# 数据库SqlServer笔试题

9.有哪些操作会使用到TempDB；如果TempDB异常变大，可能的原因是什么，该如何处理；
10.Index有哪些类型，它们的区别和实现原理是什么，索引有啥优点和缺点；如何为SQL语句创建合适的索引，索引创建时有哪些需要,注意的项，如何查看你创建的索引是否被使用；如何维护索引；索引损坏如何检查，怎么修复；T-SQL有更好的索引存在，但是运行,时并没有使用该索引，原因可能是什么；
11.Job信息我们可以通过哪些表获取；系统正在运行的语句可以通过哪些视图获取；如何获取某个T-SQL语句的IO、Time等信息；

# 一、数据库基础知识（通用）篇

## 1.说说主键、外键、超键、候选键

超键：在关系中能唯一标识元组的属性集称为关系模式的超键。一个属性可以为作为一个超键，多个属性组合在一起也可以作为一个超键。超键包含候选键和主键。
候选键：是最小超键，即没有冗余元素的超键。
主键：数据库表中对储存数据对象予以唯一和完整标识的数据列或属性的组合。一个数据列只能有一个主键，且主键的取值不能缺失，即不能为空值（Null）。
外键：在一个表中存在的另一个表的主键称此表的外键。

## 2.为什么用自增列作为主键?

如果我们定义了主键(PRIMARY KEY)，那么InnoDB会选择主键作为聚集索引、
如果没有显式定义主键，则InnoDB会选择第一个不包含有NULL值的唯一索引作为主键索引、
如果也没有这样的唯一索引，则InnoDB会选择内置6字节长的ROWID作为隐含的聚集索引(ROWID随着行记录的写入而主键递增，这个ROWID不像ORACLE的ROWID那样可引用，是隐含的)。
数据记录本身被存于主索引（一颗B+Tree）的叶子节点上。这就要求同一个叶子节点内（大小为一个内存页或磁盘页）的各条数据记录按主键顺序存放，因此每当有一条新的记录插入时，MySQL会根据其主键将其插入适当的节点和位置，如果页面达到装载因子（InnoDB默认为15/16），则开辟一个新的页（节点）
如果表使用自增主键，那么每次插入新的记录，记录就会顺序添加到当前索引节点的后续位置，当一页写满，就会自动开辟一个新的页
如果使用非自增主键（如果身份证号或学号等），由于每次插入主键的值近似于随机，因此每次新记录都要被插到现有索引页的中间某个位置，此时MySQL不得不为了将新记录插到合适位置而移动数据，甚至目标页面可能已经被回写到磁盘上而从缓存中清掉，此时又要从磁盘上读回来，这增加了很多开销，同时频繁的移动、分页操作造成了大量的碎片，得到了不够紧凑的索引结构，后续不得不通过OPTIMIZE TABLE来重建表并优化填充页面。

## 3.触发器的作用是什么?

触发器是一种特殊的存储过程，主要是通过事件来触发而被执行的。它可以强化约束，来维护数据的完整性和一致性，可以跟踪数据库内的操作从而不允许未经许可的更新和变化。可以联级运算。如，某表上的触发器上包含对另一个表的数据操作，而该操作又会导致该表触发器被触发。

## 4.什么是存储过程？用什么来调用?

存储过程是一个预编译的SQL语句，优点是允许模块化的设计，就是说只需创建一次，以后在该程序中就可以调用多次。如果某次操作需要执行多次SQL，使用存储过程比单纯SQL语句执行要快。
调用：
1）可以用一个命令对象来调用存储过程。
2）可以供外部程序调用，比如：java程序。

## 5.说说存储过程的优缺点?

优点:
1）存储过程是预编译过的，执行效率高。
2）存储过程的代码直接存放于数据库中，通过存储过程名直接调用，减少网络通讯。
3）安全性高，执行存储过程需要有一定权限的用户。
4）存储过程可以重复使用，可减少数据库开发人员的工作量。
缺点:
移植性差

## 6.说说存储过程与函数的区别

（1）存储过程用户在数据库中完成特定操作或者任务（如插入，删除等），函数用于返回特定的数据。
（2）存储过程声明用procedure，函数用function。
（3）存储过程不需要返回类型，函数必须要返回类型。
（4）存储过程可作为独立的pl-sql执行，函数不能作为独立的plsql执行，必须作为表达式的一部分。
（5）存储过程只能通过out和in/out来返回值，函数除了可以使用out，in/out以外，还可以使用return返回值。
（6）sql语句（DML或SELECT)中不可用调用存储过程，而函数可以。

## 7.什么叫视图？游标是什么?

视图:
是一种虚拟的表，具有和物理表相同的功能。可以对视图进行增，改，查，操作，试图通常是有一个表或者多个表的行或列的子集。对视图的修改会影响基本表。它使得我们获取数据更容易，相比多表查询。
游标:
是对查询出来的结果集作为一个单元来有效的处理。游标可以定在该单元中的特定行，从结果集的当前行检索一行或多行。可以对结果集当前行做修改。一般不使用游标，但是需要逐条处理数据的时候，游标显得十分重要。

## 8.视图的优缺点有哪些?

优点:
1对数据库的访问，因为视图可以有选择性的选取数据库里的一部分。
2)用户通过简单的查询可以从复杂查询中得到结果。
3)维护数据的独立性，试图可从多个表检索数据。
4)对于相同的数据可产生不同的视图。
缺点:
性能：查询视图时，必须把视图的查询转化成对基本表的查询，如果这个视图是由一个复杂的多表查询所定义，那么，那么就无法更改数据

## 9.说说drop、truncate、 delete区别

最基本:
1) drop直接删掉表。
2) truncate删除表中数据，再插入时自增长id又从1开始。
3) delete删除表中数据，可以加where字句。

（1） DELETE语句执行删除的过程是每次从表中删除一行，并且同时将该行的删除操作作为事务记录在日志中保存以便进行进行回滚操作。TRUNCATE TABLE 则一次性地从表中删除所有的数据并不把单独的删除操作记录记入日志保存，删除行是不能恢复的。并且在删除的过程中不会激活与表有关的删除触发器。执行速度快。

（2） 表和索引所占空间。当表被TRUNCATE 后，这个表和索引所占用的空间会恢复到初始大小，而DELETE操作不会减少表或索引所占用的空间。drop语句将表所占用的空间全释放掉。
（3） 一般而言，drop > truncate > delete
（4） 应用范围。TRUNCATE 只能对TABLE；DELETE可以是table和view
（5） TRUNCATE 和DELETE只删除数据，而DROP则删除整个表（结构和数据）。
（6） truncate与不带where的delete ： 只删除数据，而不删除表的结构（定义）drop语句将删除表的结构被依赖的约束（constrain),触发器（trigger)索引（index);依赖于该表的存储过程/函数将被保留，但其状态会变为：invalid。
（7） delete语句为DML（data maintain Language),这个操作会被放到 rollback segment中,事务提交后才生效。如果有相应的 tigger,执行的时候将被触发。
（8） truncate、drop是DLL（data define language),操作立即生效，原数据不放到 rollback segment中，不能回滚。
（9） 在没有备份情况下，谨慎使用 drop 与 truncate。要删除部分数据行采用delete且注意结合where来约束影响范围。回滚段要足够大。要删除表用drop;若想保留表而将表中数据删除，如果与事务无关，用truncate即可实现。如果和事务有关，或老师想触发trigger,还是用delete。
（10） Truncate table 表名 速度快,而且效率高,因为:?truncate table 在功能上与不带 WHERE 子句的 DELETE 语句相同：二者均删除表中的全部行。但 TRUNCATE TABLE 比 DELETE 速度快，且使用的系统和事务日志资源少。DELETE 语句每次删除一行，并在事务日志中为所删除的每行记录一项。TRUNCATE TABLE 通过释放存储表数据所用的数据页来删除数据，并且只在事务日志中记录页的释放。
（11） TRUNCATE TABLE 删除表中的所有行，但表结构及其列、约束、索引等保持不变。新行标识所用的计数值重置为该列的种子。如果想保留标识计数值，请改用 DELETE。如果要删除表定义及其数据，请使用 DROP TABLE 语句。
（12） 对于由 FOREIGN KEY 约束引用的表，不能使用 TRUNCATE TABLE，而应使用不带 WHERE 子句的 DELETE 语句。由于 TRUNCATE TABLE 不记录在日志中，所以它不能激活触发器。

## 10.什么是临时表，临时表什么时候删除?

临时表可以手动删除：
DROP TEMPORARY TABLE IF EXISTS temp_tb;
临时表只在当前连接可见，当关闭连接时，MySQL会自动删除表并释放所有空间。因此在不同的连接中可以创建同名的临时表，并且操作属于本连接的临时表。
创建临时表的语法与创建表语法类似，不同之处是增加关键字TEMPORARY，
如：

```
CREATE TEMPORARY TABLE tmp_table (
        NAME VARCHAR (10) NOT NULL,
        time date NOT NULL
);
select * from tmp_table;
```

## 11.说说非关系型数据库和关系型数据库区别，优势比较?

非关系型数据库的优势：
性能：NOSQL是基于键值对的，可以想象成表中的主键和值的对应关系，而且不需要经过SQL层的解析，所以性能非常高。
可扩展性：同样也是因为基于键值对，数据之间没有耦合性，所以非常容易水平扩展。
关系型数据库的优势：
复杂查询：可以用SQL语句方便的在一个表以及多个表之间做非常复杂的数据查询。

事务支持：使得对于安全性能很高的数据访问要求得以实现。
其他：
1.对于这两类数据库，对方的优势就是自己的弱势，反之亦然。
2.NOSQL数据库慢慢开始具备SQL数据库的一些复杂查询功能，比如MongoDB。
3.对于事务的支持也可以用一些系统级的原子操作来实现例如乐观锁之类的方法来曲线救国，比如Redis set nx。

## 12.什么是数据库范式，根据某个场景设计数据表?

第一范式:(确保每列保持原子性)所有字段值都是不可分解的原子值。
第一范式是最基本的范式。如果数据库表中的所有字段值都是不可分解的原子值，就说明该数据库表满足了第一范式。
第一范式的合理遵循需要根据系统的实际需求来定。比如某些数据库系统中需要用到“地址”这个属性，本来直接将“地址”属性设计成一个数据库表的字段就行。但是如果系统经常会访问“地址”属性中的“城市”部分，那么就非要将“地址”这个属性重新拆分为省份、城市、详细地址等多个部分进行存储，这样在对地址中某一部分操作的时候将非常方便。这样设计才算满足了数据库的第一范式，如下表所示。
上表所示的用户信息遵循了第一范式的要求，这样在对用户使用城市进行分类的时候就非常方便，也提高了数据库的性能。
第二范式:(确保表中的每列都和主键相关)在一个数据库表中，一个表中只能保存一种数据，不可以把多种数据保存在同一张数据库表中。
第二范式在第一范式的基础之上更进一层。第二范式需要确保数据库表中的每一列都和主键相关，而不能只与主键的某一部分相关（主要针对联合主键而言）。也就是说在一个数据库表中，一个表中只能保存一种数据，不可以把多种数据保存在同一张数据库表中。
比如要设计一个订单信息表，因为订单中可能会有多种商品，所以要将订单编号和商品编号作为数据库表的联合主键。
第三范式:(确保每列都和主键列直接相关,而不是间接相关) 数据表中的每一列数据都和主键直接相关，而不能间接相关。
第三范式需要确保数据表中的每一列数据都和主键直接相关，而不能间接相关。
比如在设计一个订单数据表的时候，可以将客户编号作为一个外键和订单表建立相应的关系。而不可以在订单表中添加关于客户其它信息（比如姓名、所属公司等）的字段。
BCNF:符合3NF，并且，主属性不依赖于主属性。
若关系模式属于第二范式，且每个属性都不传递依赖于键码，则R属于BC范式。
通常BC范式的条件有多种等价的表述：每个非平凡依赖的左边必须包含键码；每个决定因素必须包含键码。
BC范式既检查非主属性，又检查主属性。当只检查非主属性时，就成了第三范式。满足BC范式的关系都必然满足第三范式。
还可以这么说：若一个关系达到了第三范式，并且它只有一个候选码，或者它的每个候选码都是单属性，则该关系自然达到BC范式。
一般，一个数据库设计符合3NF或BCNF就可以了。
第四范式:要求把同一表内的多对多关系删除。
第五范式:从最终结构重新建立原始结构。

## 13.什么是 内连接、外连接、交叉连接、笛卡尔积等?

内连接: 只连接匹配的行
左外连接: 包含左边表的全部行（不管右边的表中是否存在与它们匹配的行），以及右边表中全部匹配的行
右外连接: 包含右边表的全部行（不管左边的表中是否存在与它们匹配的行），以及左边表中全部匹配

的行
例如1：

```
SELECT a.,b. FROM luntan LEFT JOIN usertable as b ON a.username=b.username
```

例如2：

```
SELECT a.,b. FROM city as a FULL OUTER JOIN user as b ON a.username=b.username
```

全外连接: 包含左、右两个表的全部行，不管另外一边的表中是否存在与它们匹配的行。
交叉连接: 生成笛卡尔积 - 它不使用任何匹配或者选取条件，而是直接将一个数据源中的每个行与另一个数据源的每个行都一一匹配
例如：

```
SELECT type,pub_name FROM titles CROSS JOIN publishers ORDER BY type
```

## 14.varchar和char的使用场景?

1.char的长度是不可变的，而varchar的长度是可变的。
定义一个char[10]和varchar[10]。
如果存进去的是'csdn',那么char所占的长度依然为10，除了字符'csdn'外，后面跟六个空格，varchar就立马把长度变为4了，取数据的时候，char类型的要用trim()去掉多余的空格，而varchar是不需要的。
2.char的存取速度还是要比varchar要快得多，因为其长度固定，方便程序的存储与查找。
char也为此付出的是空间的代价，因为其长度固定，所以难免会有多余的空格占位符占据空间，可谓是以空间换取时间效率。
varchar是以空间效率为首位。
3.char的存储方式是：对英文字符（ASCII）占用1个字节，对一个汉字占用两个字节。
varchar的存储方式是：对每个英文字符占用2个字节，汉字也占用2个字节。
4.两者的存储数据都非unicode的字符数据。

## 15.SQL语言分类

SQL语言共分为四大类：
一、数据查询语言DQL
二、数据操纵语言DML
三、数据定义语言DDL
四、数据控制语言DCL。

1. 数据查询语言DQL
   数据查询语言DQL基本结构是由SELECT子句，FROM子句，WHERE子句组成的查询块：
   SELECT
   FROM
   WHERE
2. 数据操纵语言DML
   数据操纵语言DML主要有三种形式：
   1) 插入：INSERT
   2) 更新：UPDATE
   3) 删除：DELETE

3. 数据定义语言DDL
   数据定义语言DDL用来创建数据库中的各种对象-----表、视图、索引、同义词、聚簇等如：
   CREATE TABLE/VIEW/INDEX/SYN/CLUSTER
   表 视图 索引 同义词 簇
   DDL操作是隐性提交的！不能rollback
4. 数据控制语言DCL
   数据控制语言DCL用来授予或回收访问数据库的某种特权，并控制数据库操纵事务发生的时间及效果，对数据库实行监视等。如：
   1) GRANT：授权。
   2) ROLLBACK [WORK] TO [SAVEPOINT]：回退到某一点。回滚---ROLLBACK；回滚命令使数据库状态回到上次最后提交的状态。其格式为：
   SQL>ROLLBACK;
   3) COMMIT [WORK]：提交。
   在数据库的插入、删除和修改操作时，只有当事务在提交到数据库时才算完成。在事务提交前，只有操作数据库的这个人才能有权看到所做的事情，别人只有在最后提交完成后才可以看到。
   提交数据有三种类型：显式提交、隐式提交及自动提交。下面分别说明这三种类型。
   (1) 显式提交
   用COMMIT命令直接完成的提交为显式提交。其格式为：SQL>COMMIT；
   (2) 隐式提交
   用SQL命令间接完成的提交为隐式提交。这些命令是：
   ALTER，AUDIT，COMMENT，CONNECT，CREATE，DISCONNECT，DROP，
   EXIT，GRANT，NOAUDIT，QUIT，REVOKE，RENAME。
   (3) 自动提交
   若把AUTOCOMMIT设置为ON，则在插入、修改、删除语句执行后，
   系统将自动进行提交，这就是自动提交。
   其格式为：SQL>SET AUTOCOMMIT ON；

## 16.说说like %和-的区别

通配符的分类
%百分号通配符:表示任何字符出现任意次数(可以是0次).
*下划线通配符:表示只能匹配单个字符,不能多也不能少,就是一个字符.*
*like操作符: LIKE作用是指示mysql后面的搜索模式是利用通配符而不是直接相等匹配进行比较.*
*注意: 如果在使用like操作符时,后面的没有使用通用匹配符效果是和=一致的,SELECT * FROM products WHERE products.prod_name like '1000';只能匹配的结果为1000,而不能匹配像JetPack 1000这样的结果.*
*%通配符使用: 匹配以"yves"开头的记录:(包括记录"yves") SELECT FROM products WHERE products.prod_name like 'yves%';*
*匹配包含"yves"的记录(包括记录"yves") SELECT FROM products WHERE products.prod_name like '%yves%';*
*匹配以"yves"结尾的记录(包括记录"yves",不包括记录"yves ",也就是yves后面有空格的记录,这里需要注意) SELECT * FROM products WHERE products.prod_name like '%yves';*
*通配符使用: SELECT FROM products WHERE products.prod_name like* 'yves'; 匹配结果为: 像"yyves"这样记录. SELECT FROM products WHERE products.prodname like 'yves'; 匹配结果为: 像"yvesHe"这样的记录.(一个下划线只能匹配一个字符,不能多也不能少)
注意事项:
注意大小写,在使用模糊匹配时,也就是匹配文本时,mysql是可能区分大小的,也可能是不区分大小写的,这个结果是取决于用户对MySQL的配置方式.如果是区分大小写,那么像YvesHe这样记录是不能被"yves__"这样的匹配条件匹配的.
注意尾部空格,"%yves"是不能匹配"heyves "这样的记录的.
注意NULL,%通配符可以匹配任意字符,但是不能匹配NULL,也就是说SELECT * FROM products WHERE products.prod_name like '%;是匹配不到products.prod_name为NULL的的记录.

技巧与建议:
正如所见， MySQL的通配符很有用。但这种功能是有代价的：通配符搜索的处理一般要比前面讨论的其他搜索所花时间更长。这里给出一些使用通配符要记住的技巧。
不要过度使用通配符。如果其他操作符能达到相同的目的，应该 使用其他操作符。
在确实需要使用通配符时，除非绝对有必要，否则不要把它们用 在搜索模式的开始处。把通配符置于搜索模式的开始处，搜索起 来是最慢的。
仔细注意通配符的位置。如果放错地方，可能不会返回想要的数.

## 17.说说count(*)、count(1)、count(column)的区别

count()*对行的数目进行计算,包含NULL*
*count(column)对特定的列的值具有的行数进行计算,不包含NULL值。*
*count()还有一种使用方式,count(1)这个用法和count()*的结果是一样的。
性能问题:
1.任何情况下SELECT COUNT() *FROM tablename是最优选择,*
*2.尽量减少SELECT COUNT()* FROM tablename WHERE COL = ‘value’ 这种查询;
3.杜绝SELECT COUNT(COL) FROM tablename WHERE COL2 = ‘value’ 的出现。
如果表没有主键,那么count(1)比count()*快。*
*如果有主键,那么count(主键,联合主键)比count()*快。
如果表只有一个字段,count()*最快。*
*count(1)跟count(主键)一样,只扫描主键。count()*跟count(非主键)一样,扫描整个表。明显前者更快一些。

## 18.什么是最左前缀原则?

多列索引:

```
ALTER TABLE people ADD INDEX lname_fname_age (lame,fname,age);
```

为了提高搜索效率，我们需要考虑运用多列索引,由于索引文件以B - Tree格式保存，所以我们不用扫描任何记录，即可得到最终结果。
注：在mysql中执行查询时，只能使用一个索引，如果我们在lname,fname,age上分别建索引,执行查询时，只能使用一个索引，mysql会选择一个最严格(获得结果集记录数最少)的索引。
最左前缀原则：顾名思义，就是最左优先，上例中我们创建了lname_fname_age多列索引,相当于创建了(lname)单列索引，(lname,fname)组合索引以及(lname,fname,age)组合索引。

## 19.什么是索引?

何为索引:
数据库索引，是数据库管理系统中一个排序的数据结构，索引的实现通常使用B树及其变种B+树。
在数据之外，数据库系统还维护着满足特定查找算法的数据结构，这些数据结构以某种方式引用（指向）数据，这样就可以在这些数据结构上实现高级查找算法。这种数据结构，就是索引。

## 20.索引的作用？它的优点缺点是什么?

索引作用:
协助快速查询、更新数据库表中数据。
为表设置索引要付出代价的:
一是增加了数据库的存储空间
二是在插入和修改数据时要花费较多的时间(因为索引也要随之变动)。

## 21.索引的优缺点有哪些?

创建索引可以大大提高系统的性能（优点）：
（1）通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。
（2）可以大大加快数据的检索速度，这也是创建索引的最主要的原因。
（3）可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。
（4）在使用分组和排序子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。
（5）通过使用索引，可以在查询的过程中，使用优化隐藏器，提高系统的性能。
增加索引也有许多不利的方面(缺点)：
（1）.创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。
（2）.索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间，如果要建立聚簇索引，那么需要的空间就会更大。
（3）.当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就降低了数据的维护速度。
（4）.哪些列适合建立索引、哪些不适合建索引?
索引是建立在数据库表中的某些列的上面。在创建索引的时候，应该考虑在哪些列上可以创建索引，在哪些列上不能创建索引。
一般来说，应该在这些列上创建索引：
（1）在经常需要搜索的列上，可以加快搜索的速度；
（2）在作为主键的列上，强制该列的唯一性和组织表中数据的排列结构；
（3）在经常用在连接的列上，这些列主要是一些外键，可以加快连接的速度；
（4）在经常需要根据范围进行搜索的列上创建索引，因为索引已经排序，其指定的范围是连续的；
（5）在经常需要排序的列上创建索引，因为索引已经排序，这样查询可以利用索引的排序，加快排序查询时间；
（6）在经常使用在WHERE子句中的列上面创建索引，加快条件的判断速度。

对于有些列不应该创建索引：
（1）对于那些在查询中很少使用或者参考的列不应该创建索引。
这是因为，既然这些列很少使用到，因此有索引或者无索引，并不能提高查询速度。相反，由于增加了索引，反而降低了系统的维护速度和增大了空间需求。
（2）对于那些只有很少数据值的列也不应该增加索引。
这是因为，由于这些列的取值很少，例如人事表的性别列，在查询的结果中，结果集的数据行占了表中数据行的很大比例，即需要在表中搜索的数据行的比例很大。增加索引，并不能明显加快检索速度。
（3）对于那些定义为text, image和bit数据类型的列不应该增加索引。
这是因为，这些列的数据量要么相当大，要么取值很少。
(4)当修改性能远远大于检索性能时，不应该创建索引。
这是因为，修改性能和检索性能是互相矛盾的。当增加索引时，会提高检索性能，但是会降低修改性能。当减少索引时，会提高修改性能，降低检索性能。因此，当修改性能远远大于检索性能时，不应该创建索引。
索引详解：带你从头到尾捋一遍MySQL索引结构！

## 22.什么样的字段适合建索引?

唯一、不为空、经常被查询的字段

## 23.说说聚集索引和非聚集索引区别?

聚合索引(clustered index):
聚集索引表记录的排列顺序和索引的排列顺序一致，所以查询效率快，只要找到第一个索引值记录，其余就连续性的记录在物理也一样连续存放。聚集索引对应的缺点就是修改慢，因为为了保证表中记录的物理和索引顺序一致，在记录插入的时候，会对数据页重新排序。

聚集索引类似于新华字典中用拼音去查找汉字，拼音检索表于书记顺序都是按照a~z排列的，就像相同的逻辑顺序于物理顺序一样，当你需要查找a,ai两个读音的字，或是想一次寻找多个傻(sha)的同音字时，也许向后翻几页，或紧接着下一行就得到结果了。
非聚合索引(nonclustered index):
非聚集索引指定了表中记录的逻辑顺序，但是记录的物理和索引不一定一致，两种索引都采用B+树结构，非聚集索引的叶子层并不和实际数据页相重叠，而采用叶子层包含一个指向表中的记录在数据页中的指针方式。非聚集索引层次多，不会造成数据重排。
非聚集索引类似在新华字典上通过偏旁部首来查询汉字，检索表也许是按照横、竖、撇来排列的，但是由于正文中是a~z的拼音顺序，所以就类似于逻辑地址于物理地址的不对应。同时适用的情况就在于分组，大数目的不同值，频繁更新的列中，这些情况即不适合聚集索引。
根本区别：
聚集索引和非聚集索引的根本区别是表记录的排列顺序和与索引的排列顺序是否一致。

# 二、SqlServer笔试基础篇

试用SQL查询语句表达下列对教学数据库中三个基本表 S、SC 、C 的查询：
S(sno,sname,SAGE,SSEX) 各字段表示学号，姓名，年龄，性别
Sc(sno,cno,grade) 各字段表示学号，课程号，成绩、
C(cno,cname, TEACHER) 各字段表示课程号，课程名和教师名 其 中 SAGE， grade 是数值型，其他均为字符型。

## 1．求年龄大于所有女同学年龄的男学生姓名和年龄。

```
SELECTSNAME,SAGE FROM S AS X
WHERE X.SSEX=' 男'AND X.SAGE >ALL (SELECT SAGE FROMS AS Y WHERE
Y.SSEX=' 女')
```

## 2.求年龄大于女同学平均年龄的男学生姓名和年龄。

```
SELECT SNAME,SAGE
FROM S
WHERE SSEX=' 男'
AND SAGE>(SELECTAVG(SAGE) FROM S WHERE SSEX='女')
```

## 3.在 SC 中检索成绩为空值的学生学号和课程号。

```
SELECT Sno,CnoFROM SC WHERE GRADE IS NULL
```

## 4.检索姓名以 WANG 打头的所有学生的姓名和年龄。

```
SELECT SNAME,SAGE FROM S
WHERE SNAME LIKE 'WANG%'
```

## 5.检索学号比 WANG 同学大，而年龄比他小的学生姓名。

```
SELECTX.SNAME FROM S AS X, S AS Y
WHERE Y .SNAME='WANG' AND X.Sno>Y.Sno AND X.SAGE
```

```
SELECT SNAME
from s
where sno>(select sno from s where SNAME='WANG') andSAGE<(select sAGE from s
where SNAME='WANG')
```

## 6.统计每门课程的学生选修人数 （超过 2 人的课程才统计）。要求输出课程号和选修人数，查询结果按人数降序排列，若人数相同，按课程号升序排列。

```
SELECTDISTINCT Cno,COUNT(Sno) FROM SC
GROUPBY Cno HAVING COUNT(Sno)>2
ORDER BY 2 DESC, Cno ASC
```

```
SELECT DISTINCT Cno,COUNT(Sno) as 人数
FROMSC GROUP BY Cno
HAVING COUNT(Sno)>2
ORDER BY 人 数 DESC, Cno ASC
```

## 7.求 LIU 老师所授课程的每门课程的学生平均成绩。

```
SELECT AVG(GRADE)
FROM SC join C on SC.Cno=C.Cno WHERE TEACHER='liu'
GROUP BY c.Cno
```

```
SELECTCNAME,AVG(GRADE) FROM SC ,C WHERE SC.Cno=C.Cno AND TEACHER='liu'
GROUP BY c.Cno,cname
```

## 8.求选修 C4 课程的学生的平均年龄。

```
SELECT AVG(SAGE )
FROMS WHERE Sno
IN(SELECT Sno FROM SC WHERE Cno='4')
```

```
SELECT AVG(SAGE)
FROM S,SC WHERES.Sno=SC.Sno AND Cno='4'
```

## 9．统计有学生选修的课程门数。

```
SELECT COUNT(DISTINCT Cno) FROM SC
```

试用 SQL 更新语句表达对教学数据库中三个基本表 S、
SC 、C的各个更新操作:

## 10.在基本表 SC 中修改 4 号课程的成绩，若成绩小于等于 75 分时提高 5%， 若成绩大于 75 分时提高 4% （用两个 UPDATE 语句实现）。

```
UPDATE SC SETGRADE=GRADE*1.05 WHERE Cno='4' AND GRADE<=75
```

```
UPDATE SC SET GRADE=GRADE*1.04 WHERE Cno='4' AND GRADE>75
```

### 11 .把低于总平均成绩的女同学成绩提高 5% 。

```
UPDATE SC SETGRADE=GRADE*1.05 WHERE GRADE<(SELECT AVG(GRADE) FROM SC)
AND Sno IN (SELECT Sno FROM SWHERE SSEX=' 女')
```

### 12 .把选修数据库原理课不及格的成绩全改为空值。

```
UPDATE SC SET GRADE=NULL
WHERE GRADE<60 AND Cno IN(SELECT Cno FROM C
WHERE CNAME=' 数据库原理 ')
```

### 13．把WANG 同学的学习选课和成绩全部删去。

```
DELETEFROM SC WHERE Sno IN(SELECT Sno FROM S
WHERE SNAME='WANG')
```

### 14 ． 在基本表 SC 中删除尚无成绩的选课元组。

```
DELETE FROM SCWHERE GRADE IS NULL
```

### 15 ． 往基本表 S 中插入一个学生元组（‘S9’，‘WU’，18）。。

```
INSERT INTO S(Sno,SNAME,SAGE) VALUES('59','WU',18)
```

### 16.什么是SQL注入式攻击?

所谓SQL注入式攻击，就是攻击者把SQL命令插入到Web表单的输入域或页面请求的查询字符串，欺骗服务器执行恶意的SQL命令。在某些表单中，用户输入的内容直接用来构造（或者影响）动态SQL命令，或作为存储过程的输入参数，这类表单特别容易受到SQL注入式攻击。

## 三、SqlServer笔试高级篇

### 1.什么是内存泄漏?

一般我们所说的内存泄漏指的是堆内存的泄漏。堆内存是程序从堆中为其分配的，大小任意的，使用完后要显示释放内存。当应用程序用关键字new 等创建对象时，就从堆中为它分配一块内存，使用完后程序调用free 或者delete 释放该内存，否则就说该内存就不能被使用，我们就说该内存被泄漏了。

### 2.维护数据库的完整性和一致性，你喜欢用触发器还是自写业务逻辑？为什么?

是这样做的，尽可能使用约束，如check, 主键，外键，非空字段等来约束，这样做效率最高，也最方便。其次是使用触发器，这种方法可以保证，无论什么业务系统访问数据库都可以保证数据的完整新和一致性。最后考虑的是自写业务逻辑，但这样做麻烦，编程复杂，效率低下。

## 3.什么是事务？什么是锁?

事务就是被绑定在一起作为一个逻辑工作单元的SQL 语句分组，如果任何一个语句操作失败那么整个操作就被失败，以后操作就会回滚到操作前状态，或者是上有个节点。为了确保要么执行，要么不执行，就可以使用事务。要将有组语句作为事务考虑，就需要通过ACID 测试，即原子性，一致性，隔离性和持久性。

锁：在所以的 DBMS中，锁是实现事务的关键，锁可以保证事务的完整性和并发性。与现实生活中锁一样，它可以使某些数据的拥有者，在某段时间内不能使用某些数据或数据结构。当然锁还分级别的。

## 4.对一个投入使用的在线事务处理表格有过多索引需要有什么样的性能考虑?

对一个表格的索引越多，数据库引擎用来更新、插入或者删除数据所需要的时间就越多，因为在数据操控发生的时候索引也必须要维护。

5.什么是相关子查询?如何使用这些查询?

相关子查询是一种包含子查询的特殊类型的查询。查询里包含的子查询会真正请求外部查询的值，从而形成一个类似于循环的状况。

## 6.什么是SQL注入式攻击?

就是攻击者把SQL命令插入到Web表单的输入域或页面请求的查询字符串，欺骗服务器执行恶意的SQL命令。在某些表单中，用户输入的内容直接用来构造（或者影响）动态SQL命令，或作为存储过程的输入参数，这类表单特别容易受到SQL注入式攻击

## 7.如何防范SQL注入式攻击?

好在要防止ASP.NET应用被SQL注入式攻击闯入并不是一件特别困难的事情，只要在利用表单输入的内容构造SQL命令之前，把所有输入内容过滤一番就可以了。过滤输入内容可以按多种方式进行。

(1) 对于动态构造SQL查询的场合，可以使用下面的技术：

第一：替换单引号，即把所有单独出现的单引号改成两个单引号，防止攻击者修改SQL命令的含义。再来看前面的例子，“SELECT * from Users WHERE login = ‘’ or ”1”=”1’ AND password = ‘’ or ”1”=”1’’ 显然会得到与“SELECT * from Users WHERE login = ’’ or ’1’=’1’ AND password = ’’ or ’1’=’1’’不同的结果。

第二：删除用户输入内容中的所有连字符，防止攻击者构造出类如“SELECT * from Users WHERE login = ’mas’ —— AND password =””之类的查询，因为这类查询的后半部分已经被注释掉，不再有效，攻击者只要知道一个合法的用户登录名称，根本不需要知道用户的密码就可以顺利获得访问权限。

第三：对于用来执行查询的数据库帐户，限制其权限。用不同的用户帐户执行查询、插入、更新、删除操作。由于隔离了不同帐户可执行的操作，因而也就防止了原本用于执行SELECT命令的地方却被用于执行INSERT、UPDATE或DELETE命令。

(2) 用存储过程来执行所有的查询。SQL参数的传递方式将防止攻击者利用单引号和连字符实施攻击。此外，它还使得数据库权限可以限制到只允许特定的存储过程执行，所有的用户输入必须遵从被调用的存储过程的安全上下文，这样就很难再发生注入式攻击了。

(3) 限制表单或查询字符串输入的长度。如果用户的登录名字最多只有10个字符，那么不要认可表单中输入的10个以上的字符，这将大大增加攻击者在SQL命令中插入有害代码的难度。

(4) 检查用户输入的合法性，确信输入的内容只包含合法的数据。数据检查应当在客户端和服务器端都执行——之所以要执行服务器端验证，是为了弥补客户端验证机制脆弱的安全性。

在客户端，攻击者完全有可能获得网页的源代码，修改验证合法性的脚本(或者直接删除脚本)，然后将非法内容通过修改后的表单提交给服务器。因此，要保证验证操作确实已经执行，唯一的办法就是在服务器端也执行验证。你可以使用许多内建的验证对象，例如RegularExpressionValidator，它们能够自

动生成验证用的客户端脚本，当然你也可以插入服务器端的方法调用。如果找不到现成的验证对象，你可以通过CustomValidator自己创建一个。
⑸ 将用户登录名称、密码等数据加密保存。加密用户输入的数据，然后再将它与数据库中保存的数据比较，这相当于对用户输入
的数据进行了“消毒”处理，用户输入的数据不再对数据库有任何特殊的意义，从而也就防止了攻击者注入SQL命令。System.Web.Security.FormsAuthentication类有一个
HashPasswordForStoringInConfigFile，非常适合于对输入数据进行消毒处理。
⑹ 检查提取数据的查询所返回的记录数量。如果程序只要求返回一个记录，但实际返回的记录却超过一行，那就当作出错处理。

(7)使用预处理语句

## 8.你可以用什么来确保表格里的字段只接受特定范围里的值?

master 主要保存系统级的信息，比如本数据库实例都有哪些数据库 ，都有哪些账号等，需备份；
model 模板，每创建一个数据库 ，都会根据这个库的结构来创建，如果改过此库，建议备份；
msdb 保存计划任务，作业之类的信息，需备份，否则会丢失作业和备份计划；
tempdb 用户对sqlserver操作时产生的临时数据依赖于此库，最常见的是临时表，不许备份；

## 9.有哪些操作会使用到TempDB；如果TempDB异常变大，可能的原因是什么，该如何处理；

每个sqlserver运行时所产生的临时数据都会用到tempdb，最常见的是执行sql脚本需要返回的记录集；异常变大的原因是执行的操作返回的记录集过大造成，找出该语句优化，减少数据范围，或者分批操作这些数据

## 10.Index有哪些类型，它们的区别和实现原理是什么，索引有啥优点和缺点；如何为SQL语句创建合适的索引，索引创建时有哪些需要,注意的项，如何查看你创建的索引是否被使用；如何维护索引；索引损坏如何检查，怎么修复；T-SQL有更好的索引存在，但是运行,时并没有使用该索引，原因可能是什么；

聚集索引，非聚集索引；聚集索引只能有一个，非聚集可有多个，数据依赖于聚集索引来保存，如果没有聚集索引，数据是一个乱序的堆；
优点：合适的索引可有效提高查询效率；缺点：过多的索引，在insert、update 和 delete 的时候增加索引的维护成本，降低并发量；
一般索引的创建要依赖于 where 和 order by 这两个关键字，执行计划可以看出是否用到了索引；
还没遇到过索引损坏的情况，如果损坏，重建之；
用不到索引可能是索引碎片过多 ，可进行碎片整理，若不行可加强制索引with(index( 索引名 ))

## 11.Job信息我们可以通过哪些表获取；系统正在运行的语句可以通过哪些视图获取；如何获取某个T-SQL语句的IO、Time等信息；

sql2000下是通过 master.dbo.sysjobs 来查看作业信息；系统正在运行的语句可通过 master.dbo.sysprocesses 结合 dbcc inputbuffer 来查看，IO，在sql2000下我本人都是通过profiler看reads，duration，sql2005下有了动态视图（dmv）；

北京群 群1：219690210，群2：377501688，群3：262827065，成都群：209844460
上海群 ：376029918　杭州群：376029918　广州群：344744167 深圳群：542733289
西安群：542733289　高级群：165150112

微信公众号:DotNet开发跳槽