**题目（.Net）**

1、存储过程的有哪些优点？为什么执行速度快

1.在数据库服务器中只有首次对存储过程中的命令进行编译，以后直接调用无需编译，加快执行速度。  
2.只提供给用户参数和结果，存储过程对查询过程封装和加密，简化用户使用，防止非法修改。  
3.存储过程可以用于降低网络流量，存储过程代码直接存储于数据库中，所以不会产生大量sql语句的代码流量。  
4.可以只赋给用户执行存储过程的权利，而不给用户操作相应数据表的权利，这样可以有效防止注入攻击。  
5.维护性高，更新存储过程通常比更改、测试以及重新部署程序集需要较少的时间和精力。

2、索引怎么理解？为什么要用索引？哪种情况下需要用索引？哪种情况下不需要用

为什么要创建索引呢？这是因为，创建索引可以大大提高系统的性能。   
第一，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。   
第二，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。   
第三，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。   
第四，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。   
第五，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。

在经常需要搜索的列上，可以加快搜索的速度；   
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；   
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；   
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；   
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；   
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。

同样，对于有些列不应该创建索引。一般来说，不应该创建索引的的这些列具有下列特点：

第一，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。   
第二，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。   
第三，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。   
第四，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。

3、如果一个类不想被继承应该怎么办？

sealed 修饰符可以应用于类、实例方法和属性。密封类不能被继承。密封方法会重写基类中的方法，但其本身不能在任何派生类中进一步重写。当应用于方法或属性时，sealed 修饰符必须始终与 override（C# 参考）

4、CLR是什么

CLR是公共语言运行时，Common Language Runtime)和Java虚拟机一样也是一个运行时环境，它负责资源管理（内存分配和垃圾收集），并保证应用和底层操作系统之间必要的分离

5、C#中的垃圾回收是怎样的回收机制？

GC的全称是garbage collection，中文名称垃圾回收，是.NET中对内存管理的一种功能。垃圾回收器跟踪并回收托管内存中分配的对象，定期执行垃圾回收以回收分配给没有有效引用的对象的内存。当使用可用内存不能满足内存请求时，GC会自动进行。

在进行垃圾回收时，垃圾回收器会首先搜索内存中的托管对象，然后从托管代码中搜索被引用的对象并标记为有效，接着释放没有被标记为有效的对象并收回内存，最后整理内存将有效对象挪动到一起。

6、值类型与引用类型的区别

1.    值类型的数据存储在内存的栈中；引用类型的数据存储在内存的堆中，而内存单元中只存放堆中对象的  
  
地址。

2.     值类型存取速度快，引用类型存取速度慢。

3.     值类型表示实际数据，引用类型表示指向存储在内存堆中的数据的指针或引用

4.     值类型继承自System.ValueType，引用类型继承自System.Object

5.     栈的内存分配是自动释放；而堆在.NET中会有GC来释放

6.      值类型的变量直接存放实际的数据，而引用类型的变量存放的则是数据的地址，即对象的引用。  
  
7.　值类型变量直接把变量的值保存在堆栈中，引用类型的变量把实际数据的地址保存在堆栈中，而实际

数据则保存在堆中。注意，堆和堆栈是两个不同的概念，在内存中的存储位置也不相同，堆一般用于存储

可变长度的数据，如字符串类型;而堆栈则用于存储固定长度的数据，如整型类型的数据int(每个int变量

占用四个字节)。由数据存储的位置可以得知，当把一个值变量赋给另一个值变量时，会在堆栈中保存两

个完全相同的值;而把一个引用变量赋给另一个引用变量，则会在堆栈中保存对同一个堆位置的两个引用

，即在堆栈中保存的是同一个堆的地址。在进行数据操作时，对于值类型，由于每个变量都有自己的值，

因此对一个变量的操作不会影响到其它变量;对于引用类型的变量，对一个变量的数据进行操作就是对这

个变量在堆中的数据进行操作，如果两个引用类型的变量引用同一个对象，实际含义就是它们在堆栈中保

存的堆的地址相同，因此对一个变量的操作就会影响到引用同一个对象的另一个变量。

8、介绍String和StringBuilder类

//最大的区别体现在字符串拼接的时候  
string str1="hello ";  
string str2="world";  
str1 += str2;  
//此时内存中存在三个对象分别为str1("hello world"), str2("hello"),以及一个没有任何引用的垃圾  
//对象"hello"(该对象,最终会被GC收集,清理)  
StringBuilder sb=new StringBuilder();  
sb.Append("hello ");  
sb.Append("world");  
//此时内存中只有sb("hello world")一个对象  
//所以在做大规模的字符串拼接的时候(比较常见的是ADO.NET中查询语句的拼接),推荐使用  
//StringBuilder而不是string,因为用string的话,你的内存将会十分可怕,不要太过依赖GC,因为GC  
//并不是实时的,不是说C#程序员就可以完全不管内存,能控制的还是要尽量控制.

9、什么是C#中的垃圾回收

首先说下C#中的变量类型吧，C#中有2个变量类型，一种是值类型，一种是引用类型，值类型是在栈上创建，这一类型用不到GC，引用类型是在堆中创建，GC主要是在这里管理对象。GC对每个对象有个引用计数，所有说只要有变量在引用它，计数器就不为了，一个变量不再引用这个对象，对象的计数器就减一，知道计数器为0时，对象就成为内存垃圾了（没有变量引用它），但是此时垃圾并没有回收。那什么时候回收呢，是在内存占用超过一定限度是，GC才启动，释放垃圾资源，说白了就是delete这些对象，将空间归还给系统。但是这还没完，空间释放后，内存空间就不连续了，所有GC还要赶一件事，就是将空间整理下，将占用的空间连续话，具体说就是将空间向上推，就是想高地值转存，这样空间就连续了，使用也方便了，然后GC就改变应用那些对象的变量里地地址，让他们指向正确的位置，所以说C#中的引用类型就是一种指针，一种动态改变值的指针。

10、SP的优点（存储过程）

1.   运行速度：   大多数高级的数据库系统都有statement   cache的，所以编译sql的花费没什么影响。但是执行存储过程要比直接执行sql花费更多（检查权限等），所以对于很简单的sql，存储过程没有什么优势。       
  2.   网络负荷：如果在存储过程中没有多次数据交互，那么实际上网络传输量和直接sql是一样的。       
  3.   团队开发：很遗憾，比起成熟的IDE，没有什么很好存储过程的IDE工具来支持，也就是说，这些必须手工完成。       
  4.   安全机制：对于传统的C/S结构，连接数据库的用户可以不同，所以安全机制有用；但是在web的三层架构中，数据库用户不是给用户用的，所以基本上，只有一个用户，拥有所有权限（最多还有一个开发用户）。这个时候，安全机制有点多余。       
  5.   用户满意：实际上这个只是要将访问数据库的接口统一，是用存储过程，还是EJB，没太大关系，也就是说，在三层结构中，单独设计出一个数据访问层，同样能实现这个目标。       
  6.   开发调试：一样由于IDE的问题，存储过程的开发调试要比一般程序困难（老版本DB2还只能用C写存储过程，更是一个灾难）。       
  7.   移植性：算了，这个不用提，反正一般的应用总是绑定某个数据库的，不然就无法靠优化数据库访问来提高性能了。       
  8.   维护性：的确，存储过程有些时候比程序容易维护，这是因为可以实时更新DB端的存储过程，但是在3层结构下，更新server端的数据访问层一样能实现这个目标，可惜现在很多平台不支持实时更新而已。

11、举例说明多态

[多态](http://zhidao.baidu.com/search?word=%E5%A4%9A%E6%80%81&fr=qb_search_exp&ie=utf8)（Polymorphism）按字面的意思就是“多种形状”。引用Charlie Calverts对[多态](http://zhidao.baidu.com/search?word=%E5%A4%9A%E6%80%81&fr=qb_search_exp&ie=utf8" \t "_blank)的描述——[多态性](http://zhidao.baidu.com/search?word=%E5%A4%9A%E6%80%81%E6%80%A7&fr=qb_search_exp&ie=utf8)是允许你将父对象设置成为和一个或更多的他的[子对象](http://zhidao.baidu.com/search?word=%E5%AD%90%E5%AF%B9%E8%B1%A1&fr=qb_search_exp&ie=utf8)相等的技术，赋值之后，父对象就可以根据当前赋值给它的[子对象](http://zhidao.baidu.com/search?word=%E5%AD%90%E5%AF%B9%E8%B1%A1&fr=qb_search_exp&ie=utf8)的特性以不同的方式运作（摘自“Delphi4 [编程技术](http://zhidao.baidu.com/search?word=%E7%BC%96%E7%A8%8B%E6%8A%80%E6%9C%AF&fr=qb_search_exp&ie=utf8)[内幕](http://zhidao.baidu.com/search?word=%E5%86%85%E5%B9%95&fr=qb_search_exp&ie=utf8)”）。简单的说，就是一句话：允许将子类类型的指针赋值给父类类型的指针。[多态性](http://zhidao.baidu.com/search?word=%E5%A4%9A%E6%80%81%E6%80%A7&fr=qb_search_exp&ie=utf8" \t "_blank)在Object Pascal和C++中都是通过[虚函数](http://zhidao.baidu.com/search?word=%E8%99%9A%E5%87%BD%E6%95%B0&fr=qb_search_exp&ie=utf8" \t "_blank)（Virtual Function）实现的。

[多态性](http://zhidao.baidu.com/search?word=%E5%A4%9A%E6%80%81%E6%80%A7&fr=qb_search_exp&ie=utf8)是允许将父对象设置成为和一个和多个它的[子对象](http://zhidao.baidu.com/search?word=%E5%AD%90%E5%AF%B9%E8%B1%A1&fr=qb_search_exp&ie=utf8)相等的技术，比如Parent:=Child；

[多态](http://zhidao.baidu.com/search?word=%E5%A4%9A%E6%80%81&fr=qb_search_exp&ie=utf8)的作用：把不同的子[类对象](http://zhidao.baidu.com/search?word=%E7%B1%BB%E5%AF%B9%E8%B1%A1&fr=qb_search_exp&ie=utf8)都当作父类来看，可以屏蔽不同子[类对象](http://zhidao.baidu.com/search?word=%E7%B1%BB%E5%AF%B9%E8%B1%A1&fr=qb_search_exp&ie=utf8)之间的差异，写出通用的代码，做出通用的编程，以适应需求的不断变化。

赋值之后，父对象就可以根据当前赋值给它的子对象的特性以不同的方式运作。也就是说，父亲的行为像儿子，而不是儿子的行为像父亲

**就是一个父类派生出来的多个子类，这些子类都继承了父类的共性，但各自拥有自己的特性，也就是说子类是父类的多种形态（多态） 你可以通过实例化某个子类来将对象传给父类，让父类（共性）去完成你所实例化的字类的属性或方法（特性）。**

12、OOP的三大特性

封装：就是将一个类的使用和实现分开，只保留部分接口和方法与外部联系

继承：子类自动继承其父级类中的属性和方法，并可以添加新的属性和方法或者对部分属性和方法进行重写。继承增加了代码的可重用性。

多态：多个子类中虽然都具有同一个方法，但是这些子类实例化的对象调用这些相同的方法后却可以获得完全不同的结果，多态性增强了软件的灵活性

13、 interface 和 abstract class的区别

|  |  |  |
| --- | --- | --- |
|  | Abstract class | interface |
| 实例化 | 不能 | 不能 |
| 类 | 一种继承关系，一个类只能使用一次继承关系。可以通过继承多个接口实现多重继承 | 一个类可以实现多个interface |
| 数据成员 | 可有自己的 | 静态的不能被修改即必须是static final，一般不在此定义 |
| 方法 | 可以私有的，非abstract方法，必须实现 | 不可有私有的，默认是public，abstract 类型 |
| 变量 | 可有私有的，默认是friendly 型，其值可以在子类中重新定义，也可以重新赋值 | 不可有私有的，默认是public static final 型，且必须给其初值，实现类中不能重新定义，不能改变其值。 |
| 设计理念 | 表示的是“is-a”关系 | 表示的是“like-a”关系 |
| 实现 | 需要继承，要用extends | 要用implements |

14、聚集索引与非聚集索引的区别

索引实际上是一组指向表中数据的指针,索引的排列顺序其实就是这组指针的顺序.  
聚集索引:表的物理存储顺序与指针(即逻辑)顺序相同  
[非聚集索引](http://zhidao.baidu.com/search?word=%E9%9D%9E%E8%81%9A%E9%9B%86%E7%B4%A2%E5%BC%95&fr=qb_search_exp&ie=utf8):物理与[逻辑顺序](http://zhidao.baidu.com/search?word=%E9%80%BB%E8%BE%91%E9%A1%BA%E5%BA%8F&fr=qb_search_exp&ie=utf8" \t "_blank)不同  
因为一个表只能有一个物理顺序,所以,[聚集索引](http://zhidao.baidu.com/search?word=%E8%81%9A%E9%9B%86%E7%B4%A2%E5%BC%95&fr=qb_search_exp&ie=utf8)的个数最多只能是1.  
其中[唯一索引](http://zhidao.baidu.com/search?word=%E5%94%AF%E4%B8%80%E7%B4%A2%E5%BC%95&fr=qb_search_exp&ie=utf8" \t "_blank)是给所做的索引增加了唯一性的约束,添加,修改索引列中数据时,不允许出现重复值.它可以是[聚集索引](http://zhidao.baidu.com/search?word=%E8%81%9A%E9%9B%86%E7%B4%A2%E5%BC%95&fr=qb_search_exp&ie=utf8" \t "_blank),也可以是非聚集的,就看你如何定义  
主键索引是把主键列定义为索引,主键具有唯一性,所以主键索引是[唯一索引](http://zhidao.baidu.com/search?word=%E5%94%AF%E4%B8%80%E7%B4%A2%E5%BC%95&fr=qb_search_exp&ie=utf8" \t "_blank)的一种特殊形式

**个通俗的举例，说明两者的区别**

      其实，我们的汉语字典的正文本身就是一个聚集索引。比如，我们要查“安”字，就会很自然地翻开字典的前几页，因为“安”的拼音是“an”，而按照拼音排序汉字的字典是以英文字母“a”开头并以“z”结尾的，那么“安”字就自然地排在字典的前部。如果您翻完了所有以“a”开头的部分仍然找不到这个字，那么就说明您的字典中没有这个字；同样的，如果查“张”字，那您也会将您的字典翻到最后部分，因为“张”的拼音是“zhang”。也就是说，字典的正文部分本身就是一个目录，您不需要再去查其他目录来找到您需要找的内容。我们把这种正文内容本身就是一种按照一定规则排列的目录称为“聚集索引”。

　   如果您认识某个字，您可以快速地从自动中查到这个字。但您也可能会遇到您不认识的字，不知道它的发音，这时候，您就不能按照刚才的方法找到您要查的字，而需要去根据“偏旁部首”查到您要找的字，然后根据这个字后的页码直接翻到某页来找到您要找的字。但您结合“部首目录”和“检字表”而查到的字的排序并不是真正的正文的排序方法，比如您查“张”字，我们可以看到在查部首之后的检字表中“张”的页码是672页，检字表中“张”的上面是“驰”字，但页码却是63页，“张”的下面是“弩”字，页面是390页。很显然，这些字并不是真正的分别位于“张”字的上下方，现在您看到的连续的“驰、张、弩”三字实际上就是他们在非聚集索引中的排序，是字典正文中的字在非聚集索引中的映射。我们可以通过这种方式来找到您所需要的字，但它需要两个过程，先找到目录中的结果，然后再翻到您所需要的页码。我们把这种目录纯粹是目录，正文纯粹是正文的排序方式称为“非聚集索引”。

15、怎么实现线程同步

16、怎么SQL优化查询

选择最有效率的表名顺序(只在基于规则的优化器中有效)： ORACLE的解析器按照从右到左的顺序处理FROM子句中的表名，FROM子句中写在最后的表(基础表 driving table)将被最先处理，在FROM子句中包含多个表的情况下,你必须选择记录条数最少的表作为基础表。如果有3个以上的表连接查询, 那就需要选择交叉表(intersection table)作为基础表, 交叉表是指那个被其他表所引用的表.

（2） WHERE子句中的连接顺序．： ORACLE采用自下而上的顺序解析WHERE子句,根据这个原理,表之间的连接必须写在其他WHERE条件之前, 那些可以过滤掉最大数量记录的条件必须写在WHERE子句的末尾.

（3） SELECT子句中避免使用‘ \* ‘： ORACLE在解析的过程中, 会将'\*' 依次转换成所有的列名, 这个工作是通过查询[数据字典](http://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E5%AD%97%E5%85%B8&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6" \t "_blank)完成的, 这意味着将耗费更多的时间

（4）减少访问数据库的次数： ORACLE在内部执行了许多工作: 解析SQL语句, 估算索引的利用率, 绑定变量 , 读数据块等；

（5）在SQL\*Plus , SQL\*Forms和Pro\*C中重新设置ARRAYSIZE参数, 可以增加每次数据库访问的检索数据量 ,建议值为200

（6）使用DECODE函数来减少处理时间： 使用DECODE函数可以避免重复扫描相同记录或重复连接相同的表.

（7）整合简单,无关联的数据库访问： 如果你有几个简单的数据库查询语句,你可以把它们整合到一个查询中(即使它们之间没有关系)

（8）删除重复记录： 最高效的删除重复记录方法 ( 因为使用了ROWID)例子：DELETE FROM EMP E WHERE E.ROWID > (SELECT MIN(X.ROWID) FROM EMP X WHERE X.EMP\_NO = E.EMP\_NO);

（9）用TRUNCATE替代DELETE： 当删除表中的记录时,在通常情况下, [回滚](http://www.baidu.com/s?wd=%E5%9B%9E%E6%BB%9A&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)段(rollback segments ) 用来存放可以被恢复的信息. 如果你没有COMMIT事务,ORACLE会将数据恢复到删除之前的状态(准确地说是恢复到执行删除命令之前的状况) 而当运用TRUNCATE时, [回滚](http://www.baidu.com/s?wd=%E5%9B%9E%E6%BB%9A&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)段不再存放任何可被恢复的信息.当命令运行后,数据不能被恢复.因此很少的资源被调用,执行时间也会很短. (译者按: TRUNCATE只在删除全表适用,TRUNCATE是DDL不是DML)

（10）尽量多使用COMMIT： 只要有可能,在程序中尽量多使用COMMIT, 这样程序的性能得到提高,需求也会因为COMMIT所释放的资源而减少: COMMIT所释放的资源:

a. [回滚](http://www.baidu.com/s?wd=%E5%9B%9E%E6%BB%9A&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)段上用于恢复数据的信息.

b. 被程序语句获得的锁

c. redo log buffer 中的空间

d. ORACLE为管理上述3种资源中的内部花费

（11）用Where子句替换HAVING子句： 避免使用HAVING子句, HAVING 只会在检索出所有记录之后才对结果集进行过滤. 这个处理需要排序,总计等操作. 如果能通过WHERE子句限制记录的数目,那就能减少这方面的开销. (非oracle中)on、where、having这三个都可以加条件的子句中，on是最先执行，where次之，having最后，因为on是先把不符合条件的记录过滤后才进行统计，它就可以减少中间运算要处理的数据，按理说应该速度是最快的，where也应该比having快点的，因为它过滤数据后才进行sum，在两个表联接时才用on的，所以在一个表的时候，就剩下where跟having比较了。在这单表查询统计的情况下，如果要过滤的条件没有涉及到要计算字段，那它们的结果是一样的，只是where可以使用rushmore技术，而having就不能，在速度上后者要慢如果要涉及到计算的字段，就表示在没计算之前，这个字段的值是不确定的，根据上篇写的[工作流程](http://www.baidu.com/s?wd=%E5%B7%A5%E4%BD%9C%E6%B5%81%E7%A8%8B&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)，where的作用时间是在计算之前就完成的，而having就是在计算后才起作用的，所以在这种情况下，两者的结果会不同。在多表联接查询时，on比where更早起作用。系统首先根据各个表之间的联接条件，把多个表合成一个临时表后，再由where进行过滤，然后再计算，计算完后再由having进行过滤。由此可见，要想过滤条件起到正确的作用，首先要明白这个条件应该在什么时候起作用，然后再决定放在那里

（12）减少对表的查询： 在含有[子查询](http://www.baidu.com/s?wd=%E5%AD%90%E6%9F%A5%E8%AF%A2&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6)的SQL语句中,要特别注意减少对表的查询.例子：SELECT TAB\_NAME FROM TABLES WHERE (TAB\_NAME,DB\_VER) = ( SELECT TAB\_NAME,DB\_VER FROM TAB\_COLUMNS WHERE VERSION = 604)

（13）通过内部函数提高SQL效率.： 复杂的SQL往往牺牲了执行效率. 能够掌握上面的运用函数解决问题的方法在实际工作中是非常有意义的

（14）使用表的别名(Alias)： 当在SQL语句中连接多个表时, 请使用表的别名并把别名前缀于每个Column上.这样一来,就可以减少解析的时间并减少那些由Column歧义引起的语法错误.

（15）用EXISTS替代IN、用NOT EXISTS替代NOT IN： 在许多基于基础表的查询中,为了满足一个条件,往往需要对另一个表进行联接.在这种情况下, 使用EXISTS(或NOT EXISTS)通常将提高查询的效率. 在[子查询](http://www.baidu.com/s?wd=%E5%AD%90%E6%9F%A5%E8%AF%A2&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6" \t "_blank)中,NOT IN子句将执行一个内部的排序和合并. 无论在哪种情况下,NOT IN都是最低效的 (因为它对[子查询](http://www.baidu.com/s?wd=%E5%AD%90%E6%9F%A5%E8%AF%A2&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6" \t "_blank)中的表执行了一个全表[遍历](http://www.baidu.com/s?wd=%E9%81%8D%E5%8E%86&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6" \t "_blank)). 为了避免使用NOT IN ,我们可以把它改写成[外连接](http://www.baidu.com/s?wd=%E5%A4%96%E8%BF%9E%E6%8E%A5&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6" \t "_blank)(Outer Joins)或NOT EXISTS.

例子：（高效）SELECT \* FROM EMP (基础表) WHERE EMPNO > 0 AND EXISTS (SELECT ‘X' FROM DEPT WHERE DEPT.DEPTNO = EMP.DEPTNO AND LOC = ‘MELB') (低效)SELECT \* FROM EMP (基础表) WHERE EMPNO > 0 AND DEPTNO IN(SELECT DEPTNO FROM DEPT WHERE LOC = ‘MELB')

（16）识别'低效执行'的SQL语句： 虽然目前各种关于SQL优化的图形化工具层出不穷,但是写出自己的SQL工具来解决问题始终是一个最好的方法：SELECT EXECUTIONS , DISK\_READS, BUFFER\_GETS, ROUND((BUFFER\_GETS-DISK\_READS)/BUFFER\_GETS,2) Hit\_radio, ROUND(DISK\_READS/EXECUTIONS,2) Reads\_per\_run, SQL\_TEXT FROM V$SQLAREA WHERE EXECUTIONS>0 AND BUFFER\_GETS > 0 AND (BUFFER\_GETS-DISK\_READS)/BUFFER\_GETS < 0.8 ORDER BY 4 DESC;

（17）用索引提高效率： 索引是表的一个概念部分,用来提高检索数据的效率，ORACLE使用了一个复杂的自平衡B-tree结构. 通常,通过索引查询数据比全表扫描要快. 当ORACLE找出执行查询和Update语句的最佳路径时, ORACLE优化器将使用索引. 同样在联结多个表时使用索引也可以提高效率. 另一个使用索引的好处是,它提供了主键(primary key)的唯一性验证.。那些LONG或LONG RAW[数据类型](http://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E7%B1%BB%E5%9E%8B&hl_tag=textlink&tn=SE_hldp01350_v6v6zkg6), 你可以索引几乎所有的列. 通常, 在大型表中使用索引特别有效. 当然,你也会发现, 在扫描小表时,使用索引同样能提高效率. 虽然使用索引能得到查询效率的提高,但是我们也必须注意到它的代价. 索引需要空间来存储,也需要定期维护。</SPAN>

17、SP和View有什么好处

提高查询效率

19、事务的四大特性

:原子性(Atomicity)  
 事务是数据库的逻辑工作单位，事务中包括的诸操作要么全做，要么全不做。  
B:一致性(Consistency)  
 事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态。一致性与原子性是密切相关的。  
C:隔离性(Isolation)  
 一个事务的执行不能被其他事务干扰。  
D:持续性/永久性(Durability)  
 一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。

20、SP的优点，什么样的业务逻辑该写在存储过程中

存储过程只在创造时进行编译，以后每次执行存储过程都不需再重新编译，而一般 SQL

语句每执行一次就编译一次,所以使用存储过程可提高数据库执行速度。

2.当对数据库进行复杂操作时(如对多个表进行

Update,Insert,Query,Delete

时），可将此复杂操作用存储过程封装起来与数据库提供的事务处理结合一起使用。这些操作，如果用程序来完成，就变成了一条条的 SQL

语句，可能要多次连接数据库。而换成存储，只需要连接一次数据库就可以了。

3.存储过程可以重复使用,可减少数据库开发人员的工作量。

4.安全性高,可设定只有某此用户才具有对指定存储过程的使用权。

21、C++中指针函数与函数指针的区别

指针函数式一种函数，返回值是某一类型的指针（返回的是地址），函数指针是指针变量，指向函数的变量。

22、string s = null 与string s=””的区别

空值和无（占用空间和不占用空间）

23、事件与委托的区别

事件是一种特殊的委托，加了event特殊关键字的委托

24、多态的表现形式

一种是父类与子类之间表现 重写   
还有一种是重载 在同一个类中，同样的方法，有不同的实现方式  
这两种也是多态的表现

25 、c# java 区别

C#是基于.net Framewrok的，原理跟java差不多，编译成中间代码，在不同的平台上再翻译成相应的机器语言。  
最大的区别就是……java的应用比C#广泛，虽然C#也自称跨平台语言（它也确实有这个能力），不过除了微软自家的系统，没有其他系统支持。  
java就不一样，包括windows、linux、mac os x、symbian，甚至非智能的山寨手机都支持。  
C#的优点是和.NET 平台上的其他语言兼容性好，在windows平台下的执行效率还蛮高的。语法和C/C++比较接近，便于C程序员上手

26 、overload overwrite 区别 overwrite 的条件

Overload：重载，是一个类中定义了多种重名的方法，只是参数不同或者类型不同，overwrite是重写，是父类和子类之间多态性的表现，子类中的方法和参数与父类相同的话说明是重写

27、oracle 外连接内连接区别

Inner join查询出的内容必须是where条件中的元素都存在于两表的，外联的话有一个表为主表，主表中的内容不管子表中是否有都会展现出

28、tibrv类型有哪些

29、tibrv怎么用的

30、操作系统中，内存管理的方式有哪些

可以分为页式、段式和段页式三种。

31、interface与abstract class的区别，interface中为什么不能声明为protected？

只有继承的子类才能访问protected类型成员 接口是public的，所以接口不能声明为protected

33、委托和虚方法的区别

具体看书

34、简要阐述冒泡排序法以及其时间复杂度

太复杂了

35、SP预编译的概念

在第一次执行时放入内存，之后就是直接调用，很好的防止了sql注入

36、有那几种排序算法

汽泡排序法 对以初步排序的数据来说这种方法的速度很快 在其它情况下运行速度较慢   
选择排序法 非常简单 对大量数据的排序速度很慢   
容易明白   
对于少量数据的排序来说速度很快   
快速排序法 对大量数据的排序来说速度很快 如果有大量重复的数据就比较麻烦   
计数排序法 当数据数值较小(1-1000之间)时，速度非常快 当数据数值较大时，速度较慢   
需额外的内存   
只能对整数类型的数据排序

37、有1-N个自然数，从其中拿走一个数，请找出被拿走的那个自然数

http://blog.csdn.net/yangcs2009/article/details/38384657

39 、SQL和oracle的区别

但SQL Sever在Window平台上的表现，和Windows操作系统的整体结合程度，使用方便性，和Microsoft开发平台的整合性都比Oracle强的很多。但Windows操作系统的稳定性及可靠性大家是有目共睹的

Sql只能在windows上使用，oracle可以适应任何平台

41 、page生命周期

 （1）请求页面：页请求发生在页生命周期开始之前。

（2）开始：在开始阶段，将设置页属性，如Request和Response。在此阶段，页还将确定请求是回发请求还是新请求，并设置IsPostBack属性。

（3）初始化页面：页面初始化期间，可以使用页中的控件，并将设置每个控件的UniqueID属性。如果当前请求是回发请求，则回发数据尚未加载，并且控件属性值尚未还原为视图状态中的值。

（4）加载页面：加载期间，如果当前请求是回发请求，则将使用从视图状态和控件状态恢复的信息加载控件属性。

（5）验证：在验证期间，将调用所有验证程序控件的Validate方法，此方法将设置各个验证程序控件和页的IsValid属性。

（6）回发事件处理：如果请求是回发请求，则将调用所有事件处理程序。

（7）呈现页面：在页面呈现期间，视图状态将被保存到页面，然后页面将调用每个控件，以将其呈现的输出提供给页的Response属性的OutputStream。

（8）卸载页面：完全呈现页、将页发送至客户端并准备丢弃时，将调用卸载。此时，将卸载页属性（如Response和Request）并执行清理。

与上面的生命周期相对应的Page类的生命周期事件如图所示。

42、如果一个类不想被继承应该怎么办？

Sealed

sealed 修饰符可以应用于类、实例方法和属性。密封类不能被继承。密封方法会重写基类中的方法，但其本身不能在任何派生类中进一步重写。当应用于方法或属性时，sealed 修饰符必须始终与 override（C# 参考）

上述问题是我对之前的资料的整理

接下来几个是补充的：

1. 简单谈一下触发器的概念
2. 简单说一下回文函数
3. 简单说一下webservice
4. 传参的类型有哪几种，分别有什么优点

值传递和引用传递。

1.对于string类型的理解

具体的百度一下,我也不是非常理解,只知道是引用类型

2.泛型的约束有哪些

http://www.cnblogs.com/ottox/archive/2009/03/02/1401307.html

3.委托能否是静态的

可以是静态的

4.c#中using的几种用法

1、引用命名空间

2、为命名空间或类型创建别名

3、使用using语句，定义一个范围，在范围结束时处理对象。(不过该对象必须实现了IDisposable接口)

5.c#是怎么自己进行垃圾回收的

http://blog.csdn.net/yangzhenping/article/details/12561529

6.如何进行Sql的优化

http://www.cnblogs.com/ATree/archive/2011/02/13/sql\_optimize\_1.html

7.装箱和拆箱时内存中会如何变化

http://www.xuebuyuan.com/504758.html

8.简述多线程

http://www.jb51.net/article/46234.htm

9.简述线程池和Task任务

http://www.cnblogs.com/x-xk/archive/2012/12/11/2804563.html

10.面向对象五大基本原则

http://www.cnblogs.com/seacryfly/archive/2011/12/29/seacryfly.html

**单一职责原则（SRP）  
开放封闭原则（OCP）   
里氏替换原则（LSP）   
依赖倒置原则（DIP）   
接口隔离原则（ISP）**

1. 你决定去泡MM，明确泡的是MM而不是GG，这是抽象类。  
   2.在是MM的前提下，根据MM的身高、身材、MiMi大小等划定一个范围，这是继承。  
   3.有的MM面目可憎，可是有魔鬼身材，你也决定把她纳入你的范围，这是重写。  
   4.你比较忙，把你的要求告诉你哥们，让他帮你留意，这是委托。  
   5.你哥们根据你的要求，找到了一个，但他确居为私有，这是异常。  
   6.由你选定的范围具体到某一MM，这是实例化。  
   7.你带MM逛商场，一路上MM唠叨不停，你不胜其烦，把她寄存起来，这是序列化。  
   8.你自己逛完，从寄存处带MM回去，这是反序列化。  
   9.你对MM不满意或者MM对你不满意，和你ByeBye了，接着你又泡了几个MM，还是ByeBye了，你痛定思痛，总结出一套泡MM的方法，并且决定以后按这套方法泡MM，这是封装。  
   10.你和你公司的几个MM同时保存恋爱关系，这是.net remoting。  
   11.你在网上找到一个MM，可是这个MM通过网络，同时在和几个GG网恋，这是Web Service。  
   12.在你泡的几个MM中，你带她们逛商场，有的喜欢买衣服，有的喜欢买化妆品，这是重载。  
   13.你发现泡MM都有一个Kiss的过程，这是接口。  
   14.你有点心灰意冷，决定让婚介公司帮你找个适合的MM，这是依赖注入或者控制反转。

英文题基本之前的资料都涉及了。一些问题是面试官根据自我介绍和项目介绍中延伸开来问的，这里没法一一举例。