 'MySQL'
mysql> SELECT CONCAT('My', NULL, 'QL');
        -> NULL
mysql> SELECT CONCAT(14.3);
        -> '14.3'
```

- `CONCAT_WS(separator,str1,str2,...)`

نوعی خاص از ()CONCAT است با این تفاوت که در رشته نتیجه، آرگومان اول مابین سایر آرگومانها قرار می‌گیرد.

```
mysql> SELECT CONCAT_WS(',','First name','Second
name','Last Name');
        -> 'First name,Second name,Last Name'
mysql> SELECT CONCAT_WS(',','First
name',NULL,'Last Name');
        -> 'First name,Last Name'
```

- `ELT(N,str1,str2,str3,...)`

با توجه به مقدار N ، رشته مربوطه را برمی‌گرداند.

```
mysql> SELECT ELT(1, 'ej', 'Heja', 'hej', 'foo');
        -> 'ej'
mysql> SELECT ELT(4, 'ej', 'Heja', 'hej', 'foo');
        -> 'foo'
```

- `FIELD(str,str1,str2,str3,...)`

ایندکس مربوط به آرگومان اول را در میان سایر آرگومانها برمی‌گرداند.

```
mysql> SELECT FIELD('ej', 'Hej', 'ej', 'Heja',
'hej', 'foo');
        -> 2
mysql> SELECT FIELD('fo', 'Hej', 'ej', 'Heja',
'hej', 'foo');
        -> 0
```

- `FIND_IN_SET(str,strlist)`

ایندکس مربوط به آرگومان اول را در میان سایر آرگومانها برمی‌گرداند.

```
mysql> SELECT FIND_IN_SET('b','a,b,c,d');
        -> 2
```

- `HEX(N_or_S)`

معادل هگزا دسیمال N_or_S را برمی‌گرداند.

```
mysql> SELECT HEX(255);
        -> 'FF'
mysql> SELECT 0x616263;
        -> 'abc'
mysql> SELECT HEX('abc');
        -> 616263
```

- INSERT(*str*,*pos*,*len*,*newstr*)

خروجی، رشته‌ای است که به تعداد len از رشته newstr در موقعیت pos از رشته str قرار گرفته است.

```
mysql> SELECT INSERT('Quadratic', 3, 4, 'What');
        -> 'QuWhattic'
mysql> SELECT INSERT('Quadratic', -1, 4, 'What');
        -> 'Quadratic'
mysql> SELECT INSERT('Quadratic', 3, 100, 'What');
        -> 'QuWhat'
```

- INSTR(*str*,*substr*)

اولین موقعیت رخداد substr را در str برمی‌گرداند.

```
mysql> SELECT INSTR('foobarbar', 'bar');
        -> 4
mysql> SELECT INSTR('xbar', 'foobar');
        -> 0
```

- LCASE(*str*)

این تابع معادل تابع LOWER() است.

- LEFT(*str*,*len*)

به تعداد len از سمت چپ رشته str را نشان می دهد.

```
mysql> SELECT LEFT('foobarbar', 5);
        -> 'fooba'
```

- LENGTH(*str*)

طول رشته str را نشان می‌دهد.

```
mysql> SELECT LENGTH('text');
        -> 4
```

- LOAD_FILE(*file_name*)

فایل را می‌خواند و محتوای فایل را به شکل رشته برمی‌گرداند.

```
mysql> UPDATE tbl_name
           SET
blob_column=LOAD_FILE('/tmp/picture')
           WHERE id=1;
```

- LOCATE(*substr*,*str*)
- LOCATE(*substr*,*str*,*pos*)

مورد اول اولین موقعیت رخداد substr را در str نشان می‌دهد. و مورد دوم موقعیت اولین رخداد substr را در str با شروع از موقعیت pos نشان می دهد.

```
mysql> SELECT LOCATE('bar', 'foobarbar');
        -> 4
mysql> SELECT LOCATE('xbar', 'foobar');
        -> 0
mysql> SELECT LOCATE('bar', 'foobarbar',5);
        -> 7
```

•LOWER(*str*)

نویسه‌های موجود در رشته str را به حالت کوچک برمی‌گرداند.

```
mysql> SELECT LOWER('QUADRATICALLY');
-> 'quadratically'
```

• LPAD(*str*,*len*,*padstr*)

طول str را از len کم کرده و به تعداد آن از متغیر padstr به سمت چپ str اضافه می‌کند. اگر len کوچکتر یا مساوی با طول str باشد، به اندازه len از رشته str نمایش داده می‌شود.

```
mysql> SELECT LPAD('hi',4,'??');
        -> '??hi'
mysql> SELECT LPAD('hi',1,'??');
        -> 'h'
```

• LTRIM(*str*)

فضاهای خالی موجود در ابتدای رشته str را حذف و رشته را نمایش می‌دهد.

```
mysql> SELECT LTRIM('  barbar');
        -> 'barbar'
```

• MID(*str*,*pos*,*len*)

شبیه تابع SUBSTRING(str,pos,len) رفتار می‌کند.

• OCT(*N*)

معادل اکتال متغیر N را برمی‌گرداند.

• ORD(*str*)

اگر نویسه سمت چپ رشته str یک نویسه چند بایتی باشد، کد مربوط به آنرا به روش زیر محاسبه و آن‌را برمی‌گرداند.

```
  (1st byte code)
+ (2nd byte code * 256)
+ (3rd byte code * 256^2) ...
```

ولی اگر نویسه سمت چپ چند بایتی نباشد، معادل کد اسکی آن‌را بر می‌گرداند.

• QUOTE(*str*)

برای نشان دادن رشته‌هایی که در آنها از نمادهای مورد استفاده SQL استفاده شده است.

```
mysql> SELECT QUOTE('Don\'t!');
        -> 'Don\'t!'
mysql> SELECT QUOTE(NULL);
-> NULL
```

- `REPEAT(str,count)`

رشته str را به تعداد count نمایش می‌دهد.

```
mysql> SELECT REPEAT('MySQL', 3);
        -> 'MySQLMySQLMySQL'
```

- `REPLACE(str,from_str,to_str)`

در رشته str ، from_str را با to_str جایگزین می کند.

- `REVERSE(str)`

رشته str را بصورت برعکس نمایش می‌دهد.

```
mysql> SELECT REVERSE('abc');
        -> 'cba'
```

- `RIGHT(str,len)`

به تعداد len از سمت راست str نمایش می‌دهد.

```
mysql> SELECT RIGHT('foobarbar', 4);
        -> 'rbar'
```

- `RPAD(str,len,padstr)`

طول str را از len کم کرده و به تعداد آن از متغیر padstr به انتهای str اضافه می‌کند. اگر len کوچکتر یا مساوی با طول str باشد، به اندازه len از رشته str نمایش داده می‌شود.

```
mysql> SELECT RPAD('hi',5,'?');
        -> 'hi???'
mysql> SELECT RPAD('hi',1,'?');
        -> 'h'
```

- `RTRIM(str)`

فضای خالی موجود در دنباله رشته str را حذف و آنرا نمایش می‌دهد.

```
mysql> SELECT RTRIM('barbar   ');
        -> 'barbar'
```

- `SPACE(N)`

به تعداد N فضای خالی برمی‌گرداند.

```
mysql> SELECT SPACE(6);
->'      '
```

- STRCMP(*expr1*,*expr2*)

اگر دو رشته یکی باشند، صفر برمی‌گرداند. اگر آرگومان اول کوچکتر از آرگومان دوم باشد 1- برمی‌گرداند و اگر آرگومان اول بزرگتر از آرگومان دوم باشد 1 برمی‌گرداند.

```
mysql> SELECT STRCMP('text', 'text2');
        -> -1
mysql> SELECT STRCMP('text2', 'text');
        -> 1
mysql> SELECT STRCMP('text', 'text');
        -> 0
```

- SUBSTRING(*str*,*pos*)
- SUBSTRING(*str* FROM *pos*)
- SUBSTRING(*str*,*pos*,*len*)
- SUBSTRING(*str* FROM *pos* FOR *len*)

تابع فوق بدون آرگومان len یک زیر رشته از رشته str که از موقعیت pos شروع شده است را نمایش می‌دهد. و با آرگومان len زیر رشته‌ای به طول len نمایش داده خواهد شد. عبارت FROM در سینتکس استاندارد SQL استفاده می‌شود.

```
mysql> SELECT SUBSTRING('Quadratically',5);
        -> 'ratically'
mysql> SELECT SUBSTRING('foobarbar' FROM 4);
        -> 'barbar'
mysql> SELECT SUBSTRING('Quadratically',5,6);
        -> 'ratica'
```

- SUBSTRING_INDEX(*str*,*delim*,*count*)

به تعداد count از رشته str با توجه به delim برمی‌گرداند.

```
mysql> SELECT SUBSTRING_INDEX('www.mysql.com',
'.', 2);
        -> 'www.mysql'
mysql> SELECT SUBSTRING_INDEX('www.mysql.com',
'.', -2);
        -> 'mysql.com'
```

- TRIM([{BOTH | LEADING | TRAILING} [*remstr*] FROM] *str*)
- TRIM(*remstr* FROM] *str*)

با توجه به آرگومانهای BOTH /LEADING /TRAILING فضای خالی را از سمت راست/ سمت چپ/ هر دو طرف حذف می‌کند. در صورت مشخص نکردن این آرگومانها حذف از هر دو طرف صورت می‌گیرد. در صورت مشخص کردن remstr بجای فضای خالی آنرا حذف می‌کند.

```
mysql> SELECT TRIM('  bar   ');
        -> 'bar'
mysql> SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx');
        -> 'barxxx'
mysql> SELECT TRIM(BOTH 'x' FROM 'xxxbarxxx');
```

```
        -> 'bar'
mysql> SELECT TRIM(TRAILING 'xyz' FROM 'barxxyz');
        -> 'barx'
```

- `UCASE(str)`

معادل ()UPPER است.

- `UNCOMPRESS(string_to_uncompress)`

رشته‌ای که توسط ()COMPRESS فشرده شده است را غیر فشرده می‌کند.

```
mysql> SELECT UNCOMPRESS(COMPRESS('any string'));
        -> 'any string'
mysql> SELECT UNCOMPRESS('any string');
        -> NULL
```

- `UNHEX(str)`

این تابع متضاد تابع HEX(str) است.

```
mysql> SELECT UNHEX('4D7953514C');
        -> 'MySQL'
mysql> SELECT 0x4D7953514C;
        -> 'MySQL'
mysql> SELECT UNHEX(HEX('string'));
        -> 'string'
mysql> SELECT HEX(UNHEX('1267'));
        -> '1267'
```

- `UPPER(str)`

این تابع رشته str را به حالت بزرگ نمایش می‌دهد.

```
mysql> SELECT UPPER('Hej');
        -> 'HEJ'
```

۹-۳- توابع عدد

۹-۳-۱- عملگرهای حسابی

در این قسمت به طور مختصر عملگرهای حسابی مفید در MySQL را بیان می‌کنیم.

- \+ جمع

```
mysql> SELECT 3+5;
        -> 8
```

- \- تفریق

```
mysql> SELECT 3-5;
        -> -2
```

- \- علامت منفی

```
mysql> SELECT - 2;
        -> -2
```

- * ضرب

```
mysql> SELECT 3*5;
```

```
        -> 15
mysql> SELECT
18014398509481984*18014398509481984.0;
        -> 324518553658426726783156020576256.0
mysql> SELECT 18014398509481984*18014398509481984;
        -> 0
```

- / تقسیم

```
mysql> SELECT 3/5;
        -> 0.60
```

تقسیم به صفر NULL برمی‌گرداند.

```
mysql> SELECT 102/(1-1);
        -> NULL
```

- DIV شبیه FLOOR()

```
mysql> SELECT 5 DIV 2;
        -> 2
```

۹-۴- توابع تاریخ و زمان

در این قسمت با توابعی که بر روی متغیرهای زمانی کار می‌کنند، آشنا می‌شویم. توجه شود که انجام عملیات در این توابع بر اساس تقویم میلادی می‌باشد. با تلاش انجام شده از طرف مرکز تحقیقات صنایع انفورماتیک ایران توابعی معادل نیز برای پشتیبانی از تقویم جلالی تهیه و به منظور ثبت آنها برای گردانندگان MySQL ارسال شده است، امید است در ویرایشهای بعد از 5 از پایگاه داده MySQL توابع مربوط به تقویم جلالی نیز در MySQL موجود و در اختیار کاربران قرار گیرد. قابل ذکر است توابع مربوط به تقویم جلالی، شبیه توابع میلادی بوده با این تفاوت که نویسه J (Jalali) در ابتدای نام تابع قرار می‌گیرد. (در مورد توابعی که برای تقویم جلالی و میلادی رفتاری متفاوت دارند تابع معادل ایجاد شده است) در زیر نمونه‌ای از پرس و جوهایی که از این توابع استفاده می‌کنند را مشاهده می‌کنیم. این پرس و جو تمامی رکوردهایی را که متغیر date_col آنها در ۳۰ روز اخیر است را نمایش می‌دهد.

```
mysql> SELECT something FROM tbl_name
    -> WHERE DATE_SUB(CURDATE(),INTERVAL 30 DAY) <=
date_col;
```

توابع پشتیبانی از تاریخ پایگاه داده MySQL عبارتند از:

- ADDDATE(*date*,INTERVAL *expr type*)
- ADDDATE(*expr*,*days*)

وقتی از INTERVAL در ابتدای ستون دوم استفاده می‌شود، ADDDATE معادل DATE_ADD() خواهد بود. و در مورد SUBDATE() نیز معادل DATE_SUB() خواهد بود.

```
mysql> SELECT DATE_ADD('1998-01-02', INTERVAL 31
DAY);
        -> '1998-02-02'
mysql> SELECT ADDDATE('1998-01-02', INTERVAL 31
DAY);
        -> '1998-02-02'

mysql> SELECT ADDDATE('1998-01-02', 31);
        -> '1998-02-02'
```

- ADDTIME(*expr*,*expr2*)

این تابع عبارت زمانی expr2 را به عبارت زمانی expr اضافه می‌کند.

```
mysql> SELECT ADDTIME('1997-12-31
23:59:59.999999',
    ->                '1 1:1:1.000002');
        -> '1998-01-02 01:01:01.000001'
mysql> SELECT ADDTIME('01:00:00.999999',
'02:00:00.999998');
        -> '03:00:01.999997'
```

- `CONVERT_TZ(dt,from_tz,to_tz)`

متغیر زمانی dt را از ناحیه زمانی from_tz به ناحیه زمانی to_tz تبدیل و خروجی را نمایش می‌دهد.

```
mysql> SELECT CONVERT_TZ('2004-01-01
12:00:00','GMT','MET');
        -> '2004-01-01 13:00:00'
mysql> SELECT CONVERT_TZ('2004-01-01
12:00:00','+00:00','-07:00');
        -> '2004-01-01 05:00:00'
```

- `CURDATE()`

زمان جاری را با فرمت 'YYYY-MM-DD' یا YYYYMMDD نمایش می‌دهد.

```
mysql> SELECT CURDATE();
        -> '1997-12-15'
mysql> SELECT CURDATE() + 0;
        -> 19971215
```

- `CURRENT_DATE`
- `CURRENT_DATE()`

این دو تابع نیز شبیه تابع CURDATE() عمل می‌کنند.

- `CURTIME()`

زمان جاری را با فرمت 'HH:MM:SS'یا HHMMSS نمایش می‌دهد.

```
mysql> SELECT CURTIME();
        -> '23:50:26'
mysql> SELECT CURTIME() + 0;
        -> 235026
```

- `CURRENT_TIME`
- `CURRENT_TIME()`

این دو تابع نیز شبیه تابع CURTIME() عمل می‌کنند.

- `CURRENT_TIMESTAMP`
- `CURRENT_TIMESTAMP()`

این دو تابع شبیه تابع NOW() عمل می‌کنند.

- `DATE(expr)`

بخش تاریخ از عبارت زمانی expr را نمایش می‌دهد.

```
mysql> SELECT DATE('2003-12-31 01:02:03');
        -> '2003-12-31'
-------------------------------------------------------------------
```

- DATEDIFF(*expr*,*expr2*)

تعداد روزهای میان دو تاریخ expr و expr2 را نمایش می‌دهد.

```
mysql> SELECT DATEDIFF('1997-12-31
23:59:59','1997-12-30');
        -> 1
mysql> SELECT DATEDIFF('1997-11-30
23:59:59','1997-12-31');
        -> -31
-------------------------------------------------------------------
```

- DATE_ADD(*date*,INTERVAL *expr type*)
- DATE_SUB(*date*,INTERVAL *expr type*)

این توابع مربوط به عملیات حسابی در تاریخ هستند. date یک تاریخ یا عبارت زمانی است که تاریخ شروع را مشخص می‌کند. expr عبارتی است که مدت زمانی را که باید به تاریخ شروع اضافه و یا از آن کم شود را مشخص می‌کند. رشته expr می‌تواند با علامت منفی (–) همراه باشد. جدول زیر ارتباط آرگومانهای type و expr را نشان می‌دهد.

نوع متغیر	فرمت expr مورد انتظار
MICROSECOND	MICROSECONDS
SECOND	SECONDS
MINUTE	MINUTES
HOUR	HOURS
DAY	DAYS
WEEK	WEEKS
MONTH	MONTHS
QUARTER	QUARTERS
YEAR	YEARS
SECOND_MICROSECOND	'SECONDS.MICROSECONDS'
MINUTE_MICROSECOND	'MINUTES.MICROSECONDS'
MINUTE_SECOND	'MINUTES:SECONDS'
HOUR_MICROSECOND	'HOURS.MICROSECONDS'
HOUR_SECOND	'HOURS:MINUTES:SECONDS'
HOUR_MINUTE	'HOURS:MINUTES'
DAY_MICROSECOND	'DAYS.MICROSECONDS'
DAY_SECOND	'DAYS HOURS:MINUTES:SECONDS'
DAY_MINUTE	'DAYS HOURS:MINUTES'
DAY_HOUR	'DAYS HOURS'
YEAR_MONTH	'YEARS-MONTHS'

بجای استفاده از دو تابع فوق در مواردی می‌توان از عملگرهای + و – نیز استفاده کرد. به مثالهای زیر توجه نمایید:

```
mysql> SELECT '1997-12-31 23:59:59' + INTERVAL 1
SECOND;
        -> '1998-01-01 00:00:00'
mysql> SELECT INTERVAL 1 DAY + '1997-12-31';
        -> '1998-01-01'
mysql> SELECT '1998-01-01' - INTERVAL 1 SECOND;
        -> '1997-12-31 23:59:59'
mysql> SELECT DATE_ADD('1997-12-31 23:59:59',
    ->                 INTERVAL 1 SECOND);
        -> '1998-01-01 00:00:00'
mysql> SELECT DATE_ADD('1997-12-31 23:59:59',
    ->                 INTERVAL 1 DAY);
        -> '1998-01-01 23:59:59'
mysql> SELECT DATE_ADD('1997-12-31 23:59:59',
    ->                 INTERVAL '1:1'
MINUTE_SECOND);
        -> '1998-01-01 00:01:00'
mysql> SELECT DATE_SUB('1998-01-01 00:00:00',
    ->                 INTERVAL '1 1:1:1'
DAY_SECOND);
        -> '1997-12-30 22:58:59'
mysql> SELECT DATE_ADD('1998-01-01 00:00:00',
    ->                 INTERVAL '-1 10' DAY_HOUR);
        -> '1997-12-30 14:00:00'
mysql> SELECT DATE_SUB('1998-01-02', INTERVAL 31
DAY);
        -> '1997-12-02'
mysql> SELECT DATE_ADD('1992-12-31
23:59:59.000002',
    ->             INTERVAL '1.999999'
SECOND_MICROSECOND);
        -> '1993-01-01 00:00:01.000001'
-------------------------------------------------+-------------------
```

- `DATE_FORMAT(`*`date`*`,`*`format`*`)`

فرمت متغیر date را مطابق با رشته format نمایش می‌دهد. جدول زیر موارد قابل استفاده برای format را نشان می‌دهد.

مشخصه	توصیف
%a	نام روزهای هفته بصورت مختصر(Sun..Sat)
%b	نام ماه‌ها بصورت مختصر (Jan..Dec)
%c	عدد ماهها(0..12)
%D	روز از ماه با پسوند انگلیسی(0th, 1st, 2nd, 3rd, ...)
%d	روز از ماه بصورت عددی(00..31)
%e	روز از ماه بصورت عددی(00..31)
%f	یک میلینیوم ثانیه(000000..999999)
%H	ساعت(00..23)

%h	ساعت(01..12)
%I	ساعت(01..12)
%i	دقیقه، عددی(00..59)
%j	روز از سال(001..366)
%k	ساعت(0..23)
%l	ساعت(1..12)
%M	نام ماه(January..December)
%m	ماه ، عددی (00..12)
%p	AMیا PM
%r	(AM or PM به همراه (hh:mm:ss زمان – ۱۲ ساعت
%S	(00..59) ثانیه‌ها
%s	(00..59) ثانیه‌ها
%T	(hh:mm:ss) زمان، ۲۴ ساعت
%U	(00..53) هفته، در مواردی که اولین روز هفته یکشنبه است
%u	(00..53) هفته، در مواردی که اولین روز هفته دوشنبه است
%V	(01..53) استفاده شده است X% هفته، در مواردی که یکشنبه اولین روز هفته است. با
%v	(01..53) استفاده شده است x% هفته، در مواردی که دوشنبه اولین روز هفته است.با
%W	(Sunday..Saturday) نام روزهای هفته
%w	(0=Sunday..6=Saturday) روز از هفته
%X	سال در برابر هفته، عددی، چهار رقم، در مواردی که یکشنبه روز اول هفته است، به همراه V% استفاده می‌شود
%x	سال در برابر هفته، عددی، چهار رقم، در مواردی که دوشنبه روز اول هفته است، به همراه v% استفاده می‌شود
%Y	سال، عددی، چهار رقم
%y	سال، عددی، دو رقم
%%	'%` یک حرف

چندین مثال:

```
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00',
'%W %M %Y');
        -> 'Saturday October 1997'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00',
'%H:%i:%s');
        -> '22:23:00'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00',
                          '%D %y %a %d %m %b %j');
        -> '4th 97 Sat 04 10 Oct 277'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00',
                          '%H %k %I %r %T %S %w');
        -> '22 22 10 10:23:00 PM 22:23:00 00 6'
mysql> SELECT DATE_FORMAT('1999-01-01', '%X %V');
        -> '1998 52'
```

- `DAY(date)`

معادل تابع DAYOFMONTH()

- `DAYOFMONTH(date)`

روز از ماه را در محدوده ۱ تا ۳۱ نمایش می‌دهد.

```
mysql> SELECT DAYOFMONTH('1998-02-03');
        -> 3
```

- `DAYOFWEEK(date)`

ایندکس روز هفته را در تاریخ نمایش می‌دهد. (1 = Sunday, 2 = Monday, ..., 7 = Saturday)

```
mysql> SELECT DAYOFWEEK('1998-02-03');
        -> 3
```

- `DAYOFYEAR(date)`

روز از سال را در محدوده ۱ تا ۳۶۶ نمایش می‌دهد.

```
mysql> SELECT DAYOFYEAR('1998-02-03');
        -> 34
```

- `EXTRACT(type FROM date)`

با توجه به type بخشی از date را نمایش می‌دهد.

```
mysql> SELECT EXTRACT(YEAR FROM '1999-07-02');
        -> 1999
mysql> SELECT EXTRACT(YEAR_MONTH FROM '1999-07-02
01:02:03');
        -> 199907
mysql> SELECT EXTRACT(DAY_MINUTE FROM '1999-07-02
01:02:03');
        -> 20102
mysql> SELECT EXTRACT(MICROSECOND
```

```
    ->                  FROM '2003-01-02
10:30:00.00123');
        -> 123
-----------------------------------------------------------------
```

- FROM_DAYS(*N*)

عدد N را به عنوان روز دریافت و معادل تاریخ آن‌را نمایش می‌دهد.

```
mysql> SELECT FROM_DAYS(729669);
        -> '1997-10-07'
-----------------------------------------------------------------
```

- FROM_UNIXTIME(*unix_timestamp*)
- FROM_UNIXTIME(*unix_timestamp*,*format*)

آرگومان unix_timestamp را با فرمت 'YYYY-MM-DD HH:MM:SS'یا YYYYMMDDHHMMSS نمایش می‌دهد.

```
mysql> SELECT FROM_UNIXTIME(875996580);
        -> '1997-10-04 22:23:00'
mysql> SELECT FROM_UNIXTIME(875996580) + 0;
        -> 19971004222300

-----------------------------------------------------------------
```

- GET_FORMAT(DATE|TIME|DATETIME, 'EUR'|'USA'|'JIS'|'ISO'|'INTERNAL')

فرمت یک رشته را برمی‌گرداند. و انواع آن بصورت زیر است:

فراخوانی تابع	نتیجه
GET_FORMAT(DATE,'USA')	'%m.%d.%Y'
GET_FORMAT(DATE,'JIS')	'%Y-%m-%d'
GET_FORMAT(DATE,'ISO')	'%Y-%m-%d'
GET_FORMAT(DATE,'EUR')	'%d.%m.%Y'
GET_FORMAT(DATE,'INTERNAL')	'%Y%m%d'
GET_FORMAT(DATETIME,'USA')	'%Y-%m-%d- %H.%i.%s'
GET_FORMAT(DATETIME,'JIS')	'%Y-%m-%d %H:%i:%s'
GET_FORMAT(DATETIME,'ISO')	'%Y-%m-%d %H:%i:%s'
GET_FORMAT(DATETIME,'EUR')	'%Y-%m-%d- %H.%i.%s'
GET_FORMAT(DATETIME,'INTERNAL')	'%Y%m%d%H%i%s'
GET_FORMAT(TIME,'USA')	'%h:%i:%s %p'
GET_FORMAT(TIME,'JIS')	'%H:%i:%s'
GET_FORMAT(TIME,'ISO')	'%H:%i:%s'
GET_FORMAT(TIME,'EUR')	'%H.%i.%S'
GET_FORMAT(TIME,'INTERNAL')	'%H%i%s'

```
mysql> SELECT DATE_FORMAT('2003-10-
03',GET_FORMAT(DATE,'EUR'));
        -> '03.10.2003'
mysql> SELECT
STR_TO_DATE('10.31.2003',GET_FORMAT(DATE,'USA'));
        -> 2003-10-31
```

- HOUR(*time*)

بخش ساعت مربوط به time را برمی‌گرداند.

```
mysql> SELECT HOUR('10:05:03');
        -> 10
```

- LAST_DAY(*date*)

آخرین روز از ماه مربوط به date را برمی‌گرداند، اگر date نا معتبر باشد، null برمی‌گرداند.

```
mysql> SELECT LAST_DAY('2003-02-05');
        -> '2003-02-28'
mysql> SELECT LAST_DAY('2004-02-05');
        -> '2004-02-29'
mysql> SELECT LAST_DAY('2004-01-01 01:01:01');
        -> '2004-01-31'
mysql> SELECT LAST_DAY('2003-03-32');
        -> NULL
```

- LOCALTIME
- LOCALTIME()

این دو تابع شبیه تابع NOW() عمل می‌کنند.

- MAKEDATE(*year*,*dayofyear*)

سال و روز از سال را دریافت و معادل تاریخ آنرا برمی‌گرداند.

```
mysql> SELECT MAKEDATE(2001,31),
MAKEDATE(2001,32);
        -> '2001-01-31', '2001-02-01'
mysql> SELECT MAKEDATE(2001,365),
MAKEDATE(2004,365);
        -> '2001-12-31', '2004-12-30'
mysql> SELECT MAKEDATE(2001,0);
        -> NULL
```

- MAKETIME(*hour*,*minute*,*second*)

ساعت، دقیقه، و ثانیه را دریافت و معادل زمان آنرا برمی‌گرداند.

```
mysql> SELECT MAKETIME(12,15,30);
        -> '12:15:30'
```

- MICROSECOND(*expr*)

قسمت مربوط به میکرو ثانیه از عبارت زمانی را برمی‌گردند.

```
mysql> SELECT MICROSECOND('12:00:00.123456');
        -> 123456
mysql> SELECT MICROSECOND('1997-12-31
23:59:59.000010');
        -> 10
```

- `MINUTE(time)`

دقیقه مربوط به زمان را برمی‌گرداند.

```
mysql> SELECT MINUTE('98-02-03 10:05:03');
        -> 5
```

- `MONTH(date)`

ماه مربوط به تاریخ را برمی‌گرداند.

```
mysql> SELECT MONTH('1998-02-03');
        -> 2
```

- `MONTHNAME(date)`

نام ماه مربوط به تاریخ را برمی‌گرداند.

```
mysql> SELECT MONTHNAME('1998-02-05');
        -> 'February'
```

- `NOW()`

زمان جاری را با فرمت 'YYYY-MM-DD HH:MM:SS'یا YYYYMMDDHHMMSS نمایش می‌دهد.

```
mysql> SELECT NOW();
        -> '1997-12-15 23:50:26'
mysql> SELECT NOW() + 0;
        -> 19971215235026
```

- `PERIOD_ADD(P,N)`

N ماه به دوره p (با فرمت YYMMیا YYYYMM) اضافه می‌کند.

```
mysql> SELECT PERIOD_ADD(9801,2);
        -> 199803
```

- `PERIOD_DIFF(P1,P2)`

تعداد ماه میان دو دوره p1 و p2 را برمی‌گرداند. این دو دوره باید با فرمت YYMM یا YYYYMM باشند.

```
mysql> SELECT PERIOD_DIFF(9802,199703);
        -> 11
```

- `QUARTER(date)`

یک چهارم مربوط به سال از تاریخ را برمی گرداند. محدوده آن بین ۱ تا ۴ است.

```
mysql> SELECT QUARTER('98-04-01');
        -> 2
```

```
-----------------------------------------------------------------
```

- SECOND(*time*)

ثانیه مربوط به زمان را برمی‌گرداند.

```
mysql> SELECT SECOND('10:05:03');
    -> 3
-----------------------------------------------------------------
```

- SEC_TO_TIME(*seconds*)

ثانیه دریافت و معادل ساعت، دقیقه و ثانیه آنرا نمایش می‌دهد.

```
mysql> SELECT SEC_TO_TIME(2378);
    -> '00:39:38'
mysql> SELECT SEC_TO_TIME(2378) + 0;
    -> 3938
-----------------------------------------------------------------
```

- STR_TO_DATE(*str*,*format*)

این تابع عکس تابع DATE_FORMAT() رفتار می‌کند.

```
mysql> SELECT STR_TO_DATE('03.10.2003 09.20',
    ->                    '%d.%m.%Y %H.%i');
    -> '2003-10-03 09:20:00'
mysql> SELECT STR_TO_DATE('10arp', '%carp');
    -> '0000-10-00 00:00:00'
mysql> SELECT STR_TO_DATE('2003-15-10 00:00:00',
    ->                    '%Y-%m-%d %H:%i:%s');
    -> NULL
-----------------------------------------------------------------
```

- SUBDATE(*date*,INTERVAL *expr type*)
- SUBDATE(*expr*,*days*)

وقتی آرگومان دوم به همراه INTERVAL بکار می‌رود، این تابع معادل DATE_SUB() خواهد بود.

```
mysql> SELECT DATE_SUB('1998-01-02', INTERVAL 31
DAY);
    -> '1997-12-02'
mysql> SELECT SUBDATE('1998-01-02', INTERVAL 31
DAY);
    -> '1997-12-02'
mysql> SELECT SUBDATE('1998-01-02 12:00:00', 31);
    -> '1997-12-02 12:00:00'
```

- SUBTIME(*expr*,*expr2*)

عبارت دوم را از عبارت اول کم کرده و نتیجه را برمی‌گرداند. expr یک زمان یا عبارت زمانی است و expr2 یک عبارت زمانی است.

```
mysql> SELECT SUBTIME('1997-12-31
23:59:59.999999',
    ->                '1 1:1:1.000002');
    -> '1997-12-30 22:58:58.999997'
mysql> SELECT SUBTIME('01:00:00.999999',
'02:00:00.999998');
    -> '-00:59:59.999999'
-----------------------------------------------------------------
```

- SYSDATE()

این تابع شبیه تابع NOW() است.

- TIME(*expr*)

بخش زمان از یک عبارت زمانی را برمی‌گرداند.

```
mysql> SELECT TIME('2003-12-31 01:02:03');
        -> '01:02:03'
mysql> SELECT TIME('2003-12-31 01:02:03.000123');
        -> '01:02:03.000123'
```

- TIMEDIFF(*expr*,*expr2*)

زمان میان زمان شروع expr و زمان پایان expr2 را برمی‌گرداند.

```
mysql> SELECT TIMEDIFF('2000:01:01 00:00:00',
    ->                 '2000:01:01
00:00:00.000001');
        -> '-00:00:00.000001'
mysql> SELECT TIMEDIFF('1997-12-31
23:59:59.000001',
    ->                 '1997-12-30
01:01:01.000002');
    -> '46:58:57.999999'
```

- TIMESTAMP(*expr*)
- TIMESTAMP(*expr*,*expr2*)

تابع فوق با یک آرگومان، تاریخ یا عبارت زمانی expr را به عنوان یک متغیر زمانی برمی‌گرداند. و با دو آرگومان عبارت زمانی expr2 را به تاریخ یا عبارت زمانی expr اضافه کرده، و نتیجه را برمی‌گرداند.

```
mysql> SELECT TIMESTAMP('2003-12-31');
        -> '2003-12-31 00:00:00'
mysql> SELECT TIMESTAMP('2003-12-31
12:00:00','12:00:00');
        -> '2004-01-01 00:00:00'
```

- TIMESTAMPADD(*interval*,*int_expr*,*datetime_expr*)

عبارت عددی int_expr را به تاریخ یا عبارت زمانی datetime_expr اضافه و نتیجه را برمی‌گرداند.

```
mysql> SELECT TIMESTAMPADD(MINUTE,1,'2003-01-02');
        -> '2003-01-02 00:01:00'
mysql> SELECT TIMESTAMPADD(WEEK,1,'2003-01-02');
        -> '2003-01-09'
```

- TIMESTAMPDIFF(*interval*,*datetime_expr1*,*datetime_expr2*)

تفاوت دو عبارت زمانی datetime_expr1 و datetime_expr2 را بصورت عدد برمی‌گرداند.

```
mysql> SELECT TIMESTAMPDIFF(MONTH,'2003-02-
01','2003-05-01');
        -> 3
mysql> SELECT TIMESTAMPDIFF(YEAR,'2002-05-
01','2001-01-01');
        -> -1
```

```
-----------------------------------------------------------------
```

- `TIME_FORMAT(time,format)`

این تابع شبیه تابع DATE_FORMAT() است با این تفاوت که رشته format تنها می‌تواند مشخص کننده ساعت، دقیقه، و ثانیه باشد.

```
mysql> SELECT TIME_FORMAT('100:00:00', '%H %k %h
%I %l');
        -> '100 100 04 04 4'
-------------------------------------------- -- ------------------
```

- `TIME_TO_SEC(time)`

زمان را دریافت و آنرا به ثانیه برمی‌گرداند.

```
mysql> SELECT TIME_TO_SEC('22:23:00');
        -> 80580
mysql> SELECT TIME_TO_SEC('00:39:38');
        -> 2378
-----------------------------------------------------------------
```

- `TO_DAYS(date)`

تاریخ date را دریافت و شماره روز آنرا از سال 0 برمی‌گرداند.

```
mysql> SELECT TO_DAYS(950501);
        -> 728779
mysql> SELECT TO_DAYS('1997-10-07');
        -> 729669
-----------------------------------------------------------------
```

- `UTC_DATE`
- `UTC_DATE()`

تاریخ UTC جاری را با فرمت 'YYYY-MM-DD'یا YYYYMMDD برمی‌گرداند.

```
mysql> SELECT UTC_DATE(), UTC_DATE() + 0;
        -> '2003-08-14', 20030814
-----------------------------------------------------------------
```

- `UTC_TIME`
- `UTC_TIME()`

زمان UTC جاری را با فرمت 'HH:MM:SS'یا HHMMSS برمی گرداند.

```
mysql> SELECT UTC_TIME(), UTC_TIME() + 0;
        -> '18:07:53', 180753
-----------------------------------------------------------------
```

- `UTC_TIMESTAMP`
- `UTC_TIMESTAMP()`

تاریخ و زمان UTC جاری را با فرمت 'YYYY-MM-DD HH:MM:SS' or YYYYMMDDHHMMSS برمی‌گرداند.

```
mysql> SELECT UTC_TIMESTAMP(), UTC_TIMESTAMP() +
0;
        -> '2003-08-14 18:08:04', 20030814180804
-------------------------------------------- -- ------------------
```

- `WEEK(date[,mode])`

این تابع شماره هفته را در تاریخ نشان می دهد.

```
mysql> SELECT WEEK('1998-02-20');
    -> 7
mysql> SELECT WEEK('1998-02-20',0);
    -> 7
mysql> SELECT WEEK('1998-02-20',1);
    -> 8
mysql> SELECT WEEK('1998-12-31',1);
    -> 53
```

- `WEEKDAY(date)`

ایندکس روز هفته را در تاریخ برمی‌گرداند(دوشنبه = 0 ، سه شنبه = 1 ، ... یکشنبه = 6)

```
mysql> SELECT WEEKDAY('1998-02-03 22:23:00');
    -> 1
mysql> SELECT WEEKDAY('1997-11-05');
    -> 2
```

- `WEEKOFYEAR(date)`

شماره هفته را در تاریخ date برمی‌گرداند.

```
mysql> SELECT WEEKOFYEAR('1998-02-20');
    -> 8
```

- `YEAR(date)`

سال را از تاریخ date برمی‌گرداند.

```
mysql> SELECT YEAR('98-02-03');
    -> 1998
```

- `YEARWEEK(date)`
- `YEARWEEK(date,start)`

سال و هفته را از تاریخ date برمی‌گرداند.

```
mysql> SELECT YEARWEEK('1987-01-01');
    -> 198653
```

MySQL علاوه بر توابع ذکر شده دارای توابع دیگری نیز است، که بررسی آنها به خواننده واگذار شده است.

بخش دهم
ترکیب عبارات MySQL

در این بخش با ساختار عبارات پشتیبانی شونده توسط MySQL آشنا می‌شویم.

۱۰-۱- عبارات دستکاری[1] داده

۱۰-۱-۱ - ساختار DELETE

برای یک جدول

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name
       [WHERE where_definition]
       [ORDER BY ...]
       [LIMIT row_count]
```

برای چندین جدول

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
       tbl_name[.*] [, tbl_name[.*] ...]
       FROM table_references
       [WHERE where_definition]
```

و یا

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
       FROM tbl_name[.*] [, tbl_name[.*] ...]
       USING table_references
       [WHERE where_definition]
```

عبارت DELETE ردیفهایی از جدول که دارای شرایطی خاص هستند را پاک می‌کند. اگر در استفاده آن از عبارت WHERE استفاده نشود، همه ردیفها پاک خواهند شد. مثال:

mysql> DELETE FROM *tbl_name* WHERE 1>0;

عبارت DELETE موارد زیر را پشتیبانی می‌کند:

- اگر کلمه کلیدی LOW_PRIORITY مشخص شود، اجرای DELETE تا زمانی که کاربران در حال خواندن جدول هستند به تعلیق می‌افتد.
- کلمه کلیدی IGNORE باعث می‌شود از همه خطاها در فرایند پاک کردن ردیفها صرف نظر کرده و فرایند پاک کردن ادامه یابد.
- اگر عبارت DELETE به همراه ORDER BY استفاده شود، ردیفها به ترتیبی که در ORDER BY مشخص شده است، پاک می‌شوند.

```
DELETE FROM somelog
WHERE user = 'jcole'
ORDER BY timestamp
LIMIT 1
```

پاک کردن از چندین جدول:

```
DELETE t1, t2 FROM t1, t2, t3 WHERE t1.id=t2.id AND
t2.id=t3.id;
```

۱۰-۱-۲- ساختار DO

```
DO expr [, expr] ...
```

[1] Manipulation

DO یکسری عبارات را اجرا می‌کند ولی هیچ نتیجه‌ای برنمی‌گرداند.

۱۰-۱-۳- ساختار HANDLER

```
HANDLER tbl_name OPEN [ AS alias ]
HANDLER tbl_name READ index_name { = | >= | <= | < }
(value1,value2,...)
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ index_name { FIRST | NEXT | PREV |
LAST }
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ { FIRST | NEXT }
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name CLOSE
```

این عبارت امکان دسترسی مستقیم به واسط موتور ذخیره جداول را می‌دهد. این امکان برای جداول MyISAM در MySQL 4.0.0 و برای جداول InnoDB در MySQL 4.0.3 موجود است.

۱۰-۱-۴- ساختار INSERT

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    VALUES ({expr | DEFAULT},...),(...),...
    [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

یا :

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    SET col_name={expr | DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

یا

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    SELECT ...
```

INSERT یک ردیف جدید به جدول موجود اضافه می‌کند. tbl_name نام جدولی است که ردیفها باید در آن ذخیره شوند. اگر از عبارت ON DUPLICATE KEY UPDATE به همراه INSERT استفاده نمایید، به منزله یک نسخه دوم از یک ردیف است و با وجود واحد بودن و وجود کلید اولیه، ردیف جدید جایگزین ردیف قبلی خواهد شد.

```
mysql> INSERT INTO table (a,b,c) VALUES (1,2,3)
    -> ON DUPLICATE KEY UPDATE c=c+1;

mysql> UPDATE table SET c=c+1 WHERE a=1;

mysql> INSERT INTO table (a,b,c) VALUES (1,2,3),(4,5,6)
    -> ON DUPLICATE KEY UPDATE c=VALUES(a)+VALUES(b);
```

با استفاده از عبارت SELECT .... INSERT امکان ذخیره سریع چندین ردیف از یک یا چند جدول به درون یک جدول وجود دارد. و ساختار آن بصورت زیر است:

```
INSERT [LOW_PRIORITY] [IGNORE] [INTO] tbl_name
[(column_list)]
    SELECT ...
```

مثال:

```
INSERT INTO tbl_temp2 (fld_id)
    SELECT tbl_temp1.fld_order_id
    FROM tbl_temp1 WHERE tbl_temp1.fld_order_id > 100;
```

برای کارخواه‌هایی که نمی‌توانند تا زمان انجام کامل فرایند INSERT منتظر بمانند، از عبارت INSERT DELAYED استفاده می‌شود.

```
INSERT DELAYED ...
```

۱۰-۱-۵- ساختار LOAD DATA INFILE

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE
'file_name.txt'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [FIELDS
        [TERMINATED BY '\t']
        [[OPTIONALLY] ENCLOSED BY '']
        [ESCAPED BY '\\' ]
    ]
    [LINES
        [STARTING BY '']
        [TERMINATED BY '\n']
    ]
    [IGNORE number LINES]
    [(col_name,...)]
```

عبارت LOAD DATA INFILE ردیفها را از فایل خوانده و با سرعت خیلی زیاد در جدول ذخیره می‌کند.

```
mysql> USE db1;
mysql> LOAD DATA INFILE 'data.txt' INTO TABLE my_table;
```

و برای ذخیره اطلاعات به جدولی موجود در پایگاه داده‌ای دیگر خواهیم داشت.

```
mysql> USE db1;
mysql> LOAD DATA INFILE 'data.txt' INTO TABLE
db2.my_table;
```

برای عدم ذخیره یکسری از خطوط اولیه موجود در فایل در جدول مربوطه می‌توان از عبارت IGNORE number LINES استفاده کرد. بطور نمونه مثال زیر فایل test.txt بدون خط اول آن را در جدول test ذخیره می‌کند.

```
mysql> LOAD DATA INFILE '/tmp/test.txt'
    -> INTO TABLE test IGNORE 1 LINES;
```

زمانی که با استفاده از عبارت SELECT ... INTO OUTFILE داده‌های جدول در فایل ذخیره می‌شود، هدر ستونها نیز به همراه داده‌های آنها به عنوان سطر اول در فایل ذخیره می‌شود، و زمانی که می‌خواهید این داده‌های موجود در فایل را به درون جدولی مشابه برگردانید، برای جلوگیری از ذخیره نام ستون‌های مربوط به جدول قبلی به درون جدول جدید می‌توان از IGNORE number استفاده کرد و از سطر اول صرف نظر نمود.

۱۰-۱-۶- ساختار REPLACE

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name [(col_name,...)]
    VALUES ({expr | DEFAULT},...),(...),...
```

یا :

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name
    SET col_name={expr | DEFAULT}, ...
```

یا :

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name [(col_name,...)]
    SELECT ...
```

REPLACE دقیقا شبیه INSERT عمل می‌کند، به این صورت که رکورد قدیمی را پاک و یک رکورد جدید جایگزین آن می‌کند.

۱۰-۱-۷- ساختار SELECT

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT]
[SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr, ...
    [INTO OUTFILE 'file_name' export_options
      | INTO DUMPFILE 'file_name']
    [FROM table_references
      [WHERE where_definition]
      [GROUP BY {col_name | expr | position}
        [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_definition]
      [ORDER BY {col_name | expr | position}
        [ASC | DESC] , ...]
      [LIMIT {[offset,] row_count | row_count OFFSET
offset}]
      [PROCEDURE procedure_name(argument_list)]
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

SELECT برای بازیافت ردیفهای انتخاب شده از یک یا چند جدول بکار می‌رود. آن همچنین می‌تواند برای محاسبه عبارات محاساباتی بدون ارجاع به هیچ جدولی استفاده شود.

```
mysql> SELECT 1 + 1;
        -> 2
```

می‌توان به جداول به صورت tbl_name AS alias_name آلیاس نسبت داد.

```
•   mysql> SELECT t1.name, t2.salary FROM employee AS
    t1, info AS t2
•       ->     WHERE t1.name = t2.name;
•   mysql> SELECT t1.name, t2.salary FROM employee
    t1, info t2
•       ->     WHERE t1.name = t2.name;
```

ستونهای مشخص شده برای خروجی می‌توانند به ORDER BY و GROUP BY ارجاع داده شوند.

```
•   mysql> SELECT college, region, seed FROM
    tournament
•       ->     ORDER BY region, seed;
•   mysql> SELECT college, region AS r, seed AS s
    FROM tournament
```

```
    ->     ORDER BY r, s;
mysql> SELECT college, region, seed FROM
tournament
    ->     ORDER BY 2, 3;
```

از ORDER BY برای مرتب سازی داده‌ها استفاده می‌شود، برای اینکه داده‌ها بصورت معکوس مرتب شوند، از DESC استفاده نمایید. در مواردی که از یک تابع گروهی استفاده می‌شود، برای مرتب سازی باید از GROUP BY استفاده شود.

```
mysql> SELECT student_name, AVG(test_score)
    ->        FROM student
    ->        GROUP BY student_name;
```

عبارت HAVING می تواند به هر ستون یا آلیاسی در عبارت SELECT ارجاع داده شود، و عملکرد آن شبیه عبارت WHERE است. در مواردی که باید از WHERE استفاده شود، از HAVING نباید استفاده کرد، بطور مثال استفاده از آن بصورت زیر اشتباه است:

```
mysql> SELECT col_name FROM tbl_name HAVING
col_name > 0;
```

و باید بصورت زیر استفاده شود:

```
mysql> SELECT col_name FROM tbl_name WHERE
col_name > 0;
```

در واقع عبارت HAVING برای ارجاع به توابع اجتماع که WHERE نمی‌تواند بکار رود، استفاده می‌شود:

```
mysql> SELECT user, MAX(salary) AS max_salary
FROM users
    ->     GROUP BY user HAVING max_salary>10;
```

از عبارت LIMIT می‌توان برای محدود کردن تعداد ردیفهای برگردانده شده توسط SELECT استفاده کرد.

بطور مثال عبارت زیر ردیفهای ۶ تا ۱۵ را برمی‌گرداند.

```
mysql> SELECT * FROM table LIMIT 5,10;
```

عبارت زیر ۵ ردیف اول را برمی‌گرداند.

```
mysql> SELECT * FROM table LIMIT 5;
```

۱۰-۱-۸- ساختار Subquery

Subquery یک عبارت SELECT است که در یک عبارت دیگر بکار می‌رود.

مثال:

```
SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM
t2);
```

۱۰-۱-۱۰- ساختار UPDATE

برای یک جدول:

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
    SET col_name1=expr1 [, col_name2=expr2 ...]
    [WHERE where_definition]
    [ORDER BY ...]
```

```
    [LIMIT row_count]
```

برای چندین جدول

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name [, tbl_name ...]
    SET col_name1=expr1 [, col_name2=expr2 ...]
    [WHERE where_definition]
```

UPDATE داده‌های موجود در ستونهای ردیفهای مشخص را با متغیرهای جدید به روز می‌کند. عبارت SET اشاره به ستونهایی دارد که باید به روز شوند. و با استفاده از عبارت WHERE ردیفهایی که قرار است به روز شوند مشخص می‌شود. عبارت ORDER BY ترتیب به روز رسانی را مشخص می‌کند. و LIMIT محدودیت تعداد ردیفهایی را که می‌توانند به روز شوند را مشخص می‌کند.

```
mysql> UPDATE persondata SET age=age+1;
mysql> UPDATE persondata SET age=age*2, age=age+1;
```

۱۰-۲- عبارات تعریف داده

۱۰-۲-۱- ساختار ALTER DATABASE

```
ALTER {DATABASE | SCHEMA} db_name
    alter_specification [, alter_specification] ...

alter_specification:
    [DEFAULT] CHARACTER SET charset_name
  | [DEFAULT] COLLATE collation_name
```

ALTER DATABASE به شما اجازه تغییر اطلاعات کلی مربوط به پایگاه داده را می‌دهد. این مشخصات در فایل 'db.opt` موجود در شاخه پایگاه داده ذخیره میشوند. برای استفاده از ALTER DATABASE شما نیاز به مجوز تغییر پایگاه داده دارید.

۱۰-۲-۲- ساختار ALTER TABLE

```
ALTER [IGNORE] TABLE tbl_name
    alter_specification [, alter_specification] ...
```

alter_specification می‌تواند شامل موارد زیر باشد:

```
    ADD [COLUMN] column_definition [FIRST | AFTER
col_name ]
  | ADD [COLUMN] (column_definition,...)
  | ADD INDEX [index_name] [index_type]
(index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
        PRIMARY KEY [index_type] (index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
        UNIQUE [index_name] [index_type]
(index_col_name,...)
  | ADD [FULLTEXT|SPATIAL] [index_name]
(index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
        FOREIGN KEY [index_name] (index_col_name,...)
        [reference_definition]
  | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP
DEFAULT}
  | CHANGE [COLUMN] old_col_name column_definition
```

```
        [FIRST|AFTER col_name]
  | MODIFY [COLUMN] column_definition [FIRST | AFTER
col_name]
  | DROP [COLUMN] col_name
  | DROP PRIMARY KEY
  | DROP INDEX index_name
  | DROP FOREIGN KEY fk_symbol
  | DISABLE KEYS
  | ENABLE KEYS
  | RENAME [TO] new_tbl_name
  | ORDER BY col_name
  | CONVERT TO CHARACTER SET charset_name [COLLATE
collation_name]
  | [DEFAULT] CHARACTER SET charset_name [COLLATE
collation_name]
  | DISCARD TABLESPACE
  | IMPORT TABLESPACE
  | table_options
```

ALTER TABLE به شما اجازه تغییر ساختار جداول موجود را می‌دهد. بطور مثال شما می‌توانید ستونی را از/به جداول موجود حذف/اضافه نمایید.

- ```
  mysql> ALTER TABLE t2 DROP COLUMN c, DROP COLUMN
  d;
  ```

۱۰-۲-۳- ساختار ALTER VIEW

```
ALTER [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

این ساختار برای تغییر تعریف یک نمای موجود استفاده می‌شود.

۱۰-۲-۴- ساختار CREATE DATABASE

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_specification [, create_specification] ...]

create_specification:
    [DEFAULT] CHARACTER SET charset_name
  | [DEFAULT] COLLATE collation_name
```

برای ایجاد یک پایگاه داده جدید استفاده می‌شود.

مثال: پایگاه داده‌ای برای ذخیره داده‌های فارسی

```
mysql > CREATE DATABASE db_name
    -> DEFAULT CHARACTER SET utf8 DEFAULT
COLLATE utf8_persian_ci;
```

۱۰-۲-۵- ساختار CREATE TABLE

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)]
    [table_options] [select_statement]
```

یا:
```

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [() LIKE old_tbl_name ()];
```

برای ایجاد جدول از این ساختار استفاده می‌شود. مثال زیر جدولی با نام tbl با سه ستون ایجاد می‌کند.

```
CREATE TABLE tbl (firstname VARCHAR(20), lastname
VARCHAR(30), age TINYINT);
```

مثال زیر جدولی با یک ستون که مجموعه نویسه و تابع تطبیق برای آن ستون مشخص شده است را ایجاد می‌کند.

```
CREATE TABLE t (c CHAR(20) CHARACTER SET utf8 COLLATE
utf8_bin);
```

۱۰-۲-۶- ساختار CREATE VIEW

```
CREATE [OR REPLACE] [ALGORITHM = {UNDEFINED | MERGE |
TEMPTABLE}]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

این عبارت یک نمای جدید ایجاد می کند، یا با استفاده از عبارت OR REPLACE آنرا با یک نمای موجود جایگزین می کند.

مثال:

```
mysql> CREATE TABLE t (qty INT, price INT);
mysql> INSERT INTO t VALUES(3, 50);
mysql> CREATE VIEW v AS SELECT qty, price, qty*price AS
value FROM t;
mysql> SELECT * FROM v;
+------+-------+-------+
| qty  | price | value |
+------+-------+-------+
|    3 |    50 |   150 |
+------+-------+-------+
```

۱۰-۲-۷- ساختار DROP DATABASE

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```

DROP DATABASE همه جداول موجود در پایگاه داده را پاک کرده و سپس خود پایگاه داده را نیز پاک می‌کند.

۱۰-۲-۸- ساختار DROP TABLE

```
DROP [TEMPORARY] TABLE [IF EXISTS]
    tbl_name [, tbl_name] ...
    [RESTRICT | CASCADE]
```

DROP TABLE یک یا چند جدول را پاک می‌کند.

۱۰-۲-۹- ساختار DROP VIEW

```
DROP VIEW [IF EXISTS]
    view_name [, view_name] ...
    [RESTRICT | CASCADE]
```

DROP VIEW یک یا چند نما را پاک می‌کند.