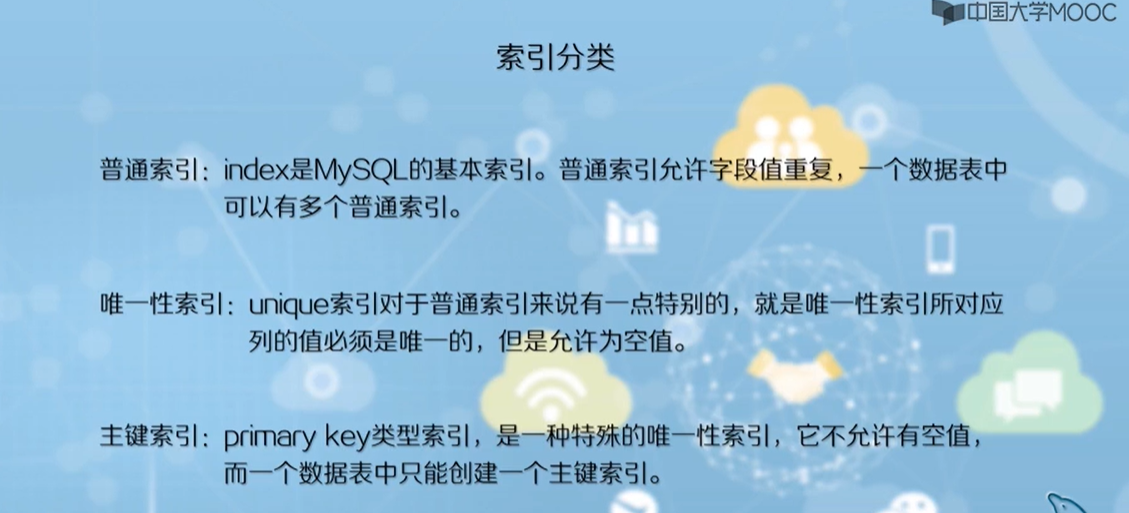
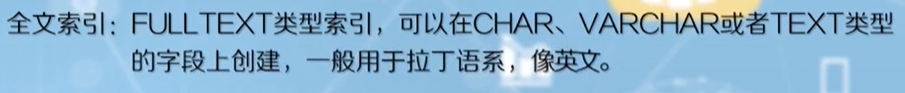
\*：通用所有字段。例如select\*from table\_name :表示查询表中 所有字段

as 后添加别名。

Truncate table 表名：用于完全清空一个表。



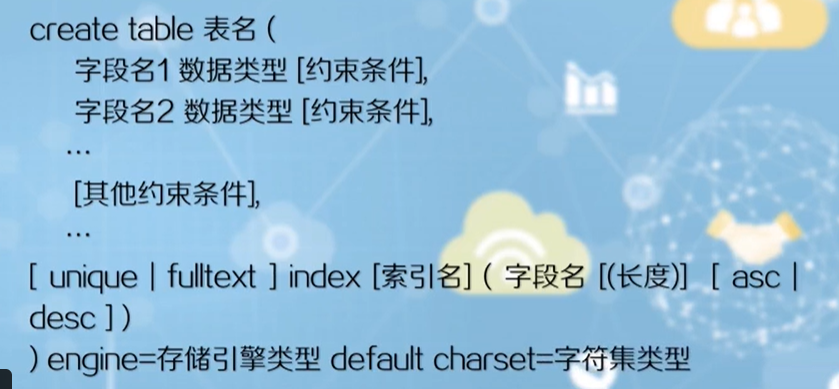




索引设计原则：

1. 索引并非越多越好
2. 字段的离散度越高，该字段越合适选作索引的字段
3. 经常排序和分组的数据列、两个表的连接字段应该建立索引
4. 更新频繁的字段不适合创建索引，不会出现在where子句中的字段不应该创建索引

创建索引：建表时创建索引



建表后创建索引：

Alter table table\_name add unique index(字段);添加唯一索引

删除索引：drop index 索引名 on 表名

数据完整性：是指数据库中的数据在逻辑上的一致性、正确性、有效性和相容性。

关系模式中有三种数据完整性：

1. 实体完整性：有主键约束或候选键约束实现。
2. 参照完整性：由外键约束实现
3. 用户自定义完整性：由非空约束、默认值约束和检查约束实现。

**Select 查询、检索 Distinct 不重复 desc 降序 asc 升序**

**group by 分组 order by 排序 limit 有限，限制 offset 偏移量**

**HAVING语句通常与GROUP BY语句联合使用，用来过滤由GROUP BY语句返回的记录集。**

**HAVING语句的存在弥补了WHERE关键字不能与聚合函数联合使用的不足。**

**with rollup关键字会在所有记录的最后加上一条记录，该记录是上面所有记录的总和。**

连接查询：连接查询是关系数据模型的主要特点，连接查询时关系数据库中最主要的查询。

分为内连接和外连接等。

内连接查询（innert join）：

Inner join使用比较运算符（=）根据每个表共有列的列值匹配两个表中的行，其查询结果仅包含符合查询和连接的条件的行。即查询结果为两个表的交集。

注意：多表查询时，为避免混淆，在查询字段前应添加表名称作为前缀

**两个表内连接查询：Select tableA.select\_list, tableB.select\_list from tableA inner join tableB on tableA.key=tableB.key (key 表示两表中意义相同的字段) （后面可以加上条件）**

**三个表内连接查询：select A.select\_list1, A.select\_list2, C.select\_list1, C.select\_list2, B.select\_list from table A inner join table B on A.select\_list1 = B.select\_list inner join table C on B.select\_list = C.select\_list;**

外连接查询（outer join）：mysql支持的外连接有两种类型：左外连接（左连接 left [outer] join） 右外连接（右连接 right [outer] join）

MySQL不能直接支持FULL JOIN，要实现full join，应该使用left join UNION right 的方式。

左连接查询：左连接的查询结果为左表的所有记录以及右表中连接字段相等的记录，如果左表的某行在右表中没有匹配行，则相关联的结果行中，右表的所有选择列均为空值（null）。

**Select tableA.select\_list, tableB.select\_list from tableA left[outer] join tableB on tableA.key=tableB.key;**

右连接查询：右连接查询时左连接的反向连接，查询结果为右表的所有记录以及左表中字段相等的记录，如果没有匹配行就返回null；

**Select tableA.select\_list, tableB.select\_list from tableA right[outer] join tableB on tableA.key=tableB.key;**

完全连接（full join）：完全连接返回左表和右表中的所有记录，包括连接字段相等的记录和不相等的记录。如果左表的某行在右表中没有匹配行，将返回null，反之亦然。

**Select tableA.select\_list, tableB.select\_list from tableA left[outer] join tableB on tableA.key=tableB.key**

**Union**

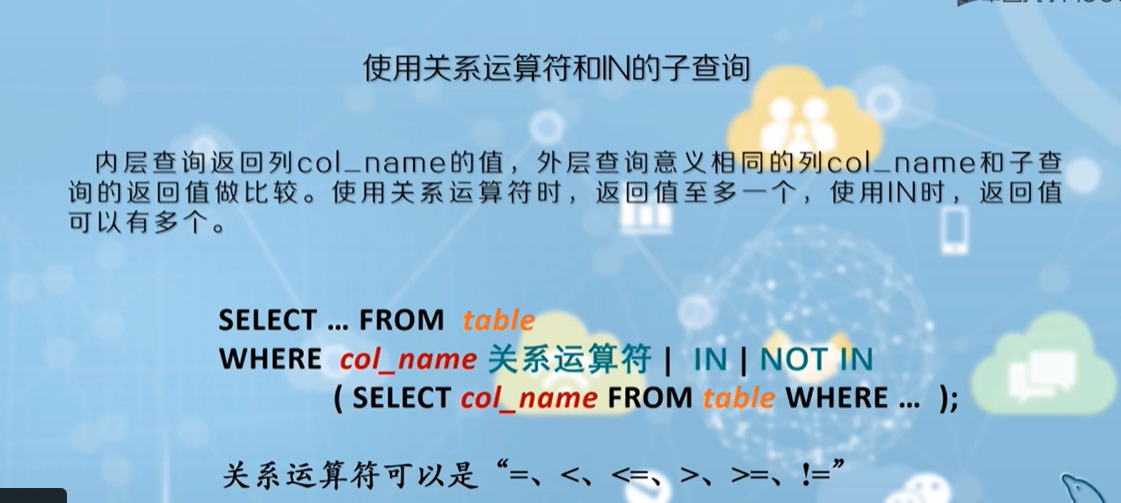
**Select tableA.select\_list, tableB.select\_list from tableA right[outer] join tableB on tableA.key=tableB.key;**

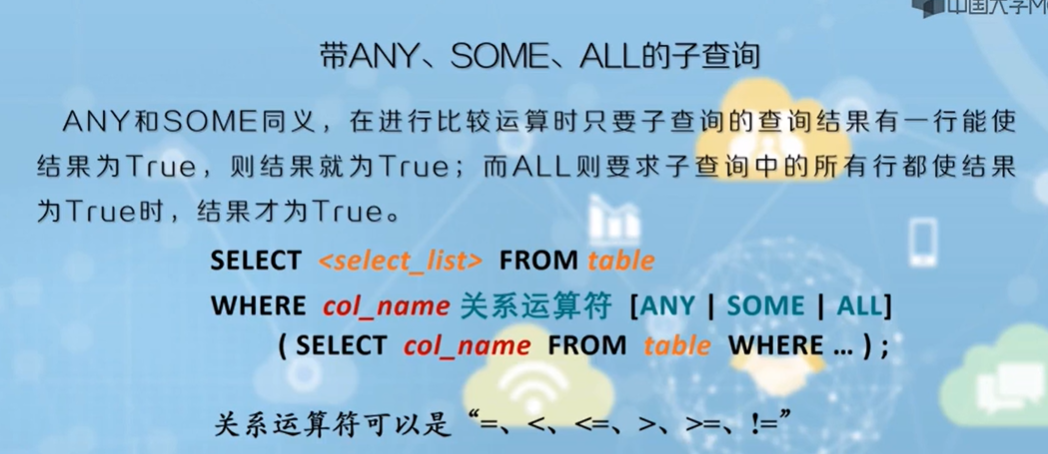
**Union用法：**

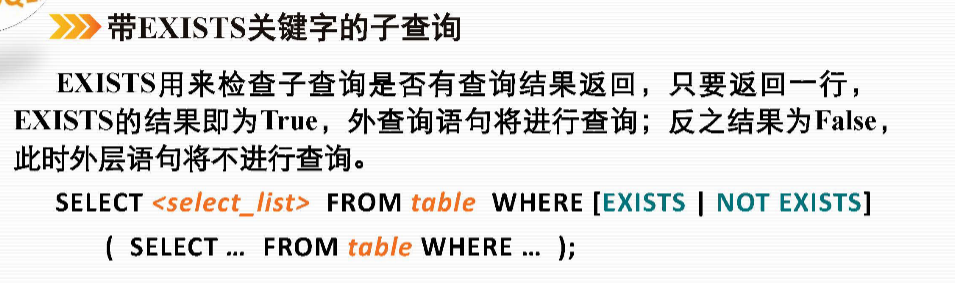
多个select语句要能够进行Union操作必须满足一下条件：一是每个查询语句的字段个数要相同，二是对应字段的类型要相同或者兼容。使用Union语句得到的查询结果得字段名称取第一名select语句得字段名称。默认情况下，Union会把结果集中重复的记录删除；如果想把所有记录都显示出来（包括重复记录）可以再Union之后加上All关键字。这此得重复记录是指每个字段都完全相同的记录。显示重复记录。Union仅显示不重复记录，Union Al 显示包括重复记录的所有记录。

子查询： 子查询也称为嵌套查询(Nested Query)，是嵌套在外层查询where子句中的查询。子查询为主查询返回其所需数据，或者对外查询的查询结果进行一步的限制。

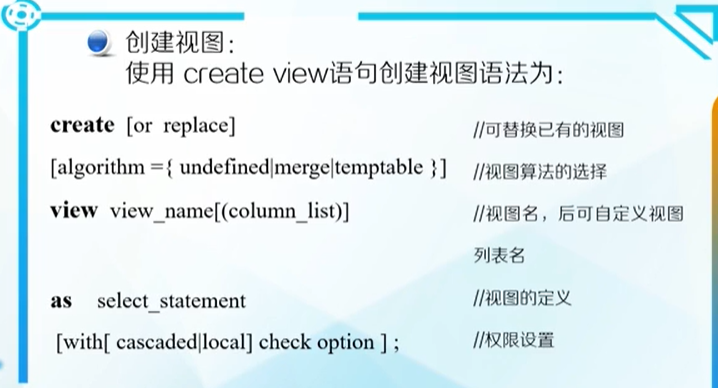
**Select … from table\_name where ( select … from table\_name where …);**











**Algorithm子句：**表示为视图选择算法，有三个选项，一般情况下我们不显示给出，使用第一个参数“undefined”，表示MySQL自动选择算法；

**View\_name：**为视图名。默认情况下，当前数据库创建视图，若想给其他数据创建视图，必须在视图名前制定目标数据库，db\_name.view\_name，视图名不能与表名相同；

**Column\_list**：可以给视图列取名称，多个列用逗号隔开；

**With check option**：用于视图更新时，保证在视图的权限范围内；

要求具有针对视图的create view语句权限，以及针对有select语句选择列上的某些权限；

**Cascaded | local：**可选参数；

**Cascade**：默认值，更新视图时要满足所有相关视图和表结构；

创建视图：

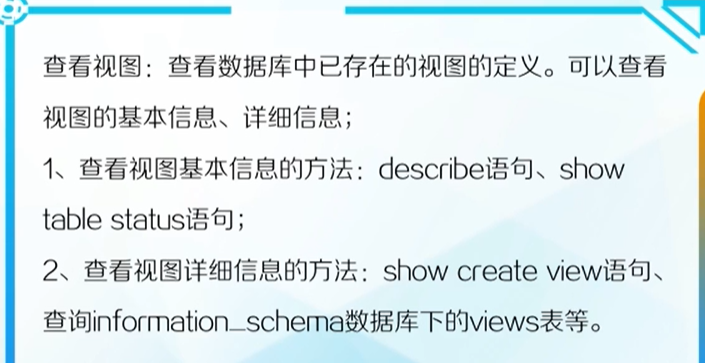
* 视图不仅可以从一个基表导出，还可以从多张基表导出来；
* 并且还可以在已有的视图基础上导出新的视图；
* 因此可以将视图的创建分为三种情况：在表单中创建视图，在多表中创建视图，在已有视图中创建新的视图。

总结：

* 视图是从一个或者多个表、或者其它视图中通过select语句导出的虚拟表；
* 数据库中只存放了视图的定义，并没有存放视图中的数据；
* 浏览视图时产生的数据均来自定义视图查询所引用的基表，并且在引用视图时动态生成；
* 通过视图可以实现对基表数据的查询或修改。

视图作用：

* 简化数据查询和处理：视图可以为用户集中多个表中的数据，使用户可以将注意力集中在他们关心的数据上，简化用户对数据的查询和处理；
* 屏蔽数据库的复杂性：数据库表中的更改不影响用户对数据库的使用，用户也不必了解复杂的数据库中的表的结构。例如，那些定义了若干张表连接的视图，就将表与表之间的连接操作对用户屏蔽起来；
* 安全性：如果想要使用户只能查询或修改用户权限访问的数据，也可以只授予用户访问视图的权限，而不授予访问表的权限，这样就提高了数据库的安全性。



1、：Show table status like ‘view\_name’;

2、：show create view view\_name;

MySQL所有数据库中表的信息都在 information\_schema数据库下。

修改视图方法：

1. 重新创建视图覆盖。
2. Alter [ algorithm={算法}]（中括号可选写区域） view view\_name [column\_list] as select\_statement(查询对象) [with [cascade | local] check option];

视图数据更新语句与表数据更新语句相同。

视图数据更新：

1. 视图若只依赖一张基表，则可以直接通过视图来更新基本表数据；
2. 视图若依赖于多张基表，则一次只能修改一个基表的数据，不能同时修改多个基表的数据。

**注意：包含以下内容时，视图更新操作不会执行**

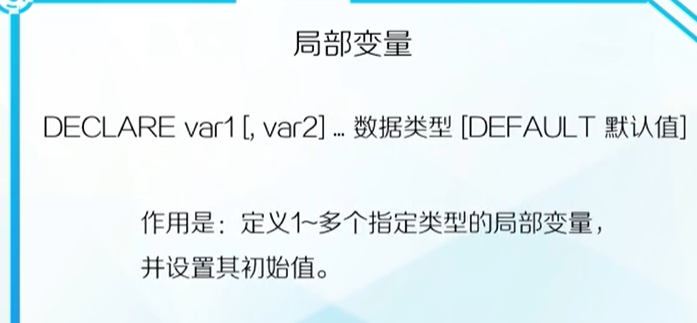
1. **视图中不包含基本表中被定义为非空的列；**
2. **在定义视图的select语句后的字段列表使用了数学表达式；**
3. **在定义视图的select语句后的字段列表中使用了聚合函数；**
4. **在定义视图的select语句引用不可更新视图；**
5. **在定义视图的select语句中使用 distinct、union、top、group by、order by、having字句。**

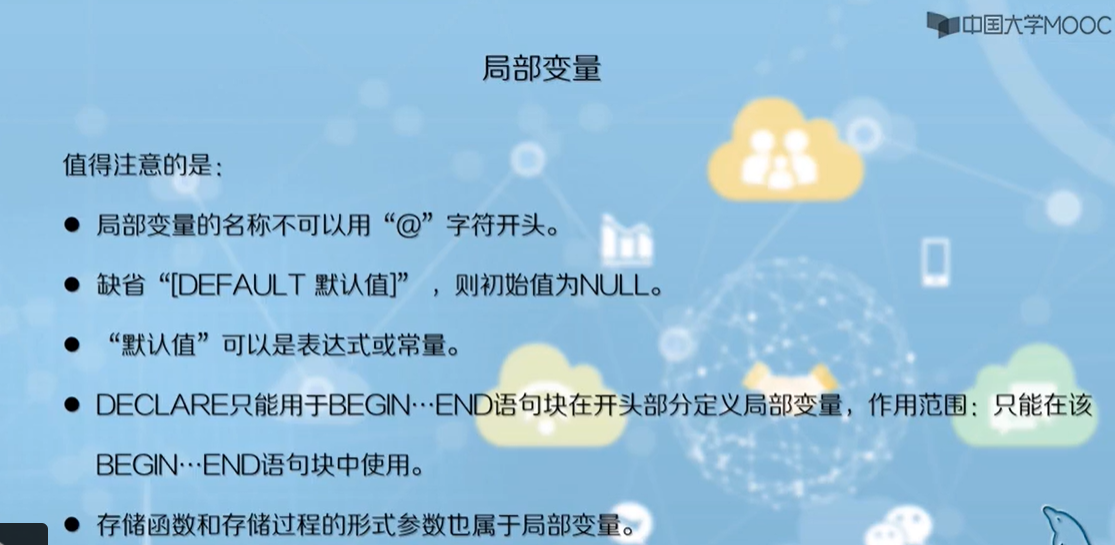
删除视图：必须使用drop权限进行删除，不会删除数据，只删除定义。 Drop view view\_name;

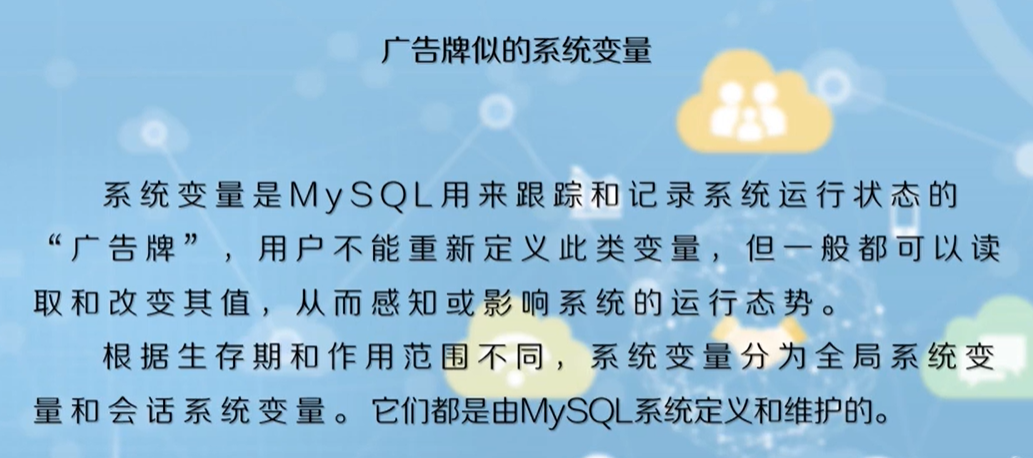
用户变量：根据其使用范围分为会话用户变量和局部变量。

从客户端与mysql服务建立连接后，就可以定义会话用户变量。

在用户断开连接之前，所定义的会话用户变量都一直起作用，但也仅限于本用户的本次连接；连接断开时自动释放会适用用户变量

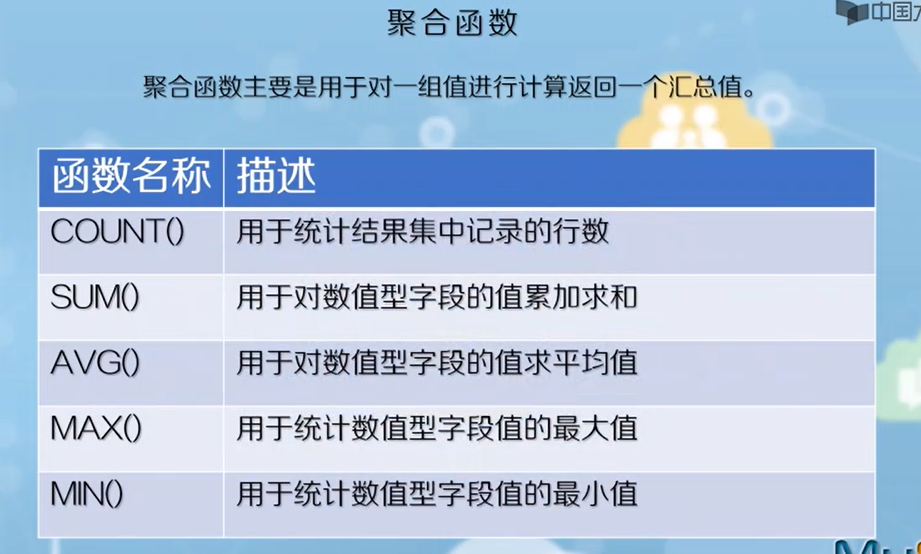


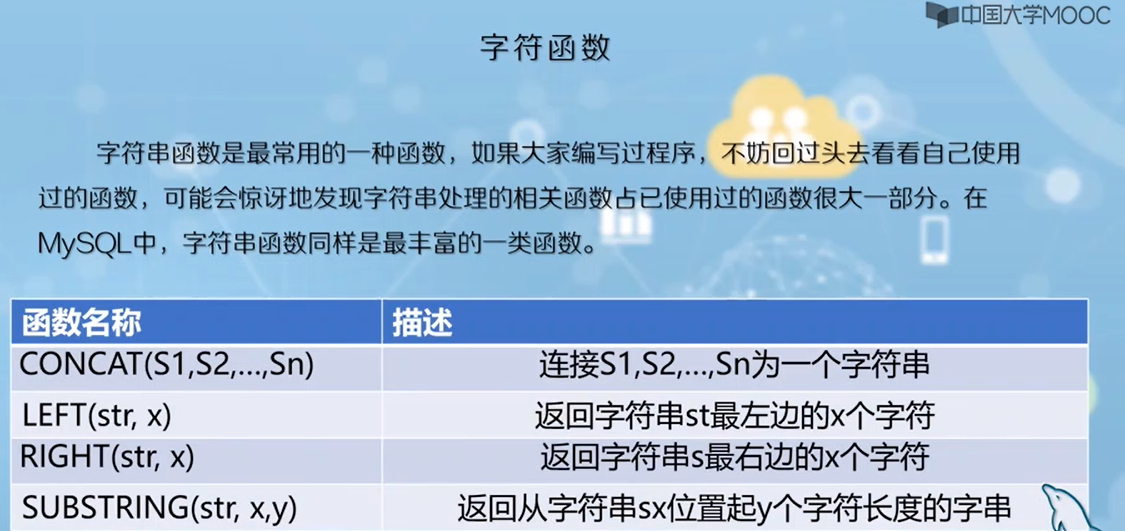




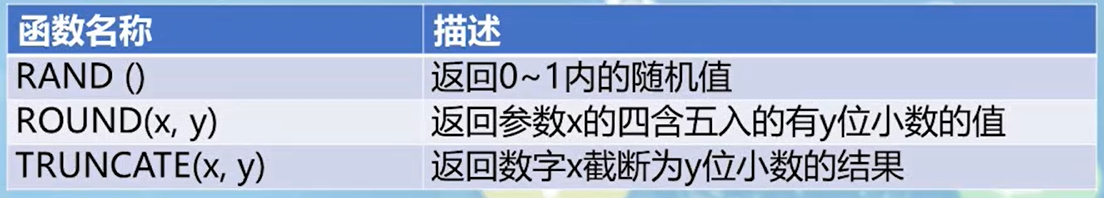
系统变量注意：

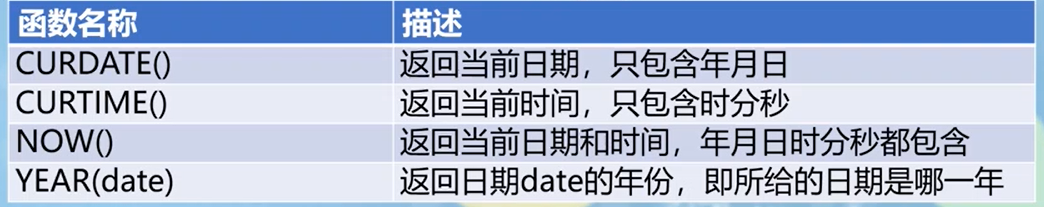
1. 系统变量多数都以2@开头
2. 用户会话变量必须以1个@开头
3. 局部变量则不能以@ 开头

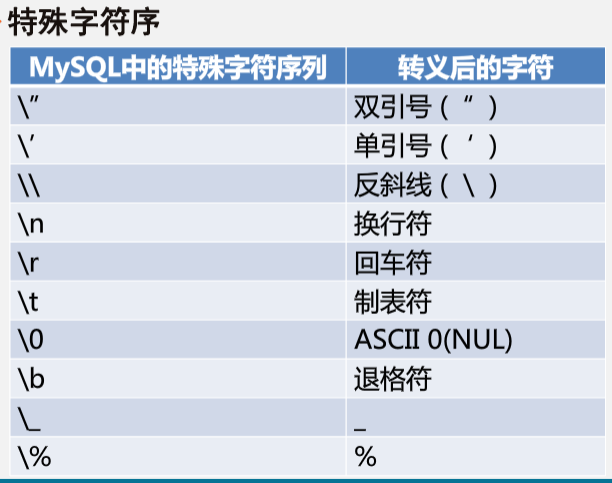




数学函数







存储引擎作用：

1. 设计并创建数据库以保存系统所需的关系或XML文档
2. 实现系统以访问和更改数据库中存储的数据。包括实现网站或使用数据的应用程序，还包括生成使用SQL Server工具和实用工具以使用数据的过程。
3. 为单位或客户部署实现的系统。
4. 提供日常管理支持以优化数据库的性能。

修改数据库引擎：

方式一：修改配置文件my.ini找到default-storage-engine=XXX 修改完成重启服务

方式二：建表指定 create......(…) type=引擎；

方式三：alter table 表名 type=引擎；

查看修改成功： show table status from 表名；show create table 表名；

查看支持存储引擎：show engine；

Mysql各大存储引擎：

InnoDB：是一个事务型的存储引擎，有行级锁定和外键约束。

适用场景：

1. 经常更新的表，适用处理多重并发的更新请求
2. 支持事务。
3. 可以从灾难中恢复（通过bin-log日志等）
4. 外键约束。只有它支持外键。
5. 支持自动增加列属性auto\_increment。

官方解释：

MyIsqm：是MySQL默认的引擎，但是它没有提供对数据库事务的支持，也不支持行级锁和外键，因此当INSERT(插入)或UPDATE(更新)数据时即写操作需要锁定整个表，效率便会低一些。MyIsam 存储引擎独立于操作系统，也就是可以在windows上使用，也可以比较简单的将数据转移到linux操作系统上去。

适用场景：

1. 不支持事务的设计，但并不代表有事务操作的项目不能用MyIsam存储引擎，可以在service层进行根据自己的业务需求进行相应的控制。
2. 不支持外键的表设计
3. 查询速度很快，如果数据库insert和update的操作较多的话表适用
4. 整天对表进行加锁场景。
5. 极度强调快速读取操作。
6. MyIASM中存储了表的行数，于是SELECT COUNT(\*) FROM TABLE时只需要直接读取已经保存好的值而不需要进行全表扫描。如果表的读操作远远多于写操作且不需要数据库事务的支持，那么MyIASM也是很好的选择。

缺点：就是不能在表损坏后恢复数据。（是不能主动恢复）

Memory(也叫HEAP)堆内存：使用存在内存中的内容来创建表。每个MEMORY表只实际对应一个磁盘文件。MEMORY类型的表访问非常得快，因为它的数据是放在内存中的，并且默认使用HASH索引。

适用场景：

1. 那些内容变化不频繁的代码表，或者作为统计操作的中间结果表，便高效地对中间结果进行分析并得到最终的统计结果。
2. 目标数据比较小，而且非常频繁的进行访问，在内存中存放数据，如果太大的数据会造成内存溢出。可以通过参数max\_heap\_table\_size控制Memory表的大小，限制Memory表的最大的大小。
3. 数据是临时的，而且必须立即可用得到，那么就可以放在内存中。
4. 存储在Memory表中的数据如果突然间丢失的话也没有太大的关系。

注意： Memory同时支持散列索引和B树索引，B树索引可以使用部分查询和通配查询，也可以使用<,>和>=等操作符方便数据挖掘，散列索引相等的比较快但是对于范围的比较慢很多。

Mrg\_MyIsam (分表的一种方式-水平分表)：

是一个相同的可以被当作一个来用的MyISAM表的集合。“相同”意味着所有表同样的列和索引信息。

也就是说，他将MyIsam引擎的多个表聚合起来，但是他的内部没有数据，真正的数据依然是MyIsam引擎的表中，但是可以直接进行查询、删除更新等操作。

比如：我们可能会遇到这样的问题，同一种类的数据会根据数据的时间分为多个表，如果这时候进行查询的话，就会比较麻烦，Merge可以直接将多个表聚合成一个表统一查询，然后再删除Merge表（删除的是定义），原来的数据不会影响

Blackhole (黑洞引擎)：

任何写入到此引擎的数据均会被丢弃掉， 不做实际存储；Select语句的内容永远是空。

他会丢弃所有的插入的数据，服务器会记录下Blackhole表的日志，所以可以用于复制数据到备份数据库。

使用场景：

1）验证dump file语法的正确性

2）以使用blackhole引擎来检测binlog功能所需要的额外负载

3）充当日志服务器