Linux Professional Institute

LPIC-1

جلسه سیزدهم: مفاهیم اولیه‌ی ایمیل و دیتابیس

By: The Albatross

thealbatross@yandex.com

https://github.com/TheAlbatrossCodes/Linux-In-Persian

فهرست مطالب

مقدمه

جلسه قبل، با مقدمات اسکریپتینگ آشنا شدیم. پس از آن در مورد چگونگی اجرای اسکریپت‌ها در پشت صحنه و زمان‌بندی اسکریپت‌ها صحبت کردیم. در این جلسه، در مورد سرورهای ایمیل و چگونگی عملکرد آنها صحبت می‌کنیم و در نهایت، با چگونگی نصب و مقدمات استفاده از دیتابیس آشنا خواهیم شد.

ایمیل

ایمیل یکی از مهم‌ترین سرویس‌های موجود در لینوکس می‌باشد. اکثر سیستم‌های لینوکسی، حتی وقتی که به اینترنت متصل نباشند، از ایمیل برای اطلاع‌رسانی به کاربر استفاده می‌کنند. برای مثال، سرویس‌هایی نظیر `cron` و `at` با ارسال یک ایمیل به کاربر، او را از انجام کارها مطلع می‌سازند. به همین دلیل، در اکثر توزیع‌های لینوکسی یک سرویس ایمیل به صورت پیش‌فرض تنظیم و در حال اجرا می‌باشد. پس می‌توان گفت که آشنایی با ایمیل و چگونگی مدیریت آن بسیار پراهمیت می‌باشد.

درک عملکرد ایمیل

قبل از صحبت در مورد سرورهای ایمیل در لینوکس، بهتر است در مورد چگونگی عملکرد ایمیل در لینوکس صحبت کنیم. سیستم‌عامل‌های لینوکسی از فلسفه‌ی یونیکس برای مدیریت ایمیل پیروی می‌کنند. یکی از نوآوری‌های یونیکس در مدیریت ایمیل، ماژولار کردن نرم‌افزارهای پردازش ایمیل بود. این یعنی که در سیستم به جای وجود یک برنامه‌ی یکپارچه برای ارسال و دریافت ایمیل، چندین برنامه‌ی کوچک که با یکدیگر برای پردازش پیام‌ها کار می‌کنند وجود خواهد داشت.

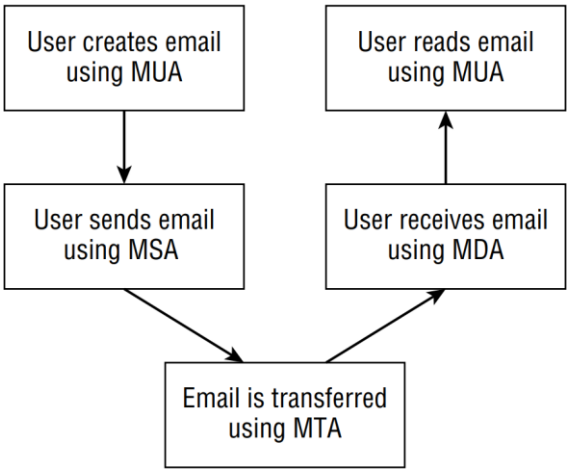
چندین پروتکل برای مدیریت ایمیل وجود دارد. معروف‌ترینِ این پروتکل‌ها، Simple Mail Transfer Protocol یا SMTP نام دارد. این پروتکل، یک پروتکل Push می‌باشد؛ بدین معنی که سیستم ارسال کننده عملیات انتقال پیام را آغاز می‌کند. این پروتکل بسیار برای ارسال اطلاعات مناسب می‌باشد، به همین دلیل در بیشتر مراحل موجود در فرآیند ارسال ایمیل، از پروتکل SMTP استفاده می‌شود. در مرحله‌ی نهایی ایمیل، از یک پروتکل Pull، مانند Post Office Protocol که به آن POP می‌گویند یا Internet Message Access Protocol که به آن IMAP می‌گویند استفاده می‌شود. در این پروتکل‌ها، سیستم دریافت کننده‌ی پیام عملیات انتقال را آغاز می‌کند.

سیستم‌های ایمیل لینوکسی، معمولا به سه بخش تقسیم می‌شوند:

- مامور انتقال ایمیل یا Mail Transfer Agent (MTA)، ایمیل‌هایی که به ما ارسال می‌شود (یا ایمیل‌هایی که ما به یک گیرنده‌ی لوکال ارسال می‌کنیم) را به یک مامور تحویل ایمیل یا Mail Delivery Agent (MDA) می‌فرستد. اگر ما ایمیلی را به یک گیرنده‌ی غیرلوکال بفرستیم، MTA سیستم ما با MTA موجود در سیستم گیرنده برای انتقال ایمیل ارتباط برقرار می‌کنند.
- مامور تحویل ایمیل یا Mail Delivery Agent (MDA) برنامه‌ای است که ایمیل‌ها را به صندوق پستی یک کاربر محلی تحویل می‌دهد. به عبارت دیگر، MDA فقط در سیستم گیرنده دارای وظیفه‌ای می‌باشد.

- مامور مشاهده‌ی ایمیل یا Mail User Agent (MUA) یک اینترفیس می‌باشد که کاربران سیستم برای مشاهده‌ی ایمیل‌های موجود در صندوق پستی خود از آن استفاده می‌کنند. MUAها پیامی دریافت نمی‌کنند، بلکه فقط محتویات موجود در صندوق پستی کاربر را به او نشان می‌دهند. اکثر MUAها قابلیت ایجاد یک پیام و ارسال آن به MTA را نیز دارا می‌باشند.

شاید تا اینجا کمی گیج شده باشید، پس بیایید با هم بازیگرها و مراحل موجود از زمان ایجاد ایمیل تا زمان تحویل ایمیل به یک گیرنده را بررسی کنیم:



تصویر ۱ - نمایی از چگونگی ایجاد، انتقال و تحویل ایمیل

۱- کاربر با استفاده از یک کلاینت ایمیل (MUA)، یک ایمیل ایجاد می‌کند.
۲- MUA ایمیل ایجاد شده توسط کاربر را به برنامه‌ی واگذاری ایمیل، یا Mail Submission Agent (MSA) می‌دهد. MSA معمولا بخشی از MTA می‌باشد.
۳- MSA ایمیل را تحویل برنامه‌ی انتقال ایمیل (MTA) می‌دهد.
٤- MTA وظیفه‌ی انتقال ایمیل به برنامه‌ی تحویل ایمیل در مقصد (MDA) را بر عهده دارد.
۵- MDA عملیات لازم برای انتقال ایمیل به MUA مقصد را انجام می‌دهد.
۶- گیرنده‌ی ایمیل از طریق MUA، ایمیل دریافتی را می‌خواند.

برنامه‌های ایمیل در لینوکس

مانند سایر برنامه‌ها در لینوکس، ما آپشن‌های زیادی برای انتخاب یک MTA داریم. سه تا از معروف‌ترین MTAها به شرح زیر می‌باشند:

- **Sendmail**

برنامه‌ی Sendmail، به دلیل تطبیق‌پذیری و انعطاف‌پذیری بالای خود، در زمان‌های قدیم یکی از معروف‌ترین برنامه‌های MTA موجود در لینوکس بود. برخی از ویژگی‌های موجود در این برنامه، نظیر Mail Listها، User Aliasها و... تبدیل به بخش‌های جداناشدنی هر سیستم ایمیل شده‌اند.
با این حال، همه‌کاره‌بودن سبب پیچیدگی بیش از حد این برنامه شده است. فایل تنظیمات Sendmail معمولا بسیار بزرگ و طولانی می‌باشد و این امر، مدیریت و نگهداری از این برنامه را بسیار دشوار می‌کند. در سال ۲۰۰۵، چندین مشکل امنیتی برای برنامه‌ی Sendmail به وجود آمد و این امر باعث شد که Sendmail محبوبیت خود را از دست دهد.

- **Postfix**

این برنامه توسط یک برنامه‌نویس و مهندس امنیت در شرکت IBM به نام Wietse Venema ایجاد شده است. Postfix به عنوان جایگزینی برای برنامه‌ی Sendmail توسعه داده شده و دارای طراحی ماژولار می‌باشد، به طوری که هر ماژول موجود در این برنامه، یک وظیفه‌ی خاص را بر عهده دارد. استفاده از معماری ماژولار، باعث بهبود امنیت می‌شود. تنظیم و نگهداری Postfix بسیار ساده‌تر از Sendmail می‌باشد و امروزه در اکثر توزیع‌های لینوکسی، به عنوان نرم‌افزار پیش‌فرض ایمیل نصب شده است.

- **Exim**

این برنامه در سال ۱۹۹۵ توسط Philip Hazel در دانشگاه کمبریج توسعه داده شده است. این برنامه نیز مانند Sendmail، یک برنامه‌ی بزرگ می‌باشد (ماژولار نیست)؛ با این تفاوت که فایل تنظیمات آن بسیار ساده‌تر از Sendmail می‌باشد و در نتیجه تنظیم و نگهداری آن بسیار ساده‌تر می‌باشد. این برنامه بسیار انعطاف‌پذیر می‌باشد و در برخی از توزیع‌های لینوکسی به عنوان نرم‌افزار پیش‌فرض ایمیل نصب شده است.

کارکردن با ایمیل

یادگرفتن چگونگی ارسال ایمیل، مشاهده‌ی ایمیل‌های دریافتی، بررسی صف ایمیل و همچنین Forward کردن ایمیل‌ها از اهمیت بالایی برخوردار می‌باشد. در این بخش، به توضیح این موارد می‌پردازیم.

ارسال و دریافت ایمیل

یکی از معروف‌ترین برنامه‌های MDA در لینوکس، برنامه‌ی `binmail` می‌باشد. علت معروف بودن این برنامه، سادگی آن می‌باشد. به صورت پیش‌فرض، این برنامه پیام‌های ذخیره شده در `/var/spool/mail` را می‌خواند، اما می‌توانیم با استفاده از آپشن‌های این برنامه، این موقعیت را تغییر دهیم.

سادگی این برنامه، به معنای این می‌باشد که این برنامه قابلیت‌ها و ویژگی‌های کمی دارد و همین امر باعث شده که در سال‌های اخیر، از محبوبیت این برنامه کاسته شود. به همین دلیل، این نرم‌افزار دیگر به صورت پیش‌فرض در اکثر توزیع‌های لینوکسی نصب نمی‌باشد و ما باید آن را به صورت دستی نصب کنیم. توجه کنید که برای نصب این برنامه در توزیع CentOS، باید از نام `mailx` استفاده کنیم:

```
[root@localhost ~]# yum install mailx
[...]
Installed:
  mailx.x86_64 0:12.5-19.el7

Complete!
```

به طور کلی، ما به صورت زیر از این برنامه استفاده می‌کنیم:

```
mail [OPTIONS] recipient...
```

به طوری که *recipient* می‌تواند یوزرنیم یک کاربر در سیستم باشد، یا می‌تواند یک آدرس ایمیل کامل، مثل test@example.com، باشد. توجه داشته باشید که ما می‌توانیم از بیش از یک *recipient* استفاده کنیم و فقط کافی است آنها را با یک Space از هم جدا کنیم.

کاربردی‌ترین آپشن‌هایی که می‌توانیم در قسمت OPTIONS قرار دهیم به شرح زیر می‌باشند:

- `-s subject`

یک Subject به ایمیل اضافه می‌کند. اگر در `subject` نوشته شده توسط ما فاصله‌ای وجود داشته باشد، باید آن را بین دو علامت " قرار دهیم.

- `-cc recipient`

با استفاده از این آپشن، می‌توانیم گیرنده یا گیرنده‌هایی که باید یک کپی از ایمیل ما را دریافت کنند را مشخص کنیم. کلیه‌ی گیرنده‌های ایمیل، می‌توانند آدرس این `recipient`ها را مشاهده کنند.

- `-bc recipient`

با استفاده از این آپشن، می‌توانیم گیرنده یا گیرنده‌هایی که باید یک کپی از ایمیل ما را دریافت کنند را مشخص کنیم. در این حالت، فقط فرستنده می‌تواند آدرس این `recipient`ها را مشاهده کند.

- `-v`

با استفاده از این آپشن می‌توانیم اطلاعات مربوط به تحویل‌دهی ایمیل را مشاهده کنیم.

بیایید با استفاده از برنامه‌ی `mail`، یک ایمیل به یکی از کاربرهای محلی موجود در سیستم خود بفرستیم:

```
[root@localhost ~]# mail -s "Some bad news" puppy
I've got some bad news for you. This Linux shit you've
been writing for the past year? No one will ever read this.
No one. Including the one who suggested doing it in the first place.

This, like many other projects before it, will join the shitbin of
history and no one will ever care or remember that these documents ever
existed.

It's sad, but it's reality.

Yours truly,
Reality
EOT
```

همانطور که می‌بینید، ما در اینجا ابتدا دستور `mail` را وارد کردیم، سپس آپشن `-s` را به آن دادیم و Subject مورد نظر برای ایمیل را مشخص کردیم. همانطور که می‌بینید، از آنجایی که Subject مورد نظر ما دارای فاصله بود، آن را بین دو علامت " قرار دادیم. پس از مشخص کردن Subject، نام کاربری که می‌خواستیم ایمیل را دریافت کند را مشخص کردیم (puppy). از آنجایی که این کاربر یک کاربر محلی می‌باشد، وارد کردن یوزرنیم او کافی است و نیازی به استفاده از یک آدرس ایمیل کامل نیست.

پس از مشخص کردن این موارد و زدن دکمه‌ی Enter، برنامه یک خط خالی روی ترمینال ایجاد کرده و به ما اجازه می‌دهد که پیام خود را در اینجا بنویسیم. وقتی که پیام خود را نوشتیم و آماده به ارسال آن بودیم، کافی است دکمه‌ی Ctrl + D را بزنیم تا این برنامه ایمیل را ارسال کرده و به ترمینال باز گردیم. به محض زدن این دکمه، یک عبارت EOT (End of Transmission) روی صفحه مشاهده می‌کنیم و پرامپت ترمینال به ما نشان داده می‌شود.

برای خواندن ایمیل، کافی است دستور `mail` را وارد کرده دکمه‌ی Enter را بزنیم. به محض انجام این کار، لیستی از ایمیل‌های دریافتی (در صورت وجود) را مشاهده می‌کنیم و یک پرامپت که با علامت & مشخص

می‌شود در اختیار ما قرار می‌گیرد. در این پرامپت، کافی است شماره‌ی ایمیلی که می‌خواهیم مشاهده کنیم را وارد کنیم:

```
[puppy@localhost ~]$ whoami
puppy
[puppy@localhost ~]$ mail
Heirloom Mail version 12.5 7/5/10.  Type ? for help.
"/var/mail/puppy": 1 message
>   1 root                  Thu Mar 25 10:12  31/1006  "Some bad news"
&
& 1
Message  1:
[...]
From: root@localhost.localdomain (root)
Status: RO

I've got some bad news for you. This Linux shit you've
been writing for the past year? No one will ever read this.
No one. Including the one who suggested doing it in the first place.

This, like many other projects before it, will join the shitbin of
history and no one will ever care or remember that these documents ever
existed.

It's sad, but it's reality.

Yours truly,
Reality

& quit
Held 1 message in /var/mail/puppy
```

همانطور که می‌بینید، کاربر puppy توانست با استفاده از دستور mail، ایمیل‌های دریافتی خود را مشاهده کند. ما می‌توانیم با استفاده از برنامه‌ی mail، اقدام به پاک کردن ایمیل‌های دریافتی نیز کنیم. برای این کار کافی است دستور delete *n* را در پرامپت mail وارد کنیم، به طوری که *n* نشان دهنده‌ی شماره‌ی ایمیلی است که قصد پاک کردن آن را داریم. برای خارج شدن از برنامه‌ی mail، کافی است دستور quit را در پرامپت وارد کنیم.

نکته: اگر هیچ ایمیلی به ما ارسال نشده باشد یا به عبارت دیگر، هیچ ایمیلی در صندوق پستی ما موجود نباشد، هنگام اجرای دستور mail، با پیام No email for username مواجه می‌شویم.

اگر ایمیل‌ها در سیستم ما در موقعیتی به جز موقعیت پیش‌فرض /var/spool/mail قرار گرفته باشند، می‌توانیم با استفاده از آپشن -f *DirectoryPath/FileName*، موقعیت قرارگیری ایمیل‌های خود را به mail بگوییم.

اگر مجوز خواندن ایمیل سایر کاربران را داشته باشیم، می‌توانیم با استفاده از آپشن -u *username* به mail بگوییم که ایمیل‌های کاربر دارای یوزرنیم *username* را به ما نشان دهد. اگر ایمیل‌های کاربر مورد نظر در موقعیت غیرپیش‌فرض باشد، می‌توانیم موقعیت قرارگیری ایمیل‌های آن کاربر را مثل قبل با آپشن -f به mail بدهیم. بار دیگر می‌گوییم که برای این کار، باید مجوزهای لازم را داشته باشیم.

بررسی صف ایمیل‌ها

برخی از اوقات، مشکلی پیش می‌آید و ایمیل نوشته شده توسط ما نمی‌تواند به مقصد ارسال شود. ما می‌توانیم با استفاده از دستور mailq، صف ایمیل‌های محلی را مشاهده کرده و از وضعیت ایمیل ارسالی خود مطلع شویم.

برای شبیه‌سازی یک ایمیل گیر کرده در صف، بیایید یک ایمیل به یک آدرس ایمیل غیرواقعی ارسال کنیم. برای این کار، ما یک ایمیل به آدرس crap@asd.com ارسال می‌کنیم. این ایمیل، وجود ندارد و باعث می‌شود که ایمیل نوشته شده توسط ما در صف ایمیل‌ها گیر کند:

```
[root@localhost ~]$ mail -s "Fake mail" crap@asd.com
Faking some email.
EOT
[root@localhost ~]$ mailq
-Queue ID- --Size-- ----Arrival Time---- -Sender/Recipient-------
YBFEB15C0C3*     461 Thu Mar 25 10:57:37  root@localhost.localdomain
                                          crap@asd.com
```

همانطور که می‌بینید، ایمیل ما در صف گیر کرده است و هنوز از سیستم ما خارج نشده است. اگر چیزی داخل صف ایمیل وجود نداشته باشد، اجرای دستور mailq جواب زیر را به ما می‌دهد:

```
[root@localhost ~]$ mailq
Mail queue is empty
```

نکته: موقعیت قرارگیری صف ایمیل، بستگی به نرم‌افزار MTA استفاده شده توسط ما دارد. صف ایمیل معمولا در موقعیتی در /var/spool قرار دارد. برای پیدا کردن موقعیت دقیق دایرکتوری صف ایمیل، می‌توانیم از دستور find /var/spool -name *QueueID* استفاده کنیم، به طوری که *QueueID*، شماره‌ی شناسایی یک ایمیل (که در ستون Queue ID خروجیِ دستور mailq وجود دارد) می‌باشد.

ریدایرکت کردن ایمیل‌ها

Aliasهای ایمیل به ما امکان می‌دهند که بتوانیم ایمیل‌های ارسال شده به یک گیرنده را به یک گیرنده‌ی دیگر تحویل دهیم. برای مثال، با استفاده از Aliasها، ما می‌توانیم کلیه‌ی ایمیل‌هایی که به کاربر admin ارسال می‌شوند را به یک کاربر دیگر در سیستم تحویل دهیم. این امر هم از نظر امنیت و هم از نظر حریم شخصی بسیار کاربردی می‌باشد. در این حالت، ما حتی نیاز نداریم که کاربری به نام admin در سیستم داشته باشیم.

برای تعریف Alias برای یوزرنیم‌های موجود در سیستم، باید دو کار انجام دهیم:

۱- نام مستعار مورد نظر برای هر کاربر را در فایل /etc/aliases اضافه کنیم.

۲- دستور newaliases را اجرا کرده تا دیتابیس Aliasها (/etc/aliases.db) آپدیت شود.

نکته: فایل /etc/aliases.db یک فایل باینری می‌باشد و ما نمی‌توانیم این فایل را به صورت مستقیم تغییر دهیم. به همین دلیل، باید فایل /etc/aliases را تغییر داده و سپس با استفاده از دستور newalises، فایل باینری Aliasها را آپدیت کنیم.

Aliasهایی که در فایل /etc/aliases تعریف می‌کنیم، باید از فرمت زیر پیروی کنند:

```
ALIAS-NAME: RECPIENT1[,RECPIENT2[,...]]
```

یعنی ما ابتدا نام مستعار یا Alias مورد نظر را مشخص کرده و در مقابل آن، کلیه‌ی یوزرنیم‌هایی که می‌خواهیم ایمیل‌های ارسال شده به این Alias را دریافت کنند را مشخص می‌کنیم. بیایید یک Alias به نام admin تعریف کنیم و کاری کنیم که کاربر puppy کلیه‌ی ایمیل‌های ارسال شده به admin را دریافت کند:

```
[root@localhost ~]# vim /etc/aliases
#
#  Aliases in this file will NOT be expanded in the header from
#  Mail, but WILL be visible over networks or from /bin/mail.
marketing:  postmaster
sales:      postmaster
support:    postmaster
[...]
admin: puppy
[root@localhost ~]# newaliases
[root@localhost ~]# mail -s "Just testing aliases bruh" admin
Is it working? No one cares man.
EOT
[puppy@localhost ~]$ whoami
puppy
[puppy@localhost ~]$ mail
Heirloom Mail version 12.5 7/5/10.  Type ? for help.
"/var/mail/puppy": 1 message 1 unread
>U  1 root                  Fri Mar 26 11:31  19/665   "Just testing aliases bruh"
& 1
[...]
From: root@localhost.localdomain (root)
Status: RO

Is it working? No one cares man.
```

همانطور که می‌بینید، ما ابتدا فایل /etc/aliases را باز کرده و یک Alias با عنوان admin درون آن وارد کردیم و این Alias را به کاربر puppy اختصاص دادیم. سپس دستور newaliases را اجرا کردیم تا این Alias جدید در سیستم ثبت شود. سپس یک ایمیل تستی برای کاربر admin فرستادیم. همانطور که می‌بینید، سیستم ایمیل ارسالی به admin را به کاربر puppy تحویل داد، پس Alias ما به درستی کار خود را انجام داده است. تست کردن Aliasها پس از ایجاد آنها بسیار کاربردی می‌باشد، چرا که خیلی از کاربران فراموش می‌کنند که دستور newaliases را اجرا کنند و در نتیجه، Alias ایجاد شده توسط آنها در سیستم ثبت نمی‌شود.

Forward کردن ایمیل‌ها

بعضا پیش می‌آید که ما بنا به دلایلی نظیر مسافرت و... دسترسی به سیستم خود نداشته باشیم و بخواهیم برای مدتی، ایمیل‌هایی که به یوزرنیم ما ارسال می‌شوند، به یک یوزرنیم دیگر در سیستم تحویل داده شوند. در این حالت، Aliasها به درد ما نمیخورند چرا که می‌خواهیم ایمیل‌های تحویل داده شده به خودمان را به یک کاربر دیگر در سیستم، Forward کنیم.

عمل Forward کردن ایمیل، توسط کاربران انجام می‌پذیرد و معمولا دو مرحله دارد:

۱- کاربر یک فایل به نام .forward در دایرکتوری Home خود ایجاد کرده و داخل آن، یوزرنیم فردی که باید ایمیل‌ها به او Forward شوند را قرار می‌دهد.

۲- کاربر با استفاده از دستور chmod، مجوزهای این فایل را به 644 (در فرمت اوکتال) تغییر می‌دهد.

بیایید با هم یک فایل `.forward` ایجاد کرده و عملکرد آن را تست کنیم:

```
[behnam@localhost ~]$ whoami
behnam
[behnam@localhost ~]$ pwd
/home/behnam
[behnam@localhost ~]$ echo puppy > .forward
[behnam@localhost ~]$ chmod 644 .forward
[behnam@localhost ~]$ mail -s "Testing this forward thing" behnam
According to all laws of aviation blablba.
EOT
[behnam@localhost ~]$ mail
No mail for Behnam
[behnam@localhost ~]$ su - puppy
Password:
[...]
[puppy@localhost ~]$ whoami
puppy
[puppy@localhost ~]$ mail
[...]
>U  1 root                  Fri Mar 26 11:31  19/665   "Just testing aliases bruh"
 U  2 behnam@localhost.loc  Fri Mar 26 12:02  22/828   "Testing this forward thing"
& 2
[..]
From: behnam@localhost.localdomain
Status: RO

According to all laws of aviation blablba.

& q
Held 2 messages in /var/spool/mail/puppy
```

همانطور که می‌بینید، پس از ایجاد فایل `.forward` و اعمال مجوزهای صحیح روی آن، ایمیلی که به یوزر behnam ارسال شد، به یوزر puppy تحویل داده شد.

برای غیرفعال کردن Forwarding، کاربر باید فایل `.forwarding` را از دایرکتوری Home خود پاک کند.

مدیریت داده‌ها با استفاده از SQL

زبان پرسمان ساخت‌یافته (Structured Query Language) یا SQL، زبانی برای بازیابی اطلاعات از یک دیتابیس می‌باشد. این زبان در دیتابیس متفاوتی پیاده‌سازی شده است، پس مهم است که ابتدا اطلاعاتی در مورد پکیج‌های دیتابیس موجود در لینوکس به دست آوریم.

انتخاب یک برنامه‌ی دیتابیس

همانطورکه گفتیم، SQL یک زبان برای دسترسی به اطلاعات می‌باشد و برنامه‌های دیتابیس SQL، این زبان را پیاده‌سازی می‌کنند. تفاوت بین SQL و یک برنامه‌ی دیتابیس SQL شبیه تفاوت بین پروتکلی نظیر SMTP و نرم‌افزارهایی که آن پروتکل را پیاده‌سازی می‌کنند، مثل Postfix، می‌باشد. از نظر تئوری، ما می‌توانیم از هر برنامه‌ی دیتابیس SQL برای رفع نیازمندی‌های خود استفاده کنیم، اما در عمل، با توجه به نوع پروژه و سایر نرم‌افزارهای موجود در پروژه، ممکن است مجبور به استفاده از یک برنامه‌ی دیتابیس SQL خاص باشیم.

لینوکس تعدادی زیادی برنامه‌ی دیتابیس SQL دارد، اما معروف‌ترین برنامه‌های دیتابیس SQL در لینوکس به شرح زیر می‌باشند:

- **MySQL**

این دیتابیس، تحت لیسانس GPL منتشر شده است و امروزه صاحب آن، شرکت اوراکل (Oracle) می‌باشد. این دیتابیس در ریپازیتوری‌های برخی از توزیع‌های لینوکسی موجود می‌باشد. برای نصب کامل این دیتابیس، ما باید کلاینت، سرور و هم‌چنین ابزارهای توسعه‌ی این دیتابیس را روی سیستم نصب کنیم.

- **MariaDB**

پس از خریداری MySQL توسط شرکت اوراکل، توسعه‌دهندگان MariaDB، دیتابیس MySQL را فورک کرده (سورس کد MySQL را برداشتند) و اقدام به توسعه‌ی این دیتابیس به عنوان یک جایگزین کامل برای MySQL کردند. کلیه مشخصات و دستورات موجود در MariaDB دقیقا مانند دستورات MySQL می‌باشند، با این تفاوت که MariaDB برخی ویژگی‌های اضافه‌تر نیز دارد.

در ریپازیتوری‌های CentOS، دیتابیس MariaDB جایگزین MySQL شده است، به طوری که حتی جستجو برای نصب MySQL نیز، MariaDB را نصب می‌کند.

- **PostgreSQL**

این دیتابیس تحت لیسانس BSD توسعه داده شده است و تحت چندین پکیج، در اکثر توزیع‌های لینوکسی قابل نصب می‌باشند. برای نصب کامل این دیتابیس نیز، باید کلاینت، سرور و چندین پکیج پشتیبان نصب کنیم.

- **SQLite**

این پکیج، یک لایبرری می‌باشد که SQL را پیاده‌سازی می‌کند. در واقع SQLite به تنهایی یک دیتابیس نیست، بلکه ابزاری است که به برنامه‌ها امکان می‌دهد تا اطلاعات را با استفاده از اینترفیس SQL درون خود ذخیره کنند. به عبارت دیگر، اگر بخواهیم برنامه‌ای بنویسیم که با استفاده از SQL اطلاعات را ذخیره می‌کند اما از یک دیتابیس کامل مثل MySQL و... استفاده نمی‌کند، می‌توانیم از SQLite استفاده کنیم.

همانطور که اشاره کردیم، برخی از برنامه‌های SQL نظیر MySQL، MariaDB و PostgreSQL از مدل کلاینت-سرور استفاده می‌کنند؛ یعنی یک برنامه (سرور) دیتابیس را مدیریت می‌کند و یک برنامه‌ی دیگر (کلاینت) امکان دسترسی به دیتابیس را به کاربران و برنامه‌ها می‌دهد. این نوع معماری در شبکه بسیار کاربردی می‌باشد، چرا که به کاربران امکان می‌دهد که از طریق شبکه به یک دیتابیس مرکزی دسترسی داشته باشند.

درک مقدمات SQL

گفتیم که SQL یک ابزار برای دسترسی به دیتابیس می‌باشد. اما به صورت دقیق‌تر، SQL یک ابزار برای دسترسی به دیتابیس‌های رابطه‌ای یا Relational می‌باشد. در این نوع دیتابیس‌ها، هر ردیف، نشان دهنده‌ی یک آیتم یا یک آبجکت می‌باشد و هر ستون (یا فیلد)، نشان دهنده‌ی یک ویژگی خاص می‌باشد. به ترکیب ردیف‌ها و ستون‌ها، یک جدول می‌گویند. نمونه‌ای از این جدول را در تصویر ۲ مشاهده می‌کنید:

ستون یا ویژگی

ردیف

lizard	green	5 inches	soft	$10
tree	green	10 feet	medium	$200
pillow	white	18 inches	soft	$5
brick	red	8 inches	hard	$1
banana	yellow	8 inches	soft	$0.10

تصویر ۲- نمونه‌ای از یک جدول دیتابیس

هر دیتابیس، می‌تواند چندین جدول داشته باشد و به علاوه، در SQL می‌توانیم چندین دیتابیس داشته باشیم. یعنی مثلا در یک سازمان بزرگ، می‌توانیم برای هر شعبه یک دیتابیس مجزا داشته باشیم و در هر دیتابیس، یک جدول برای ذخیره‌ی اطلاعات کارمندان و یک جدول برای ذخیره اطلاعات مربوط به دارایی‌های هر شعبه داشته باشیم. پس برای دسترسی به اطلاعات، باید ابتدا یک دیتابیس انتخاب و سپس یک جدول درون آن دیتابیس را انتخاب کنیم.

به صورت پیش‌فرض، جدول‌ها در SQL بدون ترتیب می‌باشند. ما می‌توانیم هنگام اجرای کوئری‌ها، کاری کنیم که نتایج با ترتیب خاصی نشان داده شوند؛ یعنی مثلا می‌توانیم کاری کنیم که ردیف‌های نشان داده شده در خروجی کوئری SQL، بر حسب اعداد موجود در یک ستون (مثلا ستون آخر در تصویر ۲) مرتب شوند.

با استفاده از SQL، می‌توانیم اطلاعاتی که از معیار خاصی پیروی می‌کنند را بازیابی کنیم. مثلا در جدولی مثل جدول نشان داده شده در تصویر ۲، ما می‌توانیم کلیه‌ی آبجکت‌های سبز را در خروجی بازیابی کنیم. علاوه بر این، ما می‌توانیم اطلاعاتی را به جدول اضافه، پاک یا حتی محتویات یک ردیف جدول را آپدیت کنیم.

هر ستون یا ویژگی در دیتابیس، یک نوع خاص از داده را درون خود ذخیره می‌کند. در تصویر ۲، واضح است که ستون دوم نشان دهنده‌ی رنگ و ستون آخر نشان دهنده‌ی قیمت می‌باشد. وارد کردن قیمت در ستون دوم یا وارد کردن رنگ در ستون آخر، جدول ما را به هم می‌ریزد. به همین دلیل، روی هر ستون، محدودیتی به نام Data Type یا Domain وجود دارد. اگر با برنامه‌نویسی آشنا باشید، مفهوم Data Type برایتان آشنا خواهد بود. در جدول ما، Data Type یا دامنه‌ی ستون دوم، یک سری کاراکتر که مشخص کننده‌ی رنگ هستند می‌باشد و دامنه‌ی ستون آخر، یک مقدار عددی می‌باشد. جدول ۱ برخی از Data Typeهای موجود در SQL را نشان می‌دهد:

جدول ۱- انواع Data Type موجود در SQL

Data Type	نشان دهنده‌ی
INTEGER (یا INT)	عدد صحیح ٤ بایتی
SMALLINT	عدد صحیح ۲ بایتی
DECIMAL	مشخص کننده‌ی تعداد ممیز برای اعداد اعشاری
NUMERIC	مشخص کننده‌ی تعداد ممیز برای اعداد اعشاری
FLOAT	عدد اشاری
DOUBLE PERCISION	عدد اعشاری با دو برابر تعداد ممیز نسبت به FLOAT
DATETIME	تاریخ و زمان

DATE	تاریخ
TIME	زمان در فرمت HH:MM:SS
CHAR	یک یا چند کاراکتر
VARCHAR	تعداد غیر مشخصی کاراکتر
ENUM	لیستی از مقادیر رشته‌ای که هر ردیف در جدول، فقط می‌تواند یکی از آن مقادیر را داشته باشد.
SET	لیستی از مقادیر رشته‌ای که هر ردیف جدول، می‌تواند صفر، یک یا بیشتر از آن مقادیر را داشته باشد.

نصب MariaDB

در بسیاری از توزیع‌ها، هیچ دیتابیسی به صورت پیش‌فرض روی سیستم نصب نیست و ما باید خودمان در صورت نیاز یک دیتابیس روی آن نصب کنیم. ما دیتابیس MariaDB را نصب می‌کنیم. ما عمل نصب را با استفاده از پکیج منیجر yum انجام می‌دهیم:

```
[root@localhost ~]# yum install mariadb mariadb-server
[...]
Installed:
  mariadb.x86_64 1:5.5.68-1.el7      mariadb-server.x86_64 1:5.5.68-1.el7

Complete!
```

پس از نصب MariaDB، باید آن را استارت بزنیم و همچنین در صورت نیاز، آن را enable کنیم تا پس از ریبوت سیستم، به صورت اتوماتیک استارت شود:

```
[root@localhost ~]# systemctl start mariadb
[root@localhost ~]# systemctl enable mariadb
Created symlink from /etc/systemd/system/multi-user.target.wants/mariadb.service
to /usr/lib/systemd/system/mariadb.service.
[root@localhost ~]# systemctl status mariadb
[...]
   Active: active (running) since Sun 2021-03-28 12:04:09 +0430; 6s ago
```

حال بهتر است اسکریپت امنیتی MariaDB را اجرا کنیم تا تنظیمات پیش‌فرض غیرامن این دیتابیس را تغییر دهیم. نکته‌ی جالب این است که برای کار کردن با دیتابیس MariaDB، از دستورهای مربوط به و دارای پیشوند mysql استفاده می‌کنیم، که البته این امر به این دلیل می‌باشد که MariaDB فورک MySQL می‌باشد و به عنوان جایگزینی برای آن عمل می‌کند. به هر حال، برای اجرای اسکریپت امنیتی، به صورت زیر عمل می‌کنیم:

```
[root@localhost ~]# mysql_secure_installation
[...]

In order to log into MariaDB to secure it, we'll need the current
password for the root user.  If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.
Enter current password for root (enter for none):
```

در اینجا اسکریپت به ما می‌گوید که در صورتی که برای کاربر روت دیتابیس (توجه کنید، کاربر روت دیتابیس، نه سیستم) رمزی انتخاب کرده‌ایم، آن را وارد کنیم. از آنجایی که ما همین الان دیتابیس را نصب کرده‌ایم، کاربر روت هیچ رمزی ندارد، پس ما فقط دکمه‌ی Enter را فشار می‌دهیم.

```
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.
Set root password? [Y/n] y
New password:
Re-enter new password:
[...]
```

در اینجا اسکریپت از ما میپرسد که آیا تمایلی به ایجاد رمز برای کاربر روت دیتابیس داریم یا نه. بهتر است یک رمز برای کاربر روت دیتابیس مشخص کنیم. پس در پرامپت اول حرف y را نوشته و در پرامپت بعدی، رمز مورد نظر را انتخاب کرده و در پرامت بعد از آن، رمز انتخابی را تکرار می‌کنیم.

از این مرحله به بعد، به کلیه‌ی سوالات با حرف y پاسخ می‌دهیم. اگر در مورد دلیل آن کنجکاو هستید، کافی است مواردی که توسط این اسکریپت در صفحه نشان داده می‌شود را بخوانید، چرا که توضیح دقیق‌تر این مسائل از حوصله‌ی ما خارج است.

```
[...]

Remove anonymous users? [Y/n] y
 ... Success!

[...]

Disallow root login remotely? [Y/n] y
 ... Success!

[...]

Remove test database and access to it? [Y/n] y
 - Dropping test database...
[...]

Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

حال بهتر است یک یوزر جدید درون دیتابیس تعریف کنیم تا مجبور نباشیم هر سری با یوزر روت وارد دیتابیس شویم. برای این کار، مجبوریم یک بار با یوزر روت وارد دیتابیس شده، سپس یک یوزر جدید تعریف کرده و از آن به بعد، از این یوزر جدید برای مدیریت دیتابیس استفاده کنیم. برای وارد شدن به دیتابیس با یوزر روت به صورت زیر عمل می‌کنیم:

```
[root@localhost ~]# mysql -u root -p
```

همانطور که می‌بینید، ما با ارائه‌ی آپشن u- به دستور mysql و وارد کردن نام root، به دیتابیس می‌گوییم که با یوزر root اقدام به لاگین کند و با استفاده از آپشن p-، به دیتابیس می‌گوییم که این یوزر، یک رمز دارد و باید رمز آن را از ما دریافت کند. به محض زدن دکمه‌ی Enter، دیتابیس رمز کاربر روت را درخواست می‌کند. پس از وارد کردن رمز صحیح، با پرامپت زیر مواجه می‌شویم:

```
MariaDB [(none)]>
```

برای ایجاد یک کاربر جدید با یوزنیم admin و رمز letmein، دستور زیر را وارد می‌کنیم:

```
MariaDB [(none)]> CREATE USER 'admin'@localhost IDENTIFIED BY 'letmein';
Query OK, 0 rows affected (0.10 sec)
```

نکته‌ای که باید به آن توجه کنید این است که در انتهای کلیه‌ی دستورهای SQL، باید علامت نقطه‌ویرگول (;) قرار دهیم. علاوه بر این، در طول این جزوه و سایر منابع مربوط به SQL، کلیه دستورات SQL با حروف بزرگ نوشته می‌شوند؛ با این حال برای SQL مهم نیست که دستورات وارد شده توسط ما با حروف بزرگ یا کوچک نوشته شده باشند.

حال می‌خواهیم کاری کنیم که یوزر admin، دسترسی کامل به کلیه‌ی دیتابیس‌ها داشته باشد و بتواند آنها را مدیریت کند. برای این کار، باید هم یوزرنیم و هم پسوورد یوزر admin را در دستور زیر قرار دهیم:

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* TO 'admin'@localhost IDENTIFIED BY
'letmein';
Query OK, 0 rows affected (0.01 sec)
```

پس از این، باید کاری کنیم که MariaDB این تنظیمات جدید را اعمال کند. برای این کار، دستور زیر را وارد می‌کنیم:

```
MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

حال باید از MariaDB خارج شویم و با یوزر admin که ساختیم وارد آن شویم. برای خروج از MariaDB عمل می‌کنیم:

```
MariaDB [(none)]> exit
Bye
```

برای ورود به MariaDB با یوزر admin به صورت زیر عمل می‌کنیم:

```
[root@localhost ~]# mysql -u admin -p
Enter password:
[...]
MariaDB [(none)]>
```

ایجاد دیتابیس و جدول

برای ایجاد یک دیتابیس، از دستور زیر استفاده می‌کنیم. دقت کنید که بر خلاف دستورات SQL، اسم دیتابیس‌ها به حروف بزرگ و کوچک حساس می‌باشد، یعنی دیتابیس test، با دیتابیس Test متفاوت خواهد بود. بیایید یک دیتابیس با نام linux ایجاد کنیم. برای این کار:

```
MariaDB [(none)]> CREATE DATABASE linux;
Query OK, 1 row affected (0.00 sec)
```

حال بیایید صحت ایجاد این دیتابیس را بررسی کنیم. با استفاده از دستور زیر، می‌توانیم کلیه‌ی دیتابیس‌های تعریف شده در MariaDB را مشاهده کنیم:

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| linux              |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.00 sec)
```

همانطور که می‌بینید، دیتابیسی که ایجاد کردیم در لیست دیتابیس‌های موجود در سیستم قرار دارد. برای این که بتوانیم از یک دیتابیس ایجاد شده استفاده کنیم و در آن جدولی ایجاد یا جدولی را مشاهده کنیم، باید ابتدا آن دیتابیس را انتخاب کنیم. بیایید دیتابیس linux را انتخاب کنیم:

```
MariaDB [(none)]> USE linux;
Database changed
MariaDB [linux]>
```

داخل هر دیتابیس، ما می‌توانیم چندین جدول ایجاد کنیم. در یک دیتابیسی که به تازگی ایجا شده، نظیر دیتابیسی که ما الان ایجاد کردیم، هیچ جدولی وجود ندارد. ما می‌توانیم صحت این امر را با اجرای دستور زیر بررسی کنیم:

```
MariaDB [linux]> SHOW TABLES;
Empty set (0.00 sec)
```

همانطور که می‌بینید، در خروجی مقدار Empty set را دریاقت می‌کنیم. این به معنای خالی بودن این دیتابیس (دیتابیس linux) می‌باشد. برای این که بتوانیم در این دیتابیس اطلاعاتی را وارد کنیم، باید ابتدا بدانیم که چه اطلاعاتی را می‌خواهیم در این دیتابیس ذخیره کنیم، یا به عبارت دیگر، باید بدانیم که چه جدول‌هایی را می‌خواهیم در این دیتابیس داشته باشیم. برای مثال، در تصویر ۲، ما یک جدول داشتیم که ویژگی‌های یک اشیای معمولی، اعم از نام، رنگ، اندازه، میزان سفتی و قیمت آنها را درون خود نگهداری می‌کرد.

بیایید جدول موجود در تصویر ۲ را در دیتابیس linux ایجاد کنیم. برای این کار:

```
MariaDB [linux]> CREATE TABLE objects (name VARCHAR(30), color VARCHAR(20),
    -> size FLOAT, hardness ENUM('soft', 'medium', 'hard'), value DECIMAL(10,2));
Query OK, 0 rows affected (0.48 sec)
```

همانطور که می‌بینید، ما این جدول را objects نامیدیم و ۵ ستون با نام‌های name، color، size، hardness و value برای آن ایجاد کردیم. هر ستون، یک Data Type دارد. ما دیتاتایپ‌ها را در جدول ۱ معرفی کردیم و آنها را توضیح دادیم. بیایید با دلیل انتخاب این Data Typeها برای هر ستون آشنا شویم:

- ستون‌های name و color، هر دو دارای دیتاتایپ VARCHAR هستند، اما سایز متفاوتی دارند. اگر دقت کنید، ما دقیقا پس از VARCHAR، درون یک پرانتز، سایز هر ستون را مشخص کرده‌ایم. مواردی که در ستون name وارد می‌شوند می‌توانند حداکثر ۳۰ کاراکتر داشته باشند و مواردی که در ستون color وارد می‌شوند، می‌توانند حداکثر ۲۰ کاراکتر داشته باشند. اگر این دو ستون را به عنوان CHAR تعریف کرده بودیم، کلیه‌ی موارد وارد شده در ستون name باید حتما ۳۰ کاراکتر و موارد ستون color، باید حتما ۲۰ کاراکتر می‌داشتند.
- ستون size دارای دیتاتایپ FLOAT می‌باشد. این دیتاتایپ می‌تواند اعداد حقیقی (مثل اعداد اعشاری و...) را درون خود نگهداری کند. توجه داشته باشید که در تصویر ۲، موارد موجود در این ستون در دو واحد اینچ و فوت می‌باشند، در دنیای واقعی کلیه‌ی اعداد ذخیره شده در یک ستون باید دارای یک واحد باشند. مثلا در این مثال، بهتر است واحد سایز، اینچ باشد.
- ستون hardness دارای دیتاتایپ ENUM می‌باشد. این یعنی که موارد وارد شده در این ستون، فقط می‌توانند یکی از رشته‌های مشخص شده در پرانتز (یعنی یا soft، یا medium یا hard) باشند. لازم است که به چگونگی مشخص کردن مقادیر مجاز در ENUM دقت کنید. همانطور که می‌بینید، مقادیر

مجاز داخل پرانتز قرار گرفته و هر مقدار بین دو علامت ' قرار می‌گیرد و هر مقدار، با علامت کاما از مقدار بعدی جدا می‌شود.

- ستون value دارای دیتا تایپ DECIMAL می‌باشد. هنگام مشخص کردن این دیتا تایپ، باید تعداد ارقام و همچنین تعداد اعشاری که پس از ارقام می‌آیند را مشخص کنیم. در اینجا، ما دیتاتایپ value را به صورت DECIMAL(10, 2) تعریف کرده‌ایم. این یعنی که اعداد وارد شده در این ستون می‌توانند ۱۰ رقم و همچنین ۲ رقم اعشار داشته باشند.

حال بیایید از صحت ایجاد جدول objects اطمینان حاصل کنیم. برای این کار، از دستور DESCRIBE به همراه نام جدول استفاده می‌کنیم. یعنی:

```
MariaDB [linux]> DESCRIBE objects;
+----------+------------------------------+------+-----+---------+-------+
| Field    | Type                         | Null | Key | Default | Extra |
+----------+------------------------------+------+-----+---------+-------+
| name     | varchar(30)                  | YES  |     | NULL    |       |
| color    | varchar(20)                  | YES  |     | NULL    |       |
| size     | float                        | YES  |     | NULL    |       |
| hardness | enum('soft','medium','hard') | YES  |     | NULL    |       |
| value    | decimal(10,2)                | YES  |     | NULL    |       |
+----------+------------------------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

وارد کردن اطلاعات درون جدول

حال که جدول مورد نظر خود را در دیتابیس ایجاد کردیم، نوبت آن رسیده که اطلاعاتی را درون آن ذخیره کنیم. برای این کار، از دستور INSERT INTO استفاده می‌کنیم:

```
MariaDB [linux]> INSERT INTO objects
    -> VALUES ('lizard', 'green', 6, 'soft', 10.00);
Query OK, 1 row affected (0.01 sec)
```

ما با استفاده از این دستور، اولین ردیف موجود در تصویر ۲ را داخل جدول objects وارد کردیم. بیایید از وارد شدن این ردیف در جدول objects اطمینان حاصل کنیم:

```
MariaDB [linux]> SELECT * FROM objects;
+--------+-------+------+----------+-------+
| name   | color | size | hardness | value |
+--------+-------+------+----------+-------+
| lizard | green |    6 | soft     | 10.00 |
+--------+-------+------+----------+-------+
1 row in set (0.00 sec)
```

همانطور که می‌بینید، استفاده از دستور بالا، کلیه‌ی محتویات موجود در جدول objects را در خروجی به ما نشان داد (در بخش بعد این دستور را بیشتر توضیح می‌دهیم). اگر به جدول موجود در تصویر ۲ نگاه کنید، می‌بینید که ما در وارد کردن اطلاعات موجود در اولین ردیف، دچار اشتباه شده‌ایم و مقدار size این ردیف را به جای ۵، برابر با ۶ قرار داده‌ایم. خوشبختانه ما می‌توانیم با استفاده از دستور UPDATE، این مشکل را حل کنیم:

```
MariaDB [linux]> UPDATE objects SET size=5 WHERE name='lizard';
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

همانطور که می‌بینید، ما پس از مشخص کردن دستور UPDATE، نام جدولی که می‌خواهیم آپدیت کنیم، یعنی objects را مشخص کردیم. سپس با استفاده از SET، گفتیم که مقدار چه ستونی، برابر با چه مقدار قرار گیرد. سپس با استفاده از WHERE، گفتیم که این تغییر در کجا باید صورت پذیرد.

پس به عبارت دیگر، ما با استفاده از این دستور به MariaDB گفتیم که مقدار ستون size در جدول objects را در هر کجا که ستون name برابر با lizard هست را برابر با ۵ قرار دهد. بیایید از صحت رفع مشکل این ردیف مطمئن شویم:

```
MariaDB [linux]> SELECT * FROM objects;
+--------+-------+------+----------+-------+
| name   | color | size | hardness | value |
+--------+-------+------+----------+-------+
| lizard | green |    5 | soft     | 10.00 |
+--------+-------+------+----------+-------+
1 row in set (0.00 sec)
```

تمرین: با توجه به مواردی که تا اینجاد یاد گرفتید، سایر موارد موجود در جدول تصویر ۲ را داخل دیتابیس وارد کنید. دقت کنید که کلیه‌ی موارد موجود در ستون size، باید در واحد اینچ باشند.

حال که تمرین را انجام دادید، بیایید محتویات جدول را چک کنیم:

```
MariaDB [linux]> SELECT * FROM objects;
+--------+--------+------+----------+--------+
| name   | color  | size | hardness | value  |
+--------+--------+------+----------+--------+
| lizard | green  |    5 | soft     |  10.00 |
| tree   | green  |  120 | medium   | 200.00 |
| pillow | white  |   18 | soft     |   5.00 |
| brick  | red    |    8 | hard     |   1.00 |
| banana | yellow |    8 | soft     |   0.10 |
+--------+--------+------+----------+--------+
5 rows in set (0.00 sec)
```

استخراج اطلاعات از دیتابیس

هدف اصلی از ذخیره‌ی اطلاعات، استخراج آن اطلاعات می‌باشد. دستور اصلی برای استخراج اطلاعات از دیتابیس، دستور SELECT که قبلا با آن آشنا شدیم می‌باشد. قدرت این دستور در این می‌باشد می‌تواند به صورت جزئی، دقیقا چیزی که ما می‌خواهیم را انتخاب کند. ما می‌توانیم با استفاده از برخی از کلیدواژه‌ها، فقط اطلاعاتی را استخراج کنیم که دارای ویژگی یا ویژگی‌های خاصی می‌باشند. فرمت کلی دستور SELECT به صورت زیر می‌باشد:

```
SELECT field(s) FROM table [ WHERE conditions ] [ ORDER BY field ]
```

ما تا به اینجا در قسمت *field(s)* از علامت * استفاده می‌کردیم. علامت * به این معنی است که این دستور کلیه‌ی ستون‌هایی که از معیارهای مشخص شده پیروزی می‌کنند را در خروجی به ما نشان می‌دهد. ما می‌توانیم به جای *، از اسم ستون‌هایی که می‌خواهیم در خروجی به ما نشان داده شوند استفاده کنیم. مثلا فرض کنید ما فقط می‌خواهیم ستون‌های color و value از جدول objects به ما نشان داده شود. برای این کار به صورت زیر عمل می‌کنیم:

```
MariaDB [linux]> SELECT value, color FROM objects;
+--------+--------+
| value  | color  |
+--------+--------+
|  10.00 | green  |
| 200.00 | green  |
|   5.00 | white  |
|   1.00 | red    |
|   0.10 | yellow |
+--------+--------+
5 rows in set (0.00 sec)
```

همانطور که می‌بینید، ما در اینجا در قسمت *field(s)* از نام دو ستون استفاده کردیم و MariaDB ستون‌ها را طبق ترتیبی که در *field(s)* مشخص کرده بودیم به ما نشان داد.

ما با استفاده از شرط‌ها، می‌توانیم فقط ردیف‌هایی که از یک شرط خاصی پیروی می‌کنند را در خروجی نشان دهیم. ما شرط‌ها را با استفاده از کلیدواژه‌ی *conditions* WHERE مشخص می‌کنیم. ما می‌توانیم *conditions* را به چند طریق مشخص کنیم:

- تطبیق دقیق

در این حالت، با استفاده از نام یک ستون، علامت = و یک مقدار، می‌توانیم فقط ردیف‌هایی که در ستون مشخص شده، مقدار مشخص شده دارند را استخراج کنیم. برای مثال، وارد کردن دستور زیر:

```
SELECT * FROM objects WHERE color='green';
```

فقط ردیف‌هایی در جدول objects که در ستون color مقدار green دارند را به ما نشان می‌دهد؛ یعنی در مثال ما فقط دو ردیف در خروجی (lizard و tree) به ما نشان داده می‌شود.

- شروط عددی

ما می‌توانیم کاری کنیم که فقط ردیف‌هایی که از یک معیار عددی خاص پیروی می‌کنند در خروجی به ما نشان داده شوند. مثلا اجرای دستور زیر:

```
SELECT * FROM objects WHERE size>10;
```

فقط ردیف‌هایی که در ستون size مقداری بیشتر از ۱۰ اینچ دارند را به ما نشان می‌دهد.

- شروط الفبایی

اپراتورهای بزرگتر-از (>) و کوچکتر-از (<) بر روی حروف الفبا نیز کار می‌کنند. یعنی ما می‌توانیم با توجه به حرف اول موجود در یک رشته، اطلاعاتی را استخراج کنیم. برای مثال، وارد کردن دستور زیر:

```
SELECT * FROM objects WHERE name>'b';
```

ردیف‌هایی در ستونname که با حرف b یا هر حرف بعد از b شروع می‌شوند را به ما نشان می‌دهد. دقت کنید که با این که ما از اپراتور > استفاده کردیم، رشته‌هایی که با حرف b شروع می‌شوند نیز در خروجی به ما نشان داده می‌شود.

- چند شرطی

ما می‌توانیم چندین شرط را با استفاده از AND و OR با هم ترکیب کنیم. برای مثال اگر بخواهیم فقط ردیف‌هایی که در ستون hardness دارای مقدار soft هستند و همچنین بیشتر از ۷٫۵ دلار قیمت دارند را بازیابی کنیم، می‌توانیم از دستور زیر استفاده کنیم:

```
SELECT * FROM objects WHERE hardness='soft' AND value>7.50;
```

ما می‌توانیم از MariaDB بخواهیم که ردیف‌های استخراج شده را با توجه به موارد موجود در یک ستون، مرتب کند. برای این کار از کلیدواژه‌ی ORDER BY به علاوه‌ی نام ستونی که میخواهیم مرتب‌سازی بر حسب آن انجام شود استفاده می‌کنیم. برای مثال اگر بخواهیم کلیه‌ی ردیف‌هایی که دارای مقدار soft در ستون hardness می‌باشند را استخراج کرده و آنها را بر حسب مقادیر موجود در ستون value مرتب کنیم، از دستور زیر استفاده می‌کنیم:

```
MariaDB [linux]> SELECT * FROM objects WHERE hardness='soft' ORDER BY value;
+--------+--------+------+----------+-------+
| name   | color  | size | hardness | value |
+--------+--------+------+----------+-------+
| banana | yellow |    8 | soft     |  0.10 |
| pillow | white  |   18 | soft     |  5.00 |
| lizard | green  |    5 | soft     | 10.00 |
+--------+--------+------+----------+-------+
3 rows in set (0.01 sec)
```

ادغام اطلاعات از چند جدول

همانطور که قبلا گفتیم، یک دیتابیس می‌تواند چندین جدول داشته باشد. این ویژگی SQL به ما امکان می‌دهد که بتوانیم برای کارهای متفاوت، جداول متفاوتی ایجاد کنیم. همانطور که می‌دانید، جدول موجود در تصویر ۲، ویژگی‌های کلی یک سری شی را به ما نشان می‌دهد. فرض کنید در همین دیتابیس، یک جدول دیگر داریم که موقعیت مکانی و همچنین وضعیت هر شی (در طیف ۱ تا ۱۰) را به ما نشان می‌دهد. این جدول به صورت زیر می‌باشد:

جدول ۲- جدول وضعیت اشیا

شناسه	نام	مکان	وضعیت
۱	banana	kitchen	۹
۲	banana	kitchen	۸
۳	tree	backyard	۲
٤	brick	garage	۱۰
۵	brick	garage	۹
۶	brick	backyard	۹
۷	lizard	living room	۸

گاهی اوقات ممکن است نیاز شود که این دو جدول را ادغام کرده و درون یک جدول قرار دهیم تا بتوانیم اطلاعات متفاوتی را از آن استخراج کنیم.

برای این که بتوانیم دو جدول را با هم ادغام کنیم، دو جدول باید یک فیلد (یا ستون) مشترک داشته باشند. SQL از این ستون مشترک، برای ادغام دو جدول با یکدیگر استفاده می‌کند. علاوه بر این، هر جدول باید یک ستون که به صورت منحصر به فرد ویژگی هر ردیف را مشخص می‌کند داشته باشد. ستونی که ویژگی منحصر به فرد هر ردیف را مشخص می‌کند، کلید اصلی یا Primary Key نام دارد. در جدول تصویر ۱، ستون name کلید اصلی می‌باشد و در جدول ۲، ستون شناسه کلید اصلی می‌باشد.

ابتدا بیایید جدول ۲ را در دیتابیس ایجاد کنیم و ردیف‌های متفاوت را درون آن وارد کنیم:

```
MariaDB [linux]> CREATE TABLE locations (id INTEGER, name VARCHAR(30),
    -> location VARCHAR(30), cond INTEGER);
Query OK, 0 rows affected (0.07 sec)
MariaDB [linux]> DESCRIBE locations;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| id       | int(11)     | YES  |     | NULL    |       |
| name     | varchar(30) | YES  |     | NULL    |       |
| location | varchar(30) | YES  |     | NULL    |       |
| cond     | int(11)     | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

همانطور که می‌بینید، ما این جدول را locations نامیدیم، ستون شناسه را id، ستون نام را name، ستون مکان را location و ستون وضعیت را cond نامیدیم. حال باید موارد موجود در هر ردیف جدول ۲ را وارد جدول locations کنیم:

```
MariaDB [linux]> INSERT INTO locations VALUES(1, 'banana', 'kitchen', 9);
Query OK, 1 row affected (0.01 sec)
MariaDB [linux]> INSERT INTO locations VALUES(2, 'banana', 'kitchen', 8);
Query OK, 1 row affected (0.00 sec)
MariaDB [linux]> INSERT INTO locations VALUES(3, 'tree', 'backyard', 2);
Query OK, 1 row affected (0.02 sec)
MariaDB [linux]> INSERT INTO locations VALUES(4, 'brick', 'garage', 10);
Query OK, 1 row affected (0.00 sec)
MariaDB [linux]> INSERT INTO locations VALUES(5, 'brick', 'garage', 9);
Query OK, 1 row affected (0.00 sec)
MariaDB [linux]> INSERT INTO locations VALUES(6, 'brick', 'backyard', 9);
Query OK, 1 row affected (0.01 sec)
MariaDB [linux]> INSERT INTO locations VALUES(7, 'lizard', 'living room', 8);
Query OK, 1 row affected (0.03 sec)
```

حال بیایید وضعیت جدول را بررسی کنیم:

```
MariaDB [linux]> SELECT * FROM locations;
+------+--------+-------------+------+
| id   | name   | location    | cond |
+------+--------+-------------+------+
|    1 | banana | kitchen     |    9 |
|    2 | banana | kitchen     |    8 |
|    3 | tree   | backyard    |    2 |
|    4 | brick  | garage      |   10 |
|    5 | brick  | garage      |    9 |
|    6 | brick  | backyard    |    9 |
|    7 | lizard | living room |    8 |
+------+--------+-------------+------+
7 rows in set (0.00 sec)
```

همانطور که می‌بینید، جدول locations دقیقا مطابق جدول ۲ پر شده است.

ما اکنون می‌توانیم با استفاده از SELECT، اطلاعاتی را از هر دو جدول استخراج کنیم یا به عبارتی، اطلاعات این دو جدول را با هم ادغام کنیم. مثلا فرض کنید می‌خواهیم موقعیت مکانی کلیه‌ی اشیای سبز رنگ را به دست آوریم. جدول اول (جدول objects) رنگ هر شی را درون خود دارد، اما موقعیت مکانی آنها را ندارد و از

طرف دیگر، جدول دوم (جدول locations) موقعیت مکانی اشیا را داشته اما اطلاعات رنگ آنها را ندارد. پس برای استخراج کلیه‌ی اشیا سبز رنگ و موقعیت مکانی آنها، دستور زیر را وارد می‌کنیم:

```
MariaDB [linux]> SELECT objects.name, objects.color, locations.location
    -> FROM objects, locations
    -> WHERE objects.name=locations.name AND objects.color='green';
+--------+-------+-------------+
| name   | color | location    |
+--------+-------+-------------+
| tree   | green | backyard    |
| lizard | green | living room |
+--------+-------+-------------+
2 rows in set (0.01 sec)
```

همانطور که می‌بینید، ما ابتدا دستور SELECT را وارد کردیم و سپس گفتیم که از کدام جدول، کدام ستون‌ها را می‌خواهیم. objects.name و objects.color به ستون name و color در جدول objects اشاره می‌کند و locations.location به ستون location در جدول locations اشاره دارد. پس از مشخص کردن نام ستون‌ها، لازم است با کلیدواژه‌ی FROM، نام جدول‌هایی که می‌خواهیم اطلاعات از آن استخراج شود را مشخص کنیم. پس از این کار، با استفاده از کلیدواژه‌ی WHERE، معیار مورد نظر برای استخراج اطلاعات را مشخص می‌کنیم. در اینجا، معیار ما این است که ستون name در جدول objects برابر با ستون name در جدول locations باشد و همچنین ستون color در جدول objects، دارای مقدار green باشد.

روش دیگر برای ادغام دو جدول، استفاده از JOIN می‌باشد. این روش بسیار شبیه به روش قبل می‌باشد، با این تفاوت که یک جدول را با استفاده از FROM مشخص کرده و جدول (یا جداول) بعدی را با استفاده از JOIN مشخص می‌کنیم. به صورت زیر:

```
MariaDB [linux]> SELECT objects.name, objects.color, locations.location
    -> FROM objects
    -> JOIN locations
    -> WHERE objects.name=locations.name AND objects.color='green';
+--------+-------+-------------+
| name   | color | location    |
+--------+-------+-------------+
| tree   | green | backyard    |
| lizard | green | living room |
+--------+-------+-------------+
2 rows in set (0.00 sec)
```

ادغام اطلاعات به این صورت، باعث می‌شود که دیتابیس ساختار ساده‌تری داشته باشد. مثلا در همین مثال ما، جدول objects ویژگی‌های ظاهری اشیا را به صورت کلی نشان می‌دهد، در حالی که جدول locations موقعیت مکانی هر شی موجود در سیستم را نشان می‌دهد. این نوع طراحی دیتابیس باعث می‌شود که سایز جداول در سیستم، بیش از اندازه بزرگ نشود. در این مثال اگر همه‌ی اطلاعات را در یک جدول ذخیره می‌کردیم، باعث می‌شد که بسیاری از اطلاعات در ردیف‌های متفاوت تکرار شوند. این امر در درازمدت باعث بزرگ شدن بیش از حد جداول و نیاز به فضای بسیار زیاد برای نگهداری جداول می‌شود.

دستور GROUP BY

یکی از دستورات بسیار مهم برای استخراج اطلاعات، GROUP BY نام دارد. این دستور که همیشه همراه با اپراتورهای ریاضی نظیر SUM( ) و... می‌آید، به ما امکان می‌دهد که استخراج اطلاعات را محدود به ستون‌های مشخص شده کنیم. برای مثال، فرض کنید می‌خواهیم مجموع قیمت کلیه‌ی اشیای موجود در سیستم را، با توجه به نام هر شی، بدانیم. برای این کار:

```
MariaDB [linux]> SELECT objects.name, objects.value, SUM(value)
    -> FROM objects, locations
    -> WHERE objects.name=locations.name
    -> GROUP BY value;
+--------+--------+------------+
| name   | value  | SUM(value) |
+--------+--------+------------+
| banana |   0.10 |       0.20 |
| brick  |   1.00 |       3.00 |
| lizard |  10.00 |      10.00 |
| tree   | 200.00 |     200.00 |
+--------+--------+------------+
4 rows in set (0.00 sec)
```

همانطور که می‌بینید، در خروجی این دستور کلیه‌ی اشیایی که هم در جدول objects و هم در جدول locations بودند، قیمت تکی هر کدام و در نهایت مجموع قیمت کل هر شی به ما نشان داده شد.

پاک کردن اطلاعات

گاهی اوقات لازم است که برخی از اطلاعات را پاک کنیم. برای پاک کردن یک ردیف، از دستور DELETE به صورت زیر استفاده می‌کنیم:

```
DELETE FROM table WHERE conditions;
```

فرض کنیم می‌خواهیم از جدول locations، ردیف مربوط به tree را حذف کنیم. برای این کار:

```
MariaDB [linux]> select * from locations;
+------+--------+-------------+------+
| id   | name   | location    | cond |
+------+--------+-------------+------+
|    1 | banana | kitchen     |    9 |
|    2 | banana | kitchen     |    8 |
|    3 | tree   | backyard    |    2 |
|    4 | brick  | garage      |   10 |
|    5 | brick  | garage      |    9 |
|    6 | brick  | backyard    |    9 |
|    7 | lizard | living room |    8 |
+------+--------+-------------+------+
7 rows in set (0.00 sec)

MariaDB [linux]> DELETE FROM locations
    -> WHERE name='tree' AND location='backyard';
Query OK, 1 row affected (0.01 sec)
```

همانطور که می‌بینید، با این که فقط یک ردیف مربوط به tree داشتیم، هنگام مشخص کردن شرط با کلیدواژه‌ی WHERE، ما علاوه بر name، location را نیز مشخص کردیم. بهتر است موقع پاک کردن یک ردیف، تا حد ممکن شرط را جزئی و دقیق عنوان کنیم. علاوه بر این، پیشنهاد می‌شود که به قبل از اجرای

دستور DELETE، از دستور SELECT استفاده کنیم تا چیزی که قرار است پاک کنیم را مشاهده کرده و از عدم نیاز به آن اطلاعات، مطمئن شویم.

بیایید از پاک شدن این ردیف اطمینان حاصل کنیم:

```
MariaDB [linux]> SELECT * FROM locations;
+------+--------+-------------+------+
| id   | name   | location    | cond |
+------+--------+-------------+------+
|    1 | banana | kitchen     |    9 |
|    2 | banana | kitchen     |    8 |
|    4 | brick  | garage      |   10 |
|    5 | brick  | garage      |    9 |
|    6 | brick  | backyard    |    9 |
|    7 | lizard | living room |    8 |
+------+--------+-------------+------+
6 rows in set (0.00 sec)
```

برای پاک کردن کل ردیف‌های موجود در جدول، می‌توانیم از DELETE * FROM *table*; استفاده کنیم. این دستور کلیه‌ی ردیف‌های موجود در جدول *table* را پاک می‌کند، اما هنوز می‌توانیم اطلاعات جدیدی درون آن قرار دهیم. برای این که یک جدول به صورت کامل از دیتابیس حذف شود (یعنی ردیف و ستون و...)، از دستور DROP TABLE *table*; استفاده می‌کنیم. این دستور، جدول *table* را به طور کامل از دیتابیس حذف می‌کند و ما دیگر نمی‌توانیم اطلاعاتی درون آن وارد کنیم.