安装过程重点记录：

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。

1配置环境变量后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>。然后再输入一系列SQL命令。

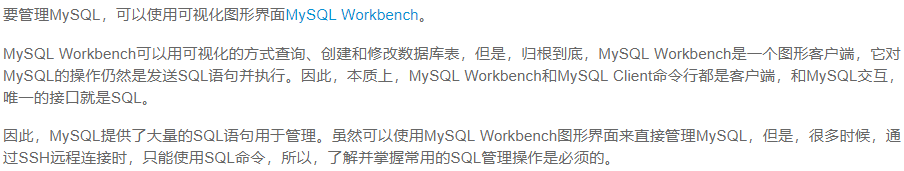
2 安装完MySQL后，除了MySQL Server，即真正的MySQL服务器外，还附赠一个MySQL Client程序。MySQL Client是一个命令行客户端，可以通过MySQL Client登录MySQL，然后，输入SQL语句并执行。

输入exit断开与MySQL Server的连接并返回到命令提示符。注意EXIT仅仅断开了客户端和服务器的连接，MySQL服务器仍然继续运行

 MySQL Client的可执行程序是mysql，MySQL Server的可执行程序是mysqld。







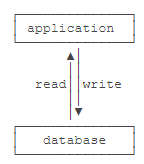
### 1关系数据库概论

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：

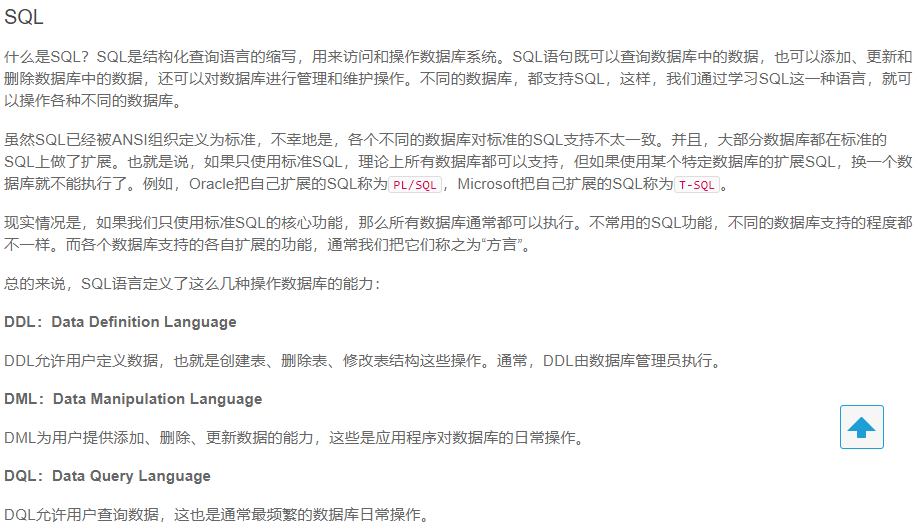
* 读写文件并解析出数据需要大量重复代码；
* 从成千上万的数据中快速查询出指定数据需要复杂的逻辑。

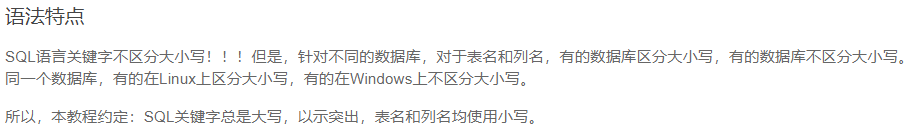
如果每个应用程序都各自写自己的读写数据的代码，一方面效率低，容易出错，另一方面，每个应用程序访问数据的接口都不相同，数据难以复用。

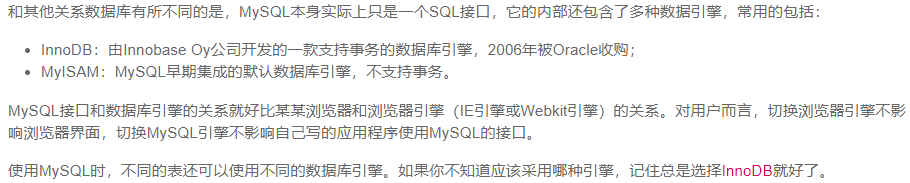
所以，数据库作为一种专门管理数据的软件就出现了。应用程序不需要自己管理数据，而是通过数据库软件提供的接口来读写数据。至于数据本身如何存储到文件，那是数据库软件的事情，应用程序自己并不关心：

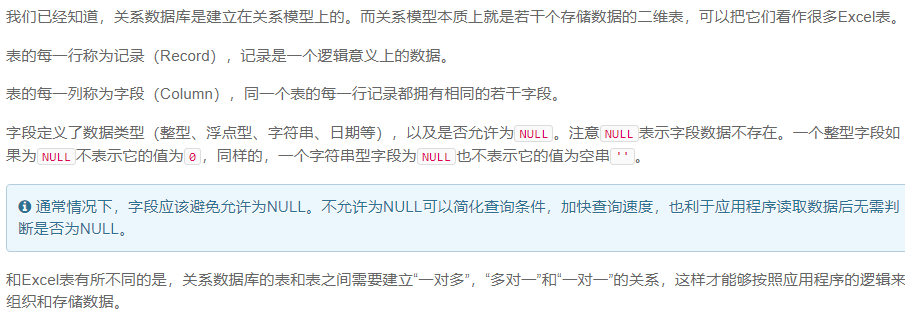


这样一来，编写应用程序的时候，数据读写的功能就被大大地简化了。



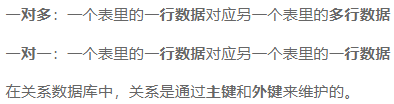










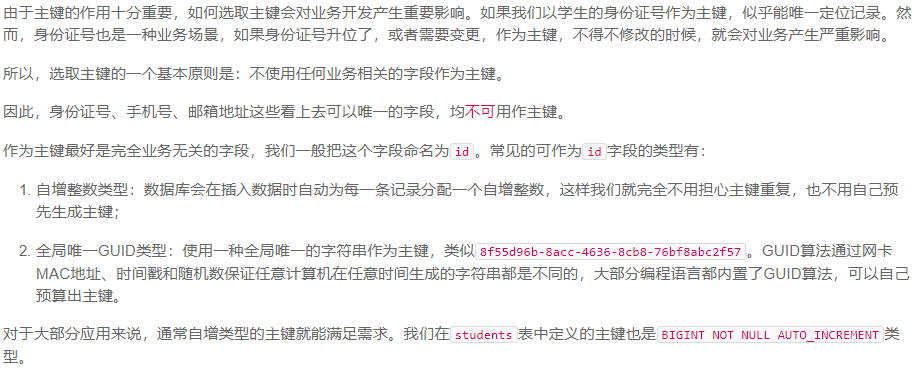


每一条记录都包含若干定义好的字段。同一个表的所有记录都有相同的字段定义。

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。

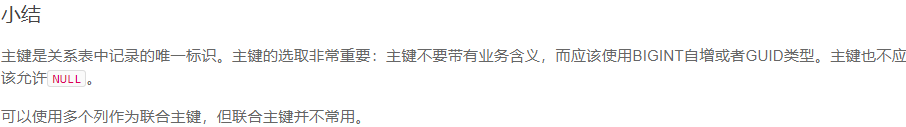
例如，假设我们把name字段作为主键，那么通过名字小明或小红就能唯一确定一条记录。但是，这么设定，就没法存储同名的同学了，因为插入相同主键的两条记录是不被允许的。

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。

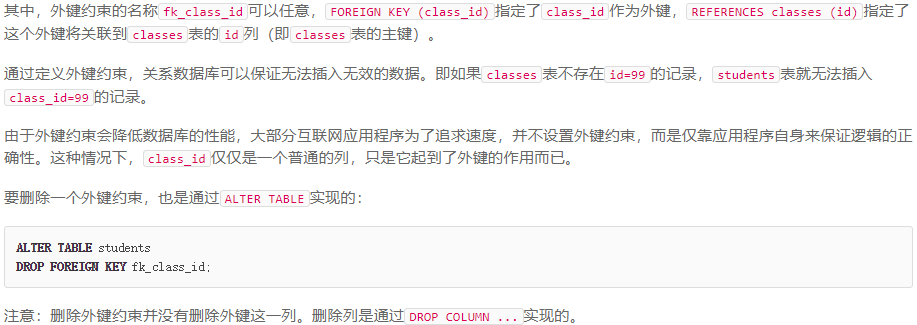






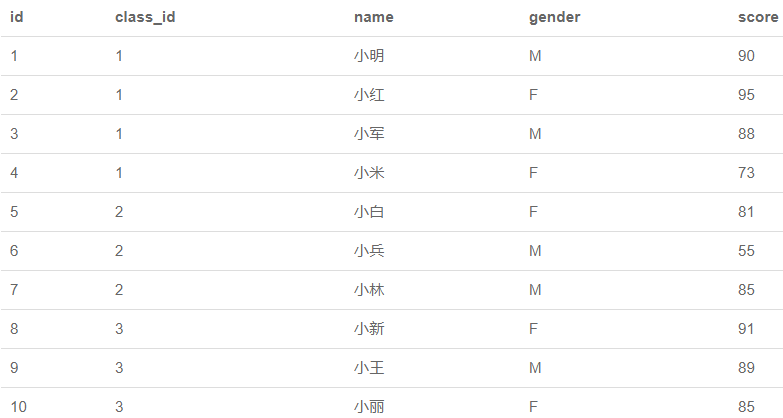






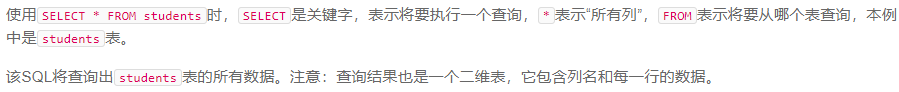
小结

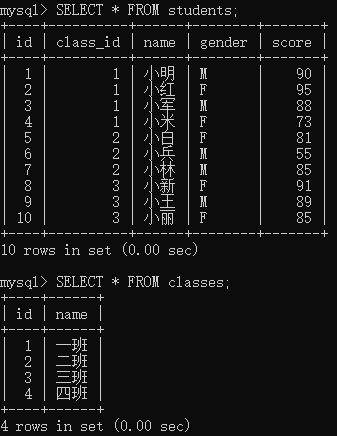
关系数据库通过外键可以实现一对多、多对多和一对一的关系。外键既可以通过数据库来约束，也可以不设置约束，仅依靠应用程序的逻辑来保证。





基本查询：







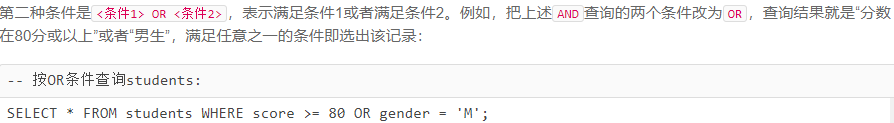
上述查询会直接计算出表达式的结果：300。虽然SELECT可以用作计算，但它并不是SQL的强项。但是，不带FROM子句的SELECT语句有一个有用的用途，就是用来判断当前到数据库的连接是否有效。许多检测工具会执行一条SELECT 1;来测试数据库连接。

**条件查询：**

SELECT语句可以通过WHERE条件来设定查询条件，查询结果是满足查询条件的记录。例如，要指定条件“分数在80分或以上的学生”，写成WHERE条件就是SELECT \* FROM students WHERE score >= 80。

其中，WHERE关键字后面的score >= 80就是条件。score是列名，该列存储了学生的成绩，因此，score >= 80就筛选出了指定条件的记录：



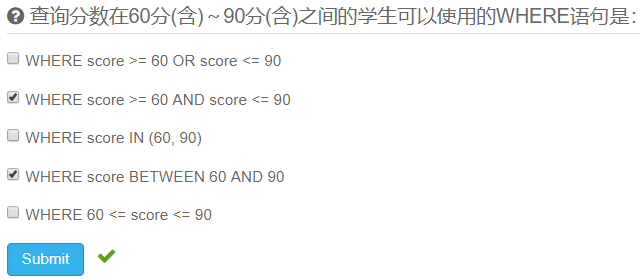


SELECT \* FROM students WHERE NOT class\_id = 2;

上述NOT条件NOT class\_id = 2其实等价于class\_id <> 2，因此，NOT查询不是很常用



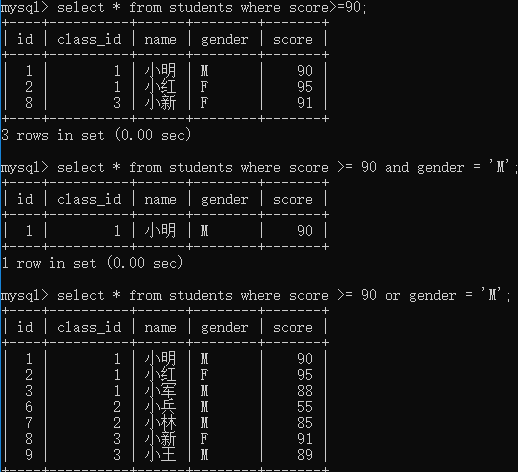




条件查询

通过WHERE条件查询，可以筛选出符合指定条件的记录，而不是整个表的所有记录。

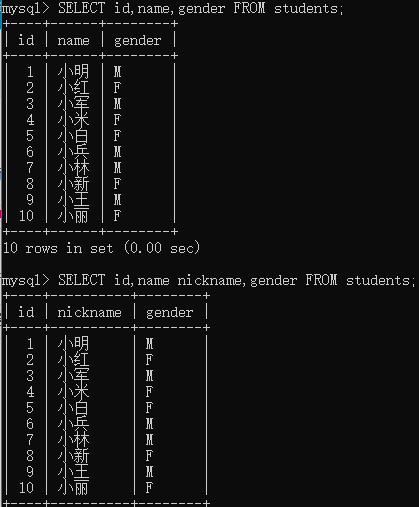
使用SELECT \* FROM <表名> WHERE <条件>可以选出表中的若干条记录。我们注意到返回的二维表结构和原表是相同的，即结果集的所有列与原表的所有列都一一对应



投影查询

使用SELECT \*表示查询表的所有列，使用SELECT 列1, 列2, 列3则可以仅返回指定列，这种操作称为投影。

SELECT语句可以对结果集的列进行重命名



排序：

我们使用SELECT查询时，细心的读者可能注意到，查询结果集通常是按照id排序的，也就是根据主键排序。这也是大部分数据库的做法。如果我们要根据其他条件排序怎么办？可以加上ORDER BY子句。例如按照成绩从低到高进行排序：

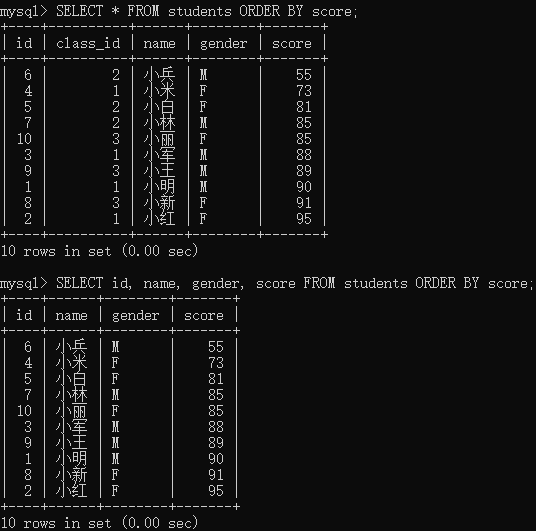




小结

使用ORDER BY可以对结果集进行排序；

可以对多列进行升序、倒序排序



分页：

使用LIMIT <M> OFFSET <N>可以对结果集进行分页，每次查询返回结果集的一部分；

分页查询需要先确定每页的数量和当前页数，然后确定LIMIT和OFFSET的值。

可见，分页查询的关键在于，首先要确定每页需要显示的结果数量pageSize（这里是3），然后根据当前页的索引pageIndex（从1开始），确定LIMIT和OFFSET应该设定的值：

LIMIT总是设定为pageSize；

OFFSET计算公式为pageSize \* (pageIndex - 1)。

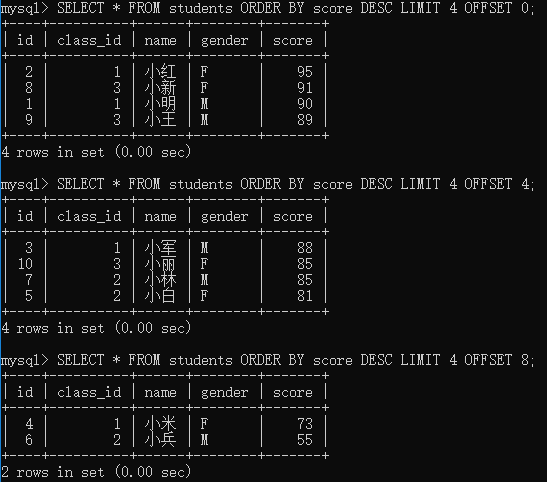
这样就能正确查询出第N页的记录集。

注意：

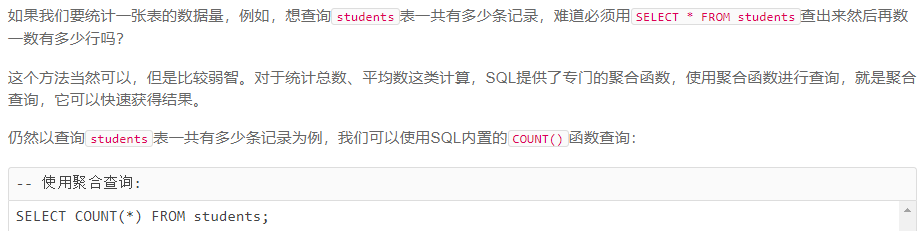
OFFSET是可选的，如果只写LIMIT 15，那么相当于LIMIT 15 OFFSET 0。

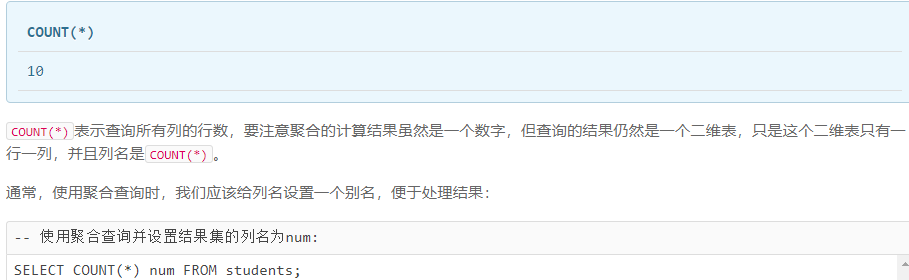
在MySQL中，LIMIT 15 OFFSET 30还可以简写成LIMIT 30, 15。

使用LIMIT <M> OFFSET <N>分页时，随着N越来越大，查询效率也会越来越低。



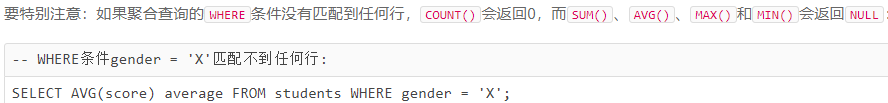
聚合查询：









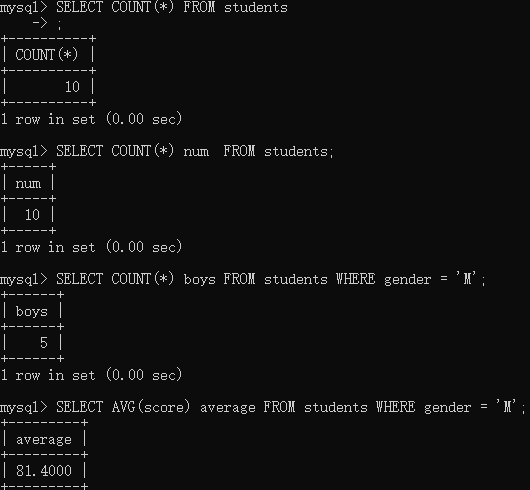


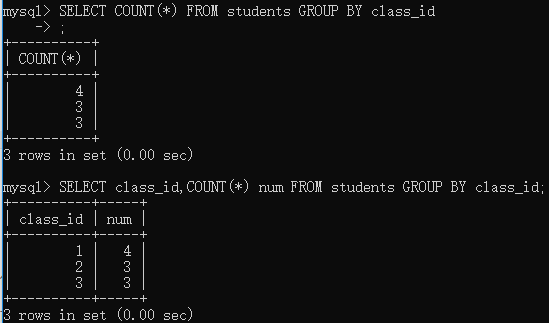


小结:

使用SQL提供的聚合查询，我们可以方便地计算总数、合计值、平均值、最大值和最小值；

聚合查询也可以添加WHERE条件。





多表查询：

SELECT查询不但可以从一张表查询数据，还可以从多张表同时查询数据。查询多张表的语法是：SELECT \* FROM <表1> <表2>。

例如，同时从students表和classes表的“乘积”，即查询数据，可以这么写：

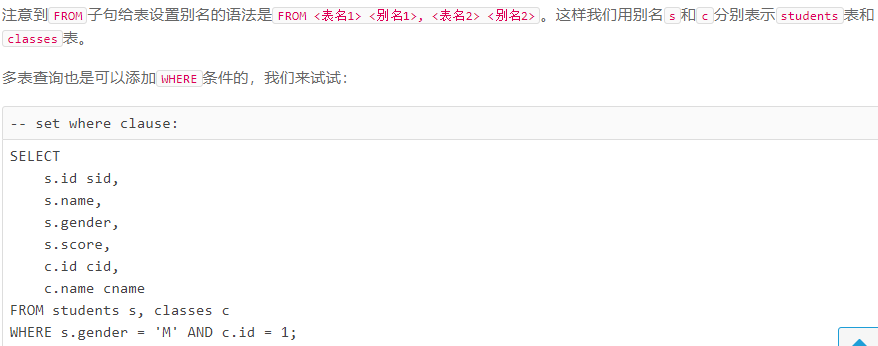




这种一次查询两个表的数据，查询的结果也是一个二维表，它是students表和classes表的“乘积”，即students表的每一行与classes表的每一行都两两拼在一起返回。结果集的列数是students表和classes表的列数之和，行数是students表和classes表的行数之积。

这种多表查询又称笛卡尔查询，使用笛卡尔查询时要非常小心，由于结果集是目标表的行数乘积，对两个各自有100行记录的表进行笛卡尔查询将返回1万条记录，对两个各自有1万行记录的表进行笛卡尔查询将返回1亿条记录。

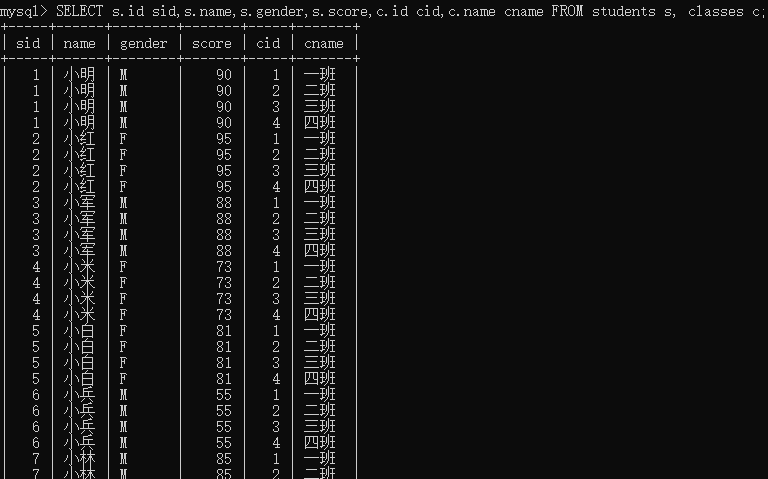
你可能还注意到了，上述查询的结果集有两列id和两列name，两列id是因为其中一列是students表的id，而另一列是classes表的id，但是在结果集中，不好区分。两列name同理



小结

使用多表查询可以获取M x N行记录；

多表查询的结果集可能非常巨大，要小心使用。



连接查询：

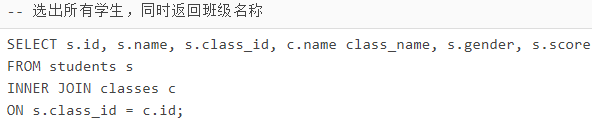
连接查询是另一种类型的多表查询。连接查询对多个表进行JOIN运算，简单地说，就是先确定一个主表作为结果集，然后，把其他表的行有选择性地“连接”在主表结果集上。

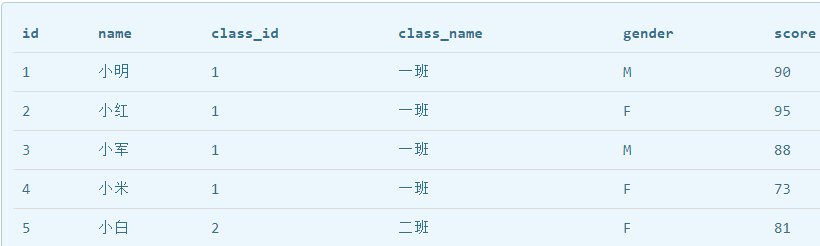
假设我们希望结果集同时包含所在班级的名称，上面的结果集只有class\_id列，缺少对应班级的name列。



现在问题来了，存放班级名称的name列存储在classes表中，只有根据students表的class\_id，找到classes表对应的行，再取出name列，就可以获得班级名称。

这时，连接查询就派上了用场。我们先使用最常用的一种内连接——INNER JOIN来实现：





注意INNER JOIN查询的写法是：

1. 先确定主表，仍然使用FROM <表1>的语法；
2. 再确定需要连接的表，使用INNER JOIN <表2>的语法；
3. 然后确定连接条件，使用ON <条件...>，这里的条件是s.class\_id = c.id，表示students表的class\_id列与classes表的id列相同的行需要连接；
4. 可选：加上WHERE子句、ORDER BY等子句

有RIGHT OUTER JOIN，就有LEFT OUTER JOIN，以及FULL OUTER JOIN。它们的区别是：

INNER JOIN只返回同时存在于两张表的行数据，由于students表的class\_id包含1，2，3，classes表的id包含1，2，3，4，所以，INNER JOIN根据条件s.class\_id = c.id返回的结果集仅包含1，2，3。

RIGHT OUTER JOIN返回右表都存在的行。如果某一行仅在右表存在，那么结果集就会以NULL填充剩下的字段。

LEFT OUTER JOIN则返回左表都存在的行。如果我们给students表增加一行，并添加class\_id=5，由于classes表并不存在id=5的行，所以，LEFT OUTER JOIN的结果会增加一行，对应的class\_name是NULL：

FULL OUTER JOIN，它会把两张表的所有记录全部选择出来，并且，自动把对方不存在的列填充为NULL





