MySQL 元数据：

（1）查询结果信息：SELECT, UPDATE 或 DELETE 语句影响的记录数；

（2）数据库和数据表的信息：包含了数据库及数据表的结构信息；

（3）MySQL 服务器信息：包含了数据库服务器的当前状态，版本号等。

```
# 获取当前数据库中所有可用的表。
my @tables = $dbh->tables ( );
foreach $table (@tables ){
    print "表名  $table\n";
}
```

获取服务器元数据：

命令	描述
SELECT VERSION( )	服务器版本信息
SELECT DATABASE( )	当前数据库名 (或者返回空)
SELECT USER( )	当前用户名
SHOW STATUS	服务器状态
SHOW VARIABLES	服务器配置变量

MySQL 序列的使用：

MySQL 序列是一组整数：1, 2, 3, ...，由于一张数据表只能有一个字段自增主键，如果你想实现其他字段也实现自动增加，就可以使用 MySQL 序列来实现。

使用 AUTO_INCREMENT

获取 AUTO_INCREMENT 值：

在 MySQL 的客户端中你可以使用 SQL 中的 LAST_INSERT_ID( ) 函数来获取最后的插入表中的自增列的值。

PERL 实例：

```
$dbh->do ("INSERT INTO insect (name,date,origin)
VALUES('moth','2001-09-14','windowsill')");
my $seq = $dbh->{mysql_insertid};
```

重置序列：

如果你删除了数据表中的多条记录，并希望对剩下数据的 AUTO_INCREMENT 列进行重新排列，那么你可以通过删除自增的列，然后重新添加来实现。 不过该操作要非常小心，如果在删除的同时又有新记录添加，有可能会出现数据混乱。操作如下所示：

```
mysql> ALTER TABLE insect DROP id;
mysql> ALTER TABLE insect
    -> ADD id INT UNSIGNED NOT NULL AUTO_INCREMENT FIRST,
    -> ADD PRIMARY KEY (id);
```

设置序列的开始值：

一般情况下序列的开始值为 1，但如果你需要指定一个开始值 100，那我们可以通过以下语句来实现：

```
mysql> CREATE TABLE insect
    -> (
    -> id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    -> PRIMARY KEY (id),
    -> name VARCHAR(30) NOT NULL,
    -> date DATE NOT NULL,
    -> origin VARCHAR(30) NOT NULL
)engine=innodb auto_increment=100 charset=utf8;
```

或者你也可以在表创建成功后，通过以下语句来实现：

```
mysql> ALTER TABLE t AUTO_INCREMENT = 100;
```

MySQL 处理重复数据：

防止表中出现重复数据：

可以在 MySQL 数据表中设置指定的字段为 PRIMARY KEY（主键） 或者 UNIQUE（唯一） 索引来保证数据的唯一性。

如果想设置表中字段 first_name, last_name 数据不能重复，可以设置双主键模式来设置数据的唯一性，如果设置了双主键，那么那个键的默认值不能为 NULL，可设置为 NOT NULL。如下所示：

```
CREATE TABLE person_tbl
(
    first_name CHAR(20) NOT NULL,
    last_name CHAR(20) NOT NULL,
```

```
    sex CHAR(10),
    PRIMARY KEY (last_name, first_name)
);
```

如果设置了唯一索引，那么在插入重复数据时，SQL 语句将无法执行成功，并抛出错。

以下实例使用了 INSERT IGNORE INTO，执行后不会出错，也不会向数据表中插入重复数据：

```
mysql> INSERT IGNORE INTO person_tbl (last_name, first_name)
    -> VALUES( 'Jay', 'Thomas');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT IGNORE INTO person_tbl (last_name, first_name)
    -> VALUES( 'Jay', 'Thomas');
Query OK, 0 rows affected (0.00 sec)
```

另一种设置数据的唯一性方法是添加一个 UNIQUE 索引，如下所示：

```
CREATE TABLE person_tbl
(
    first_name CHAR(20) NOT NULL,
    last_name CHAR(20) NOT NULL,
    sex CHAR(10),
    UNIQUE (last_name, first_name)
);
```

统计重复数据：

以下实例将统计表中 first_name 和 last_name 的重复记录数：

```
mysql> SELECT COUNT(*) as repetitions, last_name, first_name
    -> FROM person_tbl
    -> GROUP BY last_name, first_name
    -> HAVING repetitions > 1;
```

以上查询语句将返回 person_tbl 表中重复的记录数。一般情况下，查询重复的值，请执行以下操作：（1）确定哪一列包含的值可能会重复；（2）在列选择列表使用 COUNT(*)列出的那些列；（3）在 GROUP BY 子句中列出的列；（4）HAVING 子句设置重复数大于 1。

过滤重复数据：

可以在 SELECT 语句中使用 DISTINCT 关键字来过滤重复数据：

```
mysql> SELECT DISTINCT last_name, first_name
    -> FROM person_tbl;
```

也可以使用 GROUP BY 来读取数据表中不重复的数据：

```
mysql> SELECT last_name, first_name
    -> FROM person_tbl
    -> GROUP BY (last_name, first_name);
```

删除重复数据：

```
mysql> CREATE TABLE tmp SELECT last_name, first_name, sex FROM person_tbl
GROUP BY (last_name, first_name, sex);
mysql> DROP TABLE person_tbl;
mysql> ALTER TABLE tmp RENAME TO person_tbl;
```

当然也可以在数据表中添加 INDEX（索引）和 PRIMAY KEY（主键）这种简单的方法来删除表中的重复记录。方法如下：

```
mysql> ALTER IGNORE TABLE person_tbl
    -> ADD PRIMARY KEY (last_name, first_name);
```