**数据库 database**

**数据库的概念：**

-> 数据库database：存储数据的仓库

-> DBMS：数据库管理系统，不同品牌的DBMS有不同的特点

-> SQLSERVER是一种服务性的数据库，也就是说我们需要通过网络协议去访问SQLSERVER在本机安装的服务才能进行操作

**使用数据库存储数据的特点：**

1. 安全性
2. 并发访问处理（两个程序同时访问一个数据文件时会发生错误，而数据库通过给程序 "加锁" 很好的解决了这个问题）
3. 高效率
4. 海量数据存储
5. 数据完整性

**数据库设计中的三大范式（指导数据库设计得标准）：**

1. 第一范式：确保每列保持原子性，也就是说确保每一列都具备单一的职责
2. 第二范式：确保表中的每列都和主键相关
3. 第三范式：确保每列都和主键列直接相关,而不是间接相关

**数据库中文件组的使用：**

1. 通过文件组，可以把数据库中的表 指定放到不同的磁盘上（文件组上），以达到并行使用多个磁盘的io资源，提高读写效率
2. 文件组帮助我们优化刚并发访问的效率

**数据库的管理和登陆操作：**

-> 数据库的实例名：区分安装在同一个机器上的不同的MSSQLServer服务，唯一标识性

-> 连接数据库时，服务器名称的写法 微信截图_20180305164316

1. "(local)" 、"." 和 "机器名" 的效果一样，代表连接本机
2. 当想访问实例名的时候，通过 "(local)" 、"." 和 "机器名" \实例名，如图：微信截图_20180305164613
3. 通过IP地址：前提保证TCP/IP的服务是开启的
4. 127.0.0.1：回环地址，连接回本机的地址
5. 局域网IP地址
6. 外网IP地址

**数据库的组成：**

-> 数据库

-> 框架(表的分组)

-> 表：命名规范同c#中的类

-> 列 Columm：字段、类型、长度、约束等

-> 行 Row

-> 主键：数据行的唯一标识，有两种类型，自动增长类型、GUID全球唯一标识类型

-> 外键：关联到另一个表的主键

**数据库的字段类型详解：**

### **Character 字符串：** 一个字节表示一个字符（英文）

|  |  |  |
| --- | --- | --- |
| **数据类型** | **描述** | **存储** |
| char(n) | 固定长度的字符串。最多 8,000 个字符。 | n |
| varchar(n) | 可变长度的字符串。最多 8,000 个字符。 |  |
| varchar(max) | 可变长度的字符串。最多 1,073,741,824 个字符。 |  |
| text | 可变长度的字符串。最多 2GB 字符数据。 |  |

### **Unicode 字符串： 两个字节表示一个字符（中文）**

|  |  |  |
| --- | --- | --- |
| **数据类型** | **描述** | **存储** |
| nchar(n) | 固定长度的 Unicode 数据。最多 4,000 个字符。 |  |
| nvarchar(n) | 可变长度的 Unicode 数据。最多 4,000 个字符。 |  |
| nvarchar(max) | 可变长度的 Unicode 数据。最多 536,870,912 个字符。 |  |
| ntext | 可变长度的 Unicode 数据。最多 2GB 字符数据。 |  |

### Binary 类型：

|  |  |  |
| --- | --- | --- |
| **数据类型** | **描述** | **存储** |
| bit | 允许 0、1 或 NULL |  |
| binary(n) | 固定长度的二进制数据。最多 8,000 字节。 |  |
| varbinary(n) | 可变长度的二进制数据。最多 8,000 字节。 |  |
| varbinary(max) | 可变长度的二进制数据。最多 2GB 字节。 |  |
| image | 可变长度的二进制数据。最多 2GB。 |  |

### Number 类型：

|  |  |  |
| --- | --- | --- |
| **数据类型** | **描述** | **存储** |
| tinyint | 允许从 0 到 255 的所有数字。 | 1 字节 |
| smallint | 允许从 -32,768 到 32,767 的所有数字。 | 2 字节 |
| int | 允许从 -2,147,483,648 到 2,147,483,647 的所有数字。 | 4 字节 |
| bigint | 允许介于 -9,223,372,036,854,775,808 和 9,223,372,036,854,775,807 之间的所有数字。 | 8 字节 |
| decimal(p,s) | 固定精度和比例的数字。允许从 -10^38 +1 到 10^38 -1 之间的数字。  p 参数指示可以存储的最大位数（小数点左侧和右侧）。p 必须是 1 到 38 之间的值。默认是 18。  s 参数指示小数点右侧存储的最大位数。s 必须是 0 到 p 之间的值。默认是 0。 | 5-17 字节 |
| numeric(p,s) | 固定精度和比例的数字。允许从 -10^38 +1 到 10^38 -1 之间的数字。  p 参数指示可以存储的最大位数（小数点左侧和右侧）。p 必须是 1 到 38 之间的值。默认是 18。  s 参数指示小数点右侧存储的最大位数。s 必须是 0 到 p 之间的值。默认是 0。 | 5-17 字节 |
| smallmoney | 介于 -214,748.3648 和 214,748.3647 之间的货币数据。 | 4 字节 |
| money | 介于 -922,337,203,685,477.5808 和 922,337,203,685,477.5807 之间的货币数据。 | 8 字节 |
| float(n) | 从 -1.79E + 308 到 1.79E + 308 的浮动精度数字数据。参数 n 指示该字段保存 4 字节还是 8 字节。float(24) 保存 4 字节，而 float(53) 保存 8 字节。n 的默认值是 53。 | 4 或 8 字节 |
| real | 从 -3.40E + 38 到 3.40E + 38 的浮动精度数字数据。 | 4 字节 |

### Date 类型：

|  |  |  |
| --- | --- | --- |
| **数据类型** | **描述** | **存储** |
| datetime | 从 1753 年 1 月 1 日 到 9999 年 12 月 31 日，精度为 3.33 毫秒。 | 8 bytes |
| datetime2 | 从 1753 年 1 月 1 日 到 9999 年 12 月 31 日，精度为 100 纳秒。 | 6-8 bytes |
| smalldatetime | 从 1900 年 1 月 1 日 到 2079 年 6 月 6 日，精度为 1 分钟。 | 4 bytes |
| date | 仅存储日期。从 0001 年 1 月 1 日 到 9999 年 12 月 31 日。 | 3 bytes |
| time | 仅存储时间。精度为 100 纳秒。 | 3-5 bytes |
| datetimeoffset | 与 datetime2 相同，外加时区偏移。 | 8-10 bytes |
| timestamp | 存储唯一的数字，每当创建或修改某行时，该数字会更新。timestamp 基于内部时钟，不对应真实时间。每个表只能有一个 timestamp 变量。 |  |

### 其他数据类型：

|  |  |
| --- | --- |
| **数据类型** | **描述** |
| sql\_variant | 存储最多 8,000 字节不同数据类型的数据，除了 text、ntext 以及 timestamp。 |
| uniqueidentifier | 存储全局标识符 (GUID)。 |
| xml | 存储 XML 格式化数据。最多 2GB。 |
| cursor | 存储对用于数据库操作的指针的引用。 |
| table | 存储结果集，供稍后处理。 |

**数据库的迁移方案：**

1. 分离/脱机 -> 附加

如遇到权限问题的解决方案：

1. 启用Administrator账号
2. 把数据库文件放到默认的数据库文件存放目录
3. 设置数据库的兼容级别：2005/2008/2012
4. 备份 -> 还原
5. 生成SQL脚本