1. 简述ORM原理

ORM(Object Relation Maping),对象关系映射框架，实现思想就是把数据库中表数据映射为对象。.net中比较常用的有Entity Framwork，在项目中比较常用dapper，

Entity Framwork

Database first模式中是生成一个Entity实体对象，该对象包含了创建的类型和类型与数据库的映射文件。

Code first是通过反射获取对象字段名称映射到数据库列名的，也可以通过Attribute来进行修改，或者在OnModelCreating方法中进行关系映射的配置。

1. 如何设置数据库访问的超时时间。
2. 那些原因会导致不能使用数据库连接池的连接

主要考虑连接占满的情况，也就是连接没有及时释放。

1. .NET代码中的事务和数据库中的事务是如何衔接传递的。

类似session一样的东西？mongodb是这样实现的，sql server还需要查看

1. 强类型DataSet优缺点。

C#是强类型语言，DataSet是运行时确定架构，很多时候使用时需要注意架构，否则只 有运行时才会发现异常，优点就是使用比较方便，是一个关系型数据集合，还可以创建 多个表间关系

1. 如果让你设计上海市公安局的户籍数据库，针对如此庞大的数据量你有何具体的措施

优化索引，聚集索引上的键最好是int、datetime等类型，string类型的索引效率比较低。

数据存储优化，可以进行分库分表，按照区进行分库，不同区的户籍信息存储在不同的库中，身份证号中本来就包含了区号等信息，这样进行按照身份证号查询时也比较方便。

如果需要按照姓名进行查询，需要从不同库中进行查询，如果是并行执行，那么效率也不会很低，比所有数据存储在一个库中进行查询要高。

1. 在读取数据库数据时，DataAdapter和DataReader之间如何取舍，判断依据时什么。

DataAdapter内部也是使用DataReader实现的，用DataReader读取数据填充DataSet;所以使用得当的话，这两种方式都是没有问题的，在使用DataReader的时候需要注意使用CommandBehavior.CloseConnection来创建DataReader，

1. 现在有一个Excle、一个文本文件和一个orcle数据库，你需要将这些数据整合导入到sql server中，你会采用什么方式。

Sql Server Manager Studio程序来导入。

1. 数据库事务和隔离级别
2. 事务的四大属性

原子性：它所做的对数据修改操作要么全部执行，要么完全不执行

一致性：数据的更改结果是我们所期待的，就是数据一致性，比如递增操作，一个int变量为1，两个线程同时递增操作时，可能的修改结果是2，但是我们期望是3，这个就是数据不一致，程序中可以加锁来保证数据一致性。事务一致性同理，事务修改的结果和我们期望一致。

隔离性：隔离性指并发的事务是相互隔离的。即一个事务内部的操作及正在操作的数据必须封锁起来，不被其它企图进行修改的事务看到

持久性：持久性意味着当系统或介质发生故障时，确保已提交事务的更新不能丢失。即一旦一个事务提交，DBMS保证它对数据库中数据的改变应该是永久性的，耐得住任何数据库系统故障。持久性通过数据库备份和恢复来保证。

1. 存储过程的定义和优缺点

CREATE PROCEDURE [dbo].[Find\_CollecterDead\_Single]

-- Add the parameters for the stored procedure here

@F\_BuildID char(10),

@F\_SpanHour int = 3

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

SELECT @F\_BuildID, @F\_SpanHour

END

GO

缺点：会有些业务逻辑脱离程序，写在存储过程中，当有逻辑变更时需要修改存储过程，这样会使得数据库和业务挂钩，不太好。当替换数据库的时候也会导致问题。

优点：程序调用方便，存储过程只返回需要用到的数据

1. Sql server的完整性约束

–主键约束（Primary Key Constraint）：要求主键列数据唯一，并且不允许为空

–唯一约束（Unique Constraint）：要求该列唯一，允许为空，但只能出现一个空值。

–检查约束（Check Constraint）：某列取值范围限制、格式限制等，如有关年龄的约束

–默认约束（Default Constraint）：某列的默认值，如我们的男性学员较多，性别默认为“男”

–外键约束（Foreign Key Constraint）：用于两表间建立关系，需要指定引用主表的那列

1. 数据库三范式
2. Where、group by、having、order by 顺序

SELECT COUNT(\*)

FROM [TTBEMS\_Core].[dbo].[T\_A\_Temp]

where F\_Temp1=''

group by F\_Temp1

having count(\*)>0

ORDER BY F\_Temp1 DESC

Having可以使用函数，group by不能使用函数

1. Exists、not exists

是进行布尔类型判断的表达式

1. SET NOCOUNT ON;
2. Sql 语句启用事务
3. 索引的创建和优缺点

优点：当查询命中索引时，能比较高效的查询数据

缺点：索引会影响数据的插入效率，每次数据插入，每条索引都要创建一个该条数据的引用。

1. 创建视图和视图优缺点

create view View\_EdsProd

as

select \* from Tab\_EdsProd where Mid>**1**

go

优点：简单性(把数据进行了筛选，相对简单)、安全(只能查询和修改视图所看到的数据)、逻辑数据独立性(筛选条件是一个相对独立的逻辑条件，把需要用到的数据和其他数据隔离开了，聚焦特定数据)

缺点：性能、修改限制

1. 什么时候适合使用视图

考虑库存管理的一个场景：最普通的单据是入库和出库单，库管员需要看到当前的库存。对库存的处理，我们有两个方案：一是使用视图，所有的入库减去所有的出库，就是当前库存；另外就是使用中间表，建立一个库存表，记录当前的库存。

     1、使用视图的方案

     入库时，系统记录入库单据；

     出库时，系统查询库存视图，判断是否有充足的库存可以出库，然后记录出库单据；

     可见，系统只需要记录入库和出库单据，库存的计算是由DBMS在查询视图时进行的；

     2、使用库存表的方案

     入库时，系统记录入库单据，同时增加相应的库存；

     出库时，系统查询库存表，判断是否有充足的库存可以出库，然后记录出库单据，减少相应的库存；

     可见，系统除了记录入库和出库单据外，还需要更新库存表的当前库存数量；

     3、方案的比较

     对系统本身的设计和编码来说，视图方案易于实现，测试方便；库存表方案则稍微复杂。从这点上看，视图方案可以在原型阶段大展身手。

     用户体验到的性能方面，视图方案的性能压力在查询库存上，库存表方案的性能压力在业务处理上：

     视图方案：由于每次查询库存，DBMS都需 要扫描入库和出库单据，查询时间长；还可能会对入库和出库单据加锁，导致入库和出库处理延长，甚至失败（尤其是查询库存视图在一个事务中时）；库存视图如 果和其它表或者视图连接，构成复杂的SQL时，由于索引不能有效（或无法）使用，查询速度会更慢；

     库存表方案：库存表上可以建索引，查询速度比视图会快很多；在入库和出库时，更新库存表的SQL会对出入库处理的速度有一些影响，但是由于更新只影响出入库的SKU，与查询库存表并发时，加锁时间非常短，影响会比较小。

     4、结论

     视图方案适用情形：原型，数据量比较小；

     库存表方案适用情形：数据量比较大，针对库存的分析较多；

**个人感觉：经常用到且查询复杂的数据可以使用视图，方便查看**

1. SQLSERVER服务器中，给定表 table1 中有两个字段 ID、LastUpdateDate，ID表示更新的事务号， LastUpdateDate表示更新时的服务器时间，请使用一句SQL语句获得最后更新的事务号
2. 写出一条Sql语句：取出表A中第31到第40记录（SQLServer,以自动增长的ID作为主键,注意：ID可能不是连续的。

解1: select top 10 \* from A where id not in (select top 30 id from A)

解2: select top 10 \* from A where id > (select max(id) from (select top 30 id from A )as A)

1. 什么叫做SQL注入，如何防止？请举例说明

利用sql关键字对网站进行攻击。过滤关键字'等

1. Select \* 和select a,b,c有什么区别

网络开销，可能不是所有的列都要返回，特别是有些不需要的大数据列

Join连接时，可能别的表新增相同列名，导致原有查询语句异常。

拿MS SQL SERVER来说，除开上面提及到的网络开销，因为可能不是所有列都是需要被返回的，以及提到的其实一种编码规范和性能关系不大还算着边的回答，真正会在下面这种情况影响性能。SQL SERVER有一种索引叫”覆盖”索引，和一般非聚集索引的区别是可以把指定的非键列也和键列储存在索引页，比如一个表有十列，项目有个需求需要频繁的据列a，查找列b和列C. 此时可以给列a建非聚集索引并”覆盖”bc,那么这样一个通过select b,c和select\*会有很大的不同，前者可以通过INDEX seek(因索引覆盖此处无需rid或key lookup)，而后者则是通过INDEX seek以及根据索引页rid或key去表数据页lookup，而lookup是有IO消耗的，在索引碎片较大的情况下，性能甚至会低于INDEX scan

1. with tmp临时表

with data as (select \* from meterdata where xxxxxxxx)

select m.name,d.value from meters m join data d on m.id = d.id

1. 什么是领域驱动设计ddd

这个是根据业务领域来划分对象职能的对象建模方法，和软件三层架构(表示层UI、业务逻辑层BLL和数据访问层DAL)并不冲突，领域驱动设计可以让逻辑层业务划分更加清晰，即便使用三层架构，当业务逐渐增大的时候，后台代码业务逻辑可能会逐渐混乱，以至于不好重构或者修改，领域驱动设计就是解决这个问题的。这样看来领域驱动设计和微服务感觉很相似，其实微服务就是领域驱动设计的理想实现方式。