SQL？简单地说，SQL就是访问和处理关系数据库的计算机标准语言。也就是说，无论用什么编程语言（Java、Python、C++……）编写程序，只要涉及到操作关系数据库，比如，一个电商网站需要把用户和商品信息存入数据库，或者一个手机游戏需要把用户的道具、通关信息存入数据库，都必须通过SQL来完成。

SQL数据库仍然承担了各种应用程序的核心数据存储，而NoSQL数据库作为SQL数据库的补充，两者不再是二选一的问题，而是主从关系。所以，无论使用哪种编程语言，无论是Web开发、游戏开发还是手机开发，掌握SQL，是所有软件开发人员所必须的。

# 关系数据库概述

## 一、为什么需要数据库？

应用程序需要保存用户的数据，比如Word需要把用户文档保存起来，以便下次继续编辑或者拷贝到另一台电脑。

要保存用户的数据，一个最简单的方法是把用户数据写入文件。例如，要保存一个班级所有学生的信息，可以向文件中写入一个CSV文件：

id,name,gender,score

1,小明,M,90

2,小红,F,95

3,小军,M,88

4,小丽,F,88

如果要保存学校所有班级的信息，可以写入另一个CSV文件。

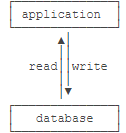
随着应用程序的功能越来越复杂，数据量越来越大，如何管理这些数据就成了大问题：

* 读写文件并解析出数据需要大量重复代码；
* 从成千上万的数据中快速查询出指定数据需要复杂的逻辑。

如果每个应用程序都各自写自己的读写数据的代码，一方面效率低，容易出错，另一方面，每个应用程序访问数据的接口都不相同，数据难以复用。

所以，数据库作为一种专门管理数据的软件就出现了。应用程序不需要自己管理数据，而是通过数据库软件提供的接口来读写数据。不关心数据如何存储到文件

编写应用程序的时候，数据读写的功能就被大大地简化了。

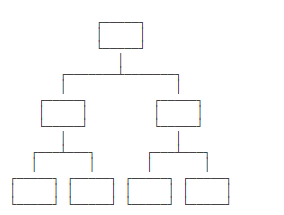


## 二、数据模型

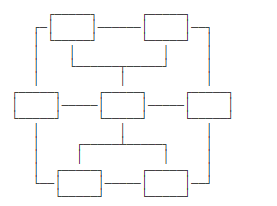
数据库按照数据结构来组织、存储和管理数据，实际上，数据库一共有三种模型：

* 层次模型
* 网状模型
* 关系模型

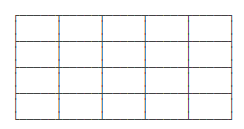
层次模型就是以“上下级”的层次关系来组织数据的一种方式，层次模型的数据结构看起来就像一颗树：



网状模型把每个数据节点和其他很多节点都连接起来，它的数据结构看起来就像很多城市之间的路网：



关系模型把数据看作是一个二维表格，任何数据都可以通过行号+列号来唯一确定，它的数据模型看起来就是一个Excel表：



最终，基于关系模型的关系数据库获得了绝对市场份额

## 三、关系数据库获得了最广泛的应用

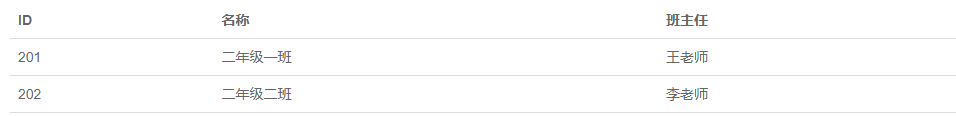
相比层次模型和网状模型，关系模型理解和使用起来最简单。

关系数据库的关系模型是基于数学理论建立的。我们把域（Domain）定义为一组具有相同数据类型的值的集合，给定一组域D1,D2,...,Dn，它们的笛卡尔集定义为D1×D2×……×Dn={(d1,d2,...,dn)|di∈Di,i=1,2,...,n}， 而D1×D2×……×Dn的子集叫作在域D1,D2,...,Dn上的关系，表示为R(D1,D2,...,Dn)，这里的R表示$#%&^@!&$#;!~％¥%:(……算了，根本讲不明白，大家也不用理解。

基于数学理论的关系模型虽然讲起来挺复杂，但是，基于日常生活的关系模型却十分容易理解。我们以学校班级为例，一个班级的学生就可以用一个表格存起来，并且定义如下：



其中，班级ID对应着另一个班级表：



通过给定一个班级名称，可以查到一条班级记录，根据班级ID，又可以查到多条学生记录，这样，二维表之间就通过ID映射建立了“一对多”关系。

## 数据类型

对于一个关系表，除了定义每一列的名称外，还需要定义每一列的数据类型。关系数据库支持的标准数据类型包括数值、字符串、时间等：



通常来说，BIGINT能满足整数存储的需求，VARCHAR(N)能满足字符串存储的需求，这两种类型是使用最广泛的。

## 六、主流关系数据库

目前，主流的关系数据库主要分为以下几类：

1. 商用数据库，例如：[Oracle](https://www.oracle.com/" \t "https://www.liaoxuefeng.com/wiki/1177760294764384/_blank)，[SQL Server](https://www.microsoft.com/sql-server/" \t "https://www.liaoxuefeng.com/wiki/1177760294764384/_blank)，[DB2](https://www.ibm.com/db2/" \t "https://www.liaoxuefeng.com/wiki/1177760294764384/_blank)等；
2. 开源数据库，例如：[MySQL](https://www.mysql.com/" \t "https://www.liaoxuefeng.com/wiki/1177760294764384/_blank)，[PostgreSQL](https://www.postgresql.org/" \t "https://www.liaoxuefeng.com/wiki/1177760294764384/_blank)等；
3. 桌面数据库，以微软[Access](https://products.office.com/access" \t "https://www.liaoxuefeng.com/wiki/1177760294764384/_blank)为代表，适合桌面应用程序使用；
4. 嵌入式数据库，以[Sqlite](https://sqlite.org/" \t "https://www.liaoxuefeng.com/wiki/1177760294764384/_blank)为代表，适合手机应用和桌面程序。

## SQL

### 什么是SQL？

### SQL是结构化查询语言的缩写，用来访问和操作数据库系统。SQL语句既可以查询数据库中的数据，也可以添加、更新和删除数据库中的数据，还可以对数据库进行管理和维护操作。不同的数据库，都支持SQL，这样，我们通过学习SQL这一种语言，就可以操作各种不同的数据库。

虽然SQL已经被ANSI组织定义为标准，不幸地是，各个不同的数据库对标准的SQL支持不太一致。并且，大部分数据库都在标准的SQL上做了扩展。也就是说，如果只使用标准SQL，理论上所有数据库都可以支持，但如果使用某个特定数据库的扩展SQL，换一个数据库就不能执行了。例如，Oracle把自己扩展的SQL称为PL/SQL，Microsoft把自己扩展的SQL称为T-SQL。

标准SQL的核心功能，那么所有数据库通常都可以执行。不常用的SQL功能，不同的数据库支持的程度都不一样。而各个数据库支持的各自扩展的功能，通常我们把它们称之为“方言”。

SQL语言定义了这么几种操作数据库的能力：

****DDL：Data Definition Language****

DDL允许用户定义数据，也就是创建表、删除表、修改表结构这些操作。通常，DDL由数据库管理员执行。

****DML：Data Manipulation Language****

DML为用户提供添加、删除、更新数据的能力，这些是应用程序对数据库的日常操作。

****DQL：Data Query Language****

DQL允许用户查询数据，这也是通常最频繁的数据库日常操作。

## 八、语法特点

SQL语言关键字不区分大小写！！！但是，针对不同的数据库，对于表名和列名，有的数据库区分大小写，有的数据库不区分大小写。同一个数据库，有的在Linux上区分大小写，有的在Windows上不区分大小写。

所以，本教程约定：SQL关键字总是大写，以示突出，表名和列名均使用小写。

# 安装MySQL

MySQL是目前应用最广泛的开源关系数据库。MySQL最早是由瑞典的MySQL AB公司开发，该公司在2008年被SUN公司收购，紧接着，SUN公司在2009年被Oracle公司收购，所以MySQL最终就变成了Oracle旗下的产品。

和其他关系数据库有所不同的是，MySQL本身实际上只是一个SQL接口，它的内部还包含了多种数据引擎，常用的包括：

* InnoDB：由Innobase Oy公司开发的一款支持事务的数据库引擎，2006年被Oracle收购；
* MyISAM：MySQL早期集成的默认数据库引擎，不支持事务。

**MySQL接口和数据库引擎的关系就好比某某浏览器和浏览器引擎（IE引擎或Webkit引擎）的关系。**对用户而言，切换浏览器引擎不影响浏览器界面，切换MySQL引擎不影响自己写的应用程序使用MySQL的接口。

使用MySQL时，不同的表还可以使用不同的数据库引擎。如果你不知道应该采用哪种引擎，记住总是选择InnoDB就好了。

因为MySQL一开始就是开源的，所以基于MySQL的开源版本，又衍生出了各种版本：

### MariaDB

由MySQL的创始人创建的一个开源分支版本，使用XtraDB引擎。

### Aurora

由Amazon改进的一个MySQL版本，专门提供给在AWS托管MySQL用户，号称5倍的性能提升。

### PolarDB

由Alibaba改进的一个MySQL版本，专门提供给在[阿里云](https://promotion.aliyun.com/ntms/yunparter/invite.html?userCode=cz36baxa" \t "https://www.liaoxuefeng.com/wiki/1177760294764384/_blank)托管的MySQL用户，号称6倍的性能提升。

而MySQL官方版本又分了好几个版本：

* Community Edition：社区开源版本，免费；
* Standard Edition：标准版；
* Enterprise Edition：企业版；
* Cluster Carrier Grade Edition：集群版。

以上版本的功能依次递增，价格也依次递增。不过，功能增加的主要是监控、集群等管理功能，对于基本的SQL功能是完全一样的。

所以使用MySQL就带来了一个巨大的好处：可以在自己的电脑上安装免费的Community Edition版本，进行学习、开发、测试，部署的时候，可以选择付费的高级版本，或者云服务商提供的兼容版本，而不需要对应用程序本身做改动。

## 安装MySQL

要在Windows或Mac上安装MySQL，首先从MySQL官方网站下载最新的MySQL Community Server版本：

[https://dev.mysql.com/downloads/mysql/](https://dev.mysql.com/downloads/mysql/" \t "https://www.liaoxuefeng.com/wiki/1177760294764384/_blank)

选择对应的操作系统版本，下载安装即可。在安装过程中，MySQL会自动创建一个root用户，并提示输入root口令。

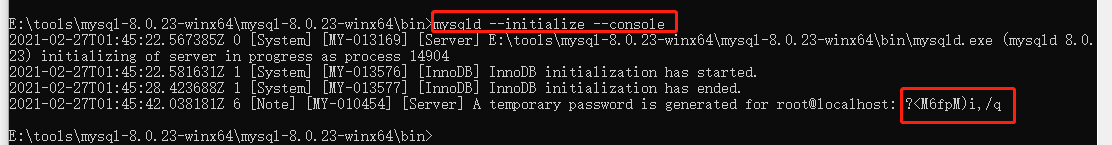
要在Linux上安装MySQL，可以使用发行版的包管理器。例如，Debian和Ubuntu用户可以简单地通过命令apt-get install mysql-server安装最新的MySQL版本。

zip格式安装

(1)打开cmd，以管理员权限运行  
(2) 初始化数据库  
cmd中进入 MySQL安装目录的 bin 目录下，执行命令：

|  |  |
| --- | --- |
| 1 | mysqld --initialize --console |

执行完成后，会打印 root 用户的初始默认密码，比如：



最后的： ?<M6fpM)i,/q 就是初始密码（不含首位空格）。在没有更改密码前，需要记住这个密码，后续登录需要用到。

(3) 安装服务：  
在MySQL安装目录的 bin 目录下执行命令：

|  |  |
| --- | --- |
| 1 | mysqld --install [服务名] |

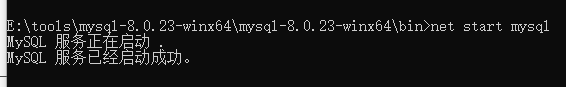
不写服务名时，默认是服务名为mysql。  
执行成功的情况如下：

IMG_257

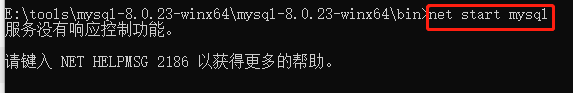
安装完成之后，就可以通过命令net start mysql启动MySQL的服务了。

|  |  |
| --- | --- |
| 1 | net start mysql |

成功后会出现：



若不成功，启动服务如果出现一下情况：



## 运行MySQL

MySQL安装后会自动在后台运行。为了验证MySQL安装是否正确，我们需要通过mysql这个命令行程序来连接MySQL服务器。

在命令提示符下输入mysql -u root -p，然后输入口令，如果一切正确，就会连接到MySQL服务器，同时提示符变为mysql>。

输入exit退出MySQL命令行。注意，MySQL服务器仍在后台运行。

# 关系模型

关系数据库是建立在关系模型上的。而关系模型本质上就是若干个存储数据的二维表，可以把它们看作很多Excel表。

表的每一行称为记录（Record），记录是一个逻辑意义上的数据。

表的每一列称为字段（Column），同一个表的每一行记录都拥有相同的若干字段。

字段定义了数据类型（整型、浮点型、字符串、日期等），以及是否允许为NULL。注意NULL表示字段数据不存在。一个整型字段如果为NULL不表示它的值为0，同样的，一个字符串型字段为NULL也不表示它的值为空串''。

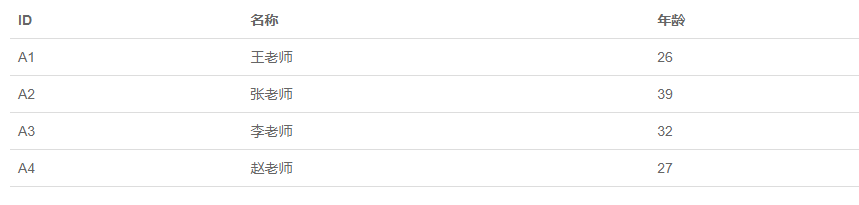
 通常情况下，字段应该避免允许为NULL。不允许为NULL可以简化查询条件，加快查询速度，也利于应用程序读取数据后无需判断是否为NULL。

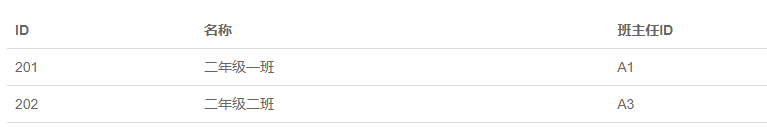
和Excel表有所不同的是，关系数据库的表和表之间需要建立“一对多”，“多对一”和“一对一”的关系，这样才能够按照应用程序的逻辑来组织和存储数据。



反过来，如果我们先在学生表中定位了一行记录，例如ID=1的小明，要确定他的班级，只需要根据他的“班级ID”对应的值201找到班级表中ID=201的记录，即二年级一班。所以，学生表和班级表是“多对一”的关系。

如果我们把班级表分拆得细一点，例如，单独创建一个教师表：

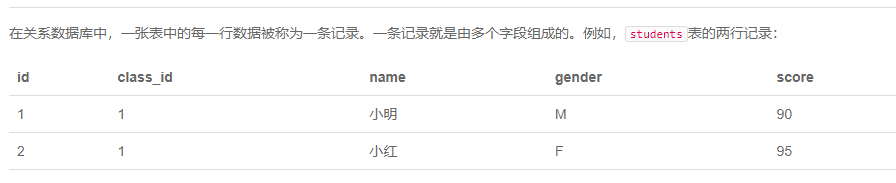


班级表只存储教师ID：

这样，一个班级总是对应一个教师，班级表和教师表就是“一对一”关系。

在关系数据库中，关系是通过主键和外键来维护的。我们在后面会分别深入讲解。

## 主键

每一条记录都包含若干定义好的字段。

同一个表的所有记录都有相同的字段定义。

对于关系表，有个很重要的约束，就是任意两条记录不能重复。

不能重复不是指两条记录不完全相同，而是指能够通过某个字段唯一区分出不同的记录，这个字段被称为主键。

例如，假设我们把name字段作为主键，那么通过名字小明或小红就能唯一确定一条记录。但是，这么设定，就没法存储同名的同学了，因为插入相同主键的两条记录是不被允许的。

对主键的要求，最关键的一点是：记录一旦插入到表中，主键最好不要再修改，因为主键是用来唯一定位记录的，修改了主键，会造成一系列的影响。

由于主键的作用十分重要，如何选取主键会对业务开发产生重要影响。如果我们以学生的身份证号作为主键，似乎能唯一定位记录。然而，身份证号也是一种业务场景，如果身份证号升位了，或者需要变更，作为主键，不得不修改的时候，就会对业务产生严重影响。

所以，**选取主键的一个基本原则是：不使用任何业务相关的字段作为主键。**

因此，身份证号、手机号、邮箱地址这些看上去可以唯一的字段，均不可用作主键。

作为**主键最好是完全业务无关的字段，我们一般把这个字段命名为id。常见的可作为id字段的类型有：**

*自增整数类型：数据库会在插入数据时自动为每一条记录分配一个自增整数，这样我们就完全不用担心主键重复，也不用自己预先生成主键；*

*全局唯一GUID类型：使用一种全局唯一的字符串作为主键，类似8f55d96b-8acc-4636-8cb8-76bf8abc2f57。GUID算法通过网卡MAC地址、时间戳和随机数保证任意计算机在任意时间生成的字符串都是不同的，大部分编程语言都内置了GUID算法，可以自己预算出主键。*

*对于大部分应用来说，通常自增类型的主键就能满足需求。我们在students表中定义的主键也是BIGINT NOT NULL AUTO\_INCREMENT类型。*

*如果使用INT自增类型，那么当一张表的记录数超过2147483647（约21亿）时，会达到上限而出错。使用BIGINT自增类型则可以最多约922亿亿条记录。*

## 联合主键

关系数据库实际上还允许通过多个字段唯一标识记录，即两个或更多的字段都设置为主键，这种主键被称为联合主键。

对于联合主键，允许一列有重复，只要不是所有主键列都重复即可：

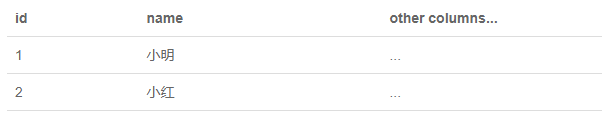


上述表的id\_num和id\_type这两列作为联合主键，那么上面的3条记录都是允许的，因为没有两列主键组合起来是相同的。

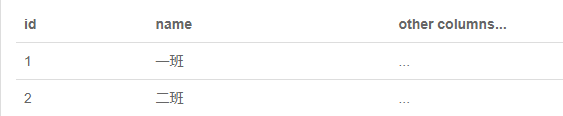
没有必要的情况下，我们尽量不使用联合主键，因为它给关系表带来了复杂度的上升。

## 外键

当我们用主键唯一标识记录时，我们就可以在students表中确定任意一个学生的记录：



我们还可以在classes表中确定任意一个班级记录：



如何确定students表的一条记录，例如，id=1的小明，属于哪个班级呢？

由于一个班级可以有多个学生，在关系模型中，这两个表的关系可以称为“一对多”，即一个classes的记录可以对应多个students表的记录。

表达这种一对多的关系，我们需要在students表中加入一列class\_id，让它的值与classes表的某条记录相对应：



可以根据class\_id这个列直接定位出一个students表的记录应该对应到classes的哪条记录。

例如：

* 小明的class\_id是1，因此，对应的classes表的记录是id=1的一班；
* 小红的class\_id是1，因此，对应的classes表的记录是id=1的一班；
* 小白的class\_id是2，因此，对应的classes表的记录是id=2的二班。

**在students表中，通过class\_id的字段，可以把数据与另一张表关联起来，这种列称为外键。**

外键并不是通过列名实现的，而是通过定义外键约束实现的：

**ALTER** **TABLE** students**ADD** **CONSTRAINT** fk\_class\_id **FOREIGN** **KEY** (class\_id)**REFERENCES** classes (id);

外键约束的名称fk\_class\_id可以任意

OREIGN KEY (class\_id)指定了class\_id作为外键

REFERENCES classes (id)指定了这个外键将关联到classes表的id列（即classes表的主键）

通过定义外键约束，关系数据库可以保证无法插入无效的数据。即如果classes表不存在id=99的记录，students表就无法插入class\_id=99的记录。

由于外键约束会降低数据库的性能，大部分互联网应用程序为了追求速度，并不设置外键约束，而是仅靠应用程序自身来保证逻辑的正确性。这种情况下，class\_id仅仅是一个普通的列，只是它起到了外键的作用而已

**要删除一个外键约束，也是通过ALTER TABLE实现的：**

**ALTER** **TABLE** students **DROP** **FOREIGN** **KEY** fk\_class\_id;

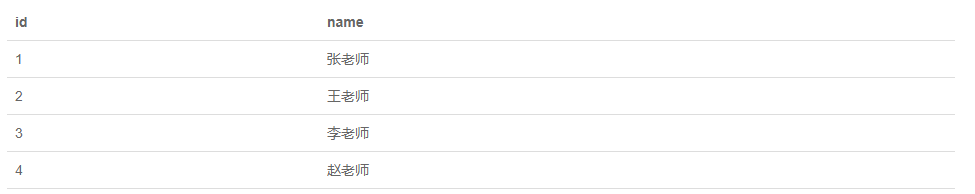
注意：删除外键约束并没有删除外键这一列。删除列是通过DROP COLUMN ...实现的。

## 多对多

通过一个表的外键关联到另一个表，我们可以定义出一对多关系。有些时候，还需要定义“多对多”关系。例如，一个老师可以对应多个班级，一个班级也可以对应多个老师，因此，班级表和老师表存在多对多关系。

多对多关系实际上是通过两个一对多关系实现的，即通过一个中间表，关联两个一对多关系，就形成了多对多关系：

teachers表：







通过**中间表**teacher\_class可知teachers到classes的关系：

* id=1的张老师对应id=1,2的一班和二班；
* id=2的王老师对应id=1,2的一班和二班；
* id=3的李老师对应id=1的一班；
* id=4的赵老师对应id=2的二班。

同理可知classes到teachers的关系：

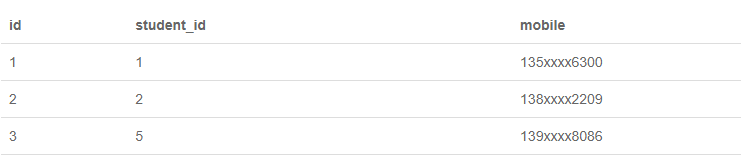
* id=1的一班对应id=1,2,3的张老师、王老师和李老师；
* id=2的二班对应id=1,2,4的张老师、王老师和赵老师；

因此，通过中间表，我们就定义了一个“多对多”关系。

## 一对一

一对一关系是指，一个表的记录对应到另一个表的唯一一个记录。

例如，students表的每个学生可以有自己的联系方式，如果把联系方式存入另一个表contacts，我们就可以得到一个“一对一”关系：



业务允许，完全可以把两个表合为一个表。但是，有些时候，如果某个学生没有手机号，那么，contacts表就不存在对应的记录。实际上，一对一关系准确地说，是contacts表一对一对应students表。

还有一些应用会把**一个大表拆成两个一对一的表**，目的是**把经常读取和不经常读取的字段分开**，以**获得更高的性能**。例如，把一个大的用户表分拆为用户基本信息表user\_info和用户详细信息表user\_profiles，**大部分时候，只需要查询user\_info表**，并不需要查询user\_profiles表，这样就**提高了查询速度。**

## 索引

**在关系数据库中，如果有上万甚至上亿条记录，在查找记录的时候，想要获得非常快的速度，就需要使用索引。**

***索引是关系数据库中对某一列或多个列的值进行预排序的数据结构***。通过使用索引，可以让数据库系统不必扫描整个表，而是直接定位到符合条件的记录，这样就大大加快了查询速度。



如果要经常根据score列进行查询，就可以对score列创建索引：

**ALTER** **TABLE** students **ADD** INDEX idx\_score (score);

使用ADD INDEX idx\_score (score)就创建了一个名称为idx\_score，使用列score的索引。索引名称是任意的，索引如果有多列，可以在括号里依次写上，例如：

**ALTER** **TABLE** students **ADD** INDEX idx\_name\_score (name, score);

索引的效率取决于索引列的值是**否散列**，即**该列的值如果越互不相同，那么索引效率越高。反过来，如果记录的列存在大量相同的值**，例如gender列，大约一半的记录值是M，另一半是F，因此，对该列创建索引就没有意义。

可以对一张表创建多个索引。**索引的优点是提高了查询效率**，**缺点是在插入、更新和删除记录时，需要同时修改索引，因此，索引越多，插入、更新和删除记录的速度就越慢。**

对于主键，关系数据库会自动对其创建主键索引。使用主键索引的效率是最高的，因为主键会保证绝对唯一。

## 唯一索引

在设计关系数据表的时候，**看上去唯一的列**，例如身份证号、邮箱地址等，因为他们**具有业务含义**，因此**不宜作为主键**。

但是，***这些列根据业务要求，又具有唯一性约束：即不能出现两条记录存储了同一个身份证号。***这个时候，就可以给该列添加一个唯一索引。例如，我们假设students表的name不能重复：

**ALTER** **TABLE** students **ADD** **UNIQUE** INDEX uni\_name (name);

通过UNIQUE关键字我们就添加了一个唯一索引。

也可以只对某一列添加一个唯一约束而不创建唯一索引：

**ALTER** **TABLE** students **ADD** **CONSTRAINT** uni\_name **UNIQUE** (name);

这种情况下，name列没有索引，但仍然具有唯一性保证。

无论**是否创建索引**，*对于用户和应用程序来说，使用关系数据库不会有任何区别*。这里的意思是说，当我们在数据库中查询时，如果有相应的索引可用，数据库系统就会自动使用索引来提高查询效率，如果没有索引，查询也能正常执行，只是速度会变慢。因此，***索引可以在使用数据库的过程中逐步优化。***

## 在线SQL

在线运行SQL的功能。实际上这是在***浏览器页面运行的一个JavaScript编写的内存型SQL数据库[AlaSQL](http://alasql.org/" \t "https://www.liaoxuefeng.com/wiki/1177760294764384/_blank)***。不必运行MySQL等实际的数据库软件，即可在线编写并执行SQL语句。

# [查询数据](https://www.liaoxuefeng.com/wiki/1177760294764384/1179610544539040)

## [基本查询](https://www.liaoxuefeng.com/wiki/1177760294764384/1179610846971200)

## [条件查询](https://www.liaoxuefeng.com/wiki/1177760294764384/1179663974004192)

## [投影查询](https://www.liaoxuefeng.com/wiki/1177760294764384/1179610872020928)

## [排序](https://www.liaoxuefeng.com/wiki/1177760294764384/1181753232323168)

## [分页查询](https://www.liaoxuefeng.com/wiki/1177760294764384/1217864791925600)

## [聚合查询](https://www.liaoxuefeng.com/wiki/1177760294764384/1179611448454560)

## [多表查询](https://www.liaoxuefeng.com/wiki/1177760294764384/1179664013849760)

## [连接查询](https://www.liaoxuefeng.com/wiki/1177760294764384/1179610888796448)

# [修改数据](https://www.liaoxuefeng.com/wiki/1177760294764384/1179610621264704)

## INSERT

## UPDATE

## DELETE

# [MySQL](https://www.liaoxuefeng.com/wiki/1177760294764384/1246617733597824)

## 管理MYSQL

## 实用SQL语句

# [事务](https://www.liaoxuefeng.com/wiki/1177760294764384/1179611198786848)

## Read Uncommitted

## Read Committed

## REPETABLE Read

## Serializable

# [开发集成](https://www.liaoxuefeng.com/wiki/1177760294764384/1179611234827808)

## [期末总结](https://www.liaoxuefeng.com/wiki/1177760294764384/1179611348902464)

### 小结

1.主键是关系表中记录的唯一标识。主键的选取非常重要：主键不要带有业务含义，而应该使用BIGINT自增或者GUID类型。主键也不应该允许NULL。可以使用多个列作为联合主键，但联合主键并不常用。

2.通过中间表，我们就定义了一个“多对多”关系。

多对多关系实际上是通过两个一对多关系实现的，即通过一个中间表，关联两个一对多关系，就形成了多对多关系

通过大表拆分，将经常的读取，和不经常读取的表拆分，获取一对一关系的表，提高查询速度

关系数据库通过外键可以实现一对多、多对多和一对一的关系。外键既可以通过数据库来约束，也可以不设置约束，仅依靠应用程序的逻辑来保证

1. 通过对数据库表创建索引，可以提高查询速度。

通过创建唯一索引，可以保证某一列的值具有唯一性。

数据库索引对于用户和应用程序来说都是透明的。