**MCP**

**主讲老师：张胜利**

1. **什么是数据库？**

**数据库（DB）是存放数据的仓库，只不过这些数据存在一定的关联，并按一定的格式存放在计算机内。广义上讲，数据不仅包含数字，还包括文本、图像、音频、视频等。**

**（二）什么是数据库管理系统？**

**数据库管理系统（DBMS）是管理数据库的系统，它按一定的数据模型组织数据。**

**DBMS应提供如下功能：**

**（1）数据定义功能：可定义数据库中的数据对象。**

**（2）数据操纵功能：可对数据库表进行基本操作，如插入、删除、修改、查询等。**

**（3）数据的完整性检查功能：保证用户输入的数据满足相应的约束条件。**

**（4）数据库的安全保护功能：保证只有具有权限的用户才能访问数据库中的数据。**

**（5）数据库的并发控制功能：使多个应用程序可在同一时刻并发地访问数据库的数据。**

**（6）数据库系统的故障恢复功能：使数据库在运行出现故障时进行数据库恢复，以保证数据库可靠运行。**

**（7）在网络环境下访问数据库的功能。**

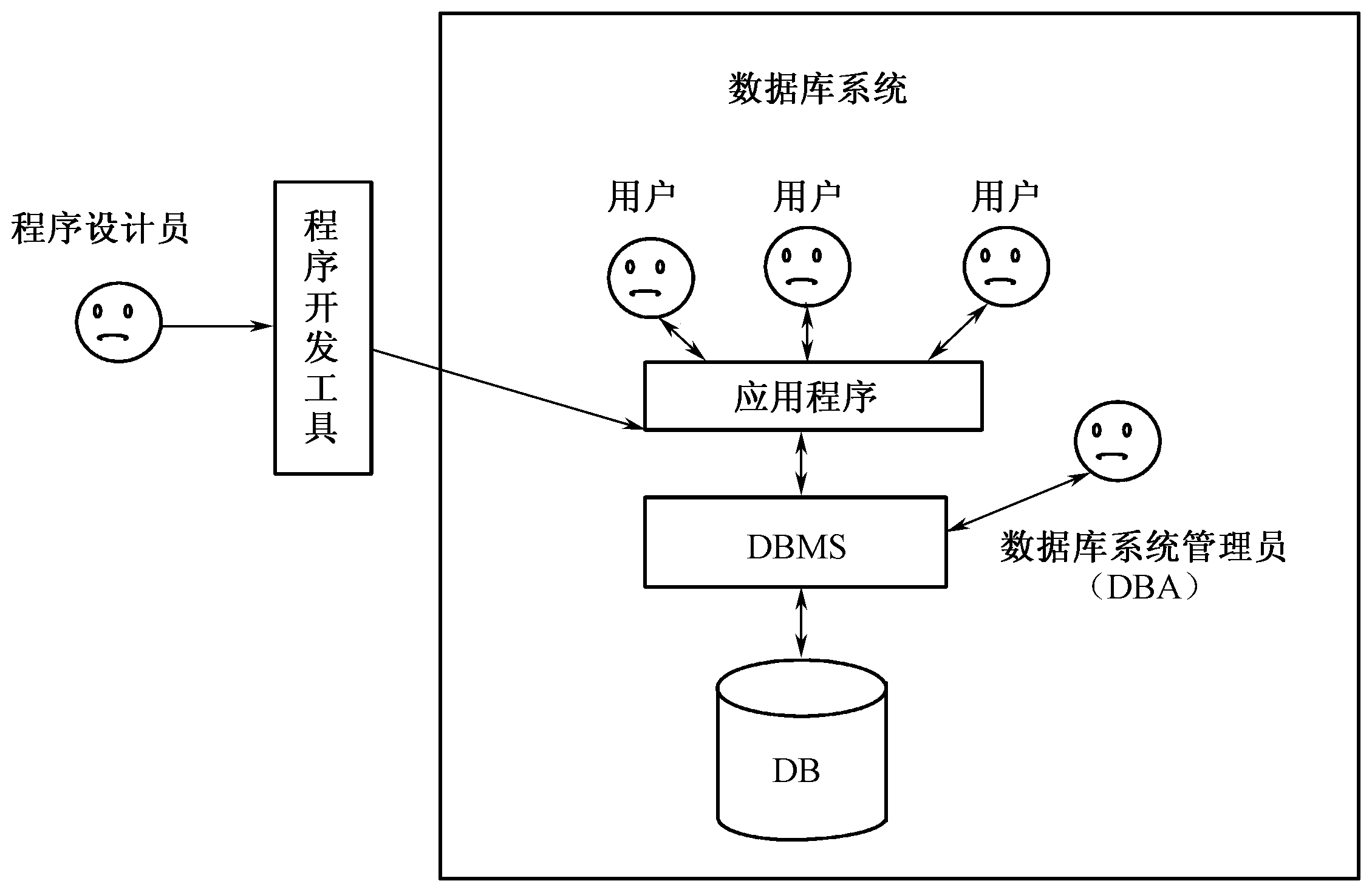
**（8）方便、有效地存取数据库信息的接口和工具。编程人员通过程序开发工具与数据库的接口编写数据库应用程序。数据库系统管理员（DBA，DataBase Adminitrator）通过提供的工具对数据库进行管理。**

**（三）请列举常见的数据库管理系统：**

1. **Oracle：大型商用数据库，收费挺高，但目前很多大型企业、银行在使用它。但前两年曾爆出可口可乐公司因为Oracle太贵而准备换成其他类型的。**
2. **SqlServer：中大型数据库，商用也需要收费。很多中小企业在使用它。微软出品，秉承了文档齐全、操作简单方便的一贯优点，非常适合入门学习。**
3. **MySql：中型数据库，商用有免费版。MySql原来与Oracle和SqlServer差距较大，但经过多年的发展，现在的MySql 功能也已经非常完善。谷歌等不少著名公司等一直在使用。**
4. **Access：小型数据库。这个数据库的最大优点是 文件型 的，在Windows平台上，只要有Access文件就能用，不像上边的几种，需要进行安装。**
5. **SQLite：和Access有点类似，也是一种文件型数据库，但可以运行在Windows、Linux、Android手机等多种平台上，在这一点上比Access要好。**

**（四）数据库系统的构成：**

**数据、数据库、数据库管理系统与操作数据库的应用程序，加上支撑它们的硬件平台、软件平台和与数据库有关的人员一起构成了一个完整的数据库系统。如图示：**

**.（五）数据模型：**

1. **层次模型：以树状层次结构组织数据。**
2. **网状模型：每一个数据用一个节点表示，每个节点与其他节点都有联系，这样，数据库中的所有数据节点就构成了一个复杂的网络。**
3. **关系模型：以二维表格（关系表）的形式组织数据库中的数据。**
4. **关系型数据库语言:**

**SQL结构化查询语言(Structured Query Language)**

**功能：**

* **数据查询语言DQL(Data Query Language )**

**由SELECT子句，FROM子句，WHERE子句组成的查询块**

* **数据操纵语言DML(Data Manipulation Language)**

**主要有三种形式：  
1) 插入：INSERT 2) 更新：UPDATE 3) 删除：DELETE**

* **数据定义语言DDL(Data Definition Language)  
  数据定义语言DDL用来创建数据库中的各种对象表、视图、索引等如：CREATE TABLE/VIEW/INDEX**
* **数据控制语言DCL(Data Control Language)  
  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。  
  3) COMMIT [WORK]：提交。**

**（七）数据库设计：**

1. **概念结构设计：**

**E-R模型：**

**用矩形框表示实体集,实体的属性用椭圆框表示,用线段连接实体集与属性,实体间的联系用菱形框表示,当一个属性或属性组合指定为主码时,在实体集与属性的连接线上标记一斜线。**

**一对一的联系(1 : 1) 一对多的联系(1 : n) 多对多的联系(m : n)**

**Eg:系、学生和课程作为实体集;一个系有多个学生，而一个学生仅属于一个系，所以系和学生之间是一对多的联系;一个学生可以选修多门课程，而一门课程有多个学生选修，所以学生和课程之间是多对多的联系。**

1. **逻辑结构设计：**

**E－R图向关系模式的转换。**

**（八）数据类型：  
在计算机中数据有两种牲征：类型和长度**

**系统提供的数据类型分为几大类：**

1. **整数数据类型:**
2. **INT （INTEGER）**

**存储从-2的31次方(-2,147,483,648)到2的31次方(2,147,483,647)之间的所有正负整数。 每个INT 类型的数据按4 个字节存储。**

1. **SMALLINT**

**存储从-2的15次方(-32,768)到2的15次方-1 (32,767)之间的所有正负整数。每个SMALLINT类型的数据占用2 个字节的存储空间。**

1. **TINYINT**

**存储从0 到255 之间的所有正整数。每个TINYINT类型的数据占用1 个字节的存储空间。**

1. **BIGINT**

**存储从-2^63 (-9223372036854775808) 到2^63-1 (9223372036854775807)之间的所有正负整数。每个BIGINT 类型的数据占用8个字节的存储空间。**

1. **浮点数据类型(只入不舍)**
2. **REAL**

**可精确到第7 位小数，其范围为从-3.40E -38 到3.40E +38。 每个REAL类型的数据占用4 个字节的存储空间。**

1. **FLOAT**

**可精确到第15 位小数，其范围为从-1.79E -308 到1.79E +308。 每个FLOAT 类型的数据占用8 个字节的存储空间。**

1. **DECIMAL**

**可以提供小数所需要的实际存储空间，但也有一定的限制，您可以用2 到17 个字节来存储从-10的38次方-1 到10的38次方-1 之间的数值。可精确到第３８位小数。例如：decimal （15 5），表示共有15 位数，其中整数10 位，小数5。**

1. **NUMERIC**

**NUMERIC数据类型与DECIMAL数据类型完全相同。**

1. **MONEY**

**货币数据值介于 -2^63 (-922,337,203,685,477.5808) 与 2^63 - 1 (+922,337,203,685,477.5807) 之间，精确到货币单位的千分之十。**

**可精确到第4位小数**

1. **SMALLMONEY**

**货币数据值介于 -214,748.3648 与 +214,748.3647 之间，精确到货币单位的千分之十。 可精确到第4位小数。**

1. **二进制数据类型**
2. **BINARY**

**用于存储二进制数据。其定义形式为BINARY（n）， n表示数据的长度，取值为1 到8000 。在输入数据时必须在数据前加上字符“0X” 作为二进制标识，如：要输入“abc ”则应输入“0xabc ”。**

1. **VARBINARY**

**VARBINARY 数据类型的定义形式为ARBINARY（n）。 n 的取值也为1 到8000，若输入的数据过长，将会截掉其超出部分。  
 一般情况下，由于BINARY 数据类型长度固定，因此它比VARBINARY 类型的处理速度快。**

1. **逻辑数据类型**

**占用1 个字节的存储空间，其值为0 或1 。如果输入0 或1 以外的值，将被视为1。 BIT 类型不能定义为NULL 值（所谓NULL 值是指空值或无意义的值）。**

1. **字符数据类型**
2. **CHAR**

**定义形式为CHAR(n), 以CHAR 类型存储的每个字符和符号占一个字节的存储空间。n表示所有字符所占的存储空间，n 的取值为1 到8000。**

1. **NCHAR**

**NCHAR 数据类型的定义形式为NCHAR(n)。 它与CHAR 类型相似。不同的是NCHAR数据类型n 的取值为1 到4000。**

1. **VARCHAR**

**VARCHAR数据类型的定义形式为VARCHA(n)。 它与CHAR 类型相似，n 的取值也为1 到8000，若输入的数据过长，将会截掉其超出部分.一般情况下，由于CHAR 数据类型长度固定，因此它比VARCHAR 类型的处理速度快。**

1. **NVARCHAR**

**NVARCHAR数据类型的定义形式为NVARCHAR(n)。 它与VARCHAR 类型相似。不同的是，NVARCHAR数据类型采用UNICODE 标准字符集（Character Set）， n 的取值为1 到4000。**

1. **文本和图形数据类型**

**这类数据类型用于存储大量的字符或二进制数据。**

1. **TEXT**

**存储大量文本数据，其容量理论上为1 到2的31次方-1 （2147483647）个字节，在实际应用时需要视硬盘的存储空间而定。**

1. **NTEX**

**NTEXT数据类型与TEXT.类型相似。不同的是NTEXT 类型采用UNICODE 标准字符集(Character Set)。**

1. **IMAGE**

**IMAGE 数据类型用于存储大量的二进制数据(Binary Data)。其存储数据的模式与TEXT 数据类型相同。**

1. **日期和时间数据类型**
2. **DATE和DATETIME**

**DATE是SQL Server 2008新引进的数据类型。（年月日）可以表示的日期范围从公元元年1月1日到9999年12月31日。只需要3个字节的存储空间。**

**DATETIME用于存储日期和时间的结合体。它可以存储从公元1753 年1 月1 日零时起到公元9999 年12 月31 日23 时59 分59 秒之间。需要8个字节的存储空间。**

1. **DATETIME 2**

**类似于之前DATETIME类型，不过其精度比较高，可以精确到小数点后面7位（100ns），其使用语法：datetime2(n)。**

**Eg: Declare @dt as datetime2(5) set @dt = getdate()**

**select @dt**

1. **Datetimeoffset**

**加入了时区偏移量部分，时区偏移量表示为 [+|-] HH:MM。 HH 是范围从 00 到 14 的 2 位数，表示时区偏移量的小时数。 MM 是范围从 00 到 59 的 2 位数，表示时区偏移量的附加分钟数。 时间格式支持到最小 100 毫微秒。 必需的 + 或 - 符号指示在 UTC（通用协调时间或格林尼治标准时间）中是加上还是减去时区偏移量以获取本地时间。**

**Eg: declare @dt as datetimeoffset(7)**

**Set @dt = '2008-08-08 08:08:08.0 +8:00' select @dt**

**八、特殊类型:Uniqueidentifier             
用来存储一个全局唯一标识符，即GUID。仅限于 36 个字符，超过该长度的字符将被截断。这个数几乎没有机会在另一个系统中被重建。可以使用NEWID 函数或转换一个字符串为唯一标识符来初始化具有唯一标识符的列.**

1. **使用NewID() 为UniqueIdentifier赋值，函数NewID() 用于产生随机的GUID数值，无序的GUID 。**
2. **有序的GUID**

**在一台指定的Server上，NewSequentialId() 产生的GUID是有序的，后产生的GUID比先产生的GUID数值要大，即该函数产生的GUID值是始终增加的，SQL Sever保证产生的GUID比之前的GUID大。 NewSequentialID()函数只能用于表列的Default约束中。**

**create table jobs(**

**id uniqueidentifier rowguidcol primary key not null**

**constraint [DF\_jobs\_id] default (NEWSEQUENTIALID()),**

**account varchar(64) not null,password varchar(64)not null)**

**insert into jobs(account,password) values('tom','0000'),('jim','1111');**

**select rowguidcol from jobs;**

**（九）xmL数据类型:**

**xml数据格式的转换:**

1. **for xml path;设置根节点名称<===>for xml raw,elements;**

**Eg:**

**select student\_no,name,sex,class from student**

**for xml path/path(‘’)/path(‘student’);**

1. **@设置节点内属性**

**Eg:**

**select student\_no as '@student\_no',name,sex,class from student for xml path;**

1. **/设置子节点**

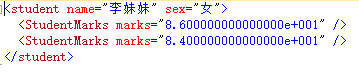
**Eg:**

**select student\_no as 'student/no',name,sex,class from student for xml path**

1. **for xml auto将表行数据转化为一行xml数据节点**

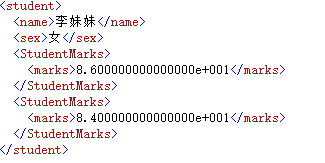
**Eg: select name,sex,marks from student inner join StudentMarks on Student.student\_no=StudentMarks.StudentCode**

**where Student.student\_no=1 for xml auto;**

****

1. **for xml auto,elements转为节点属性**

**select name,sex,marks from student inner join StudentMarks on Student.student\_no=StudentMarks.StudentCode where Student.student\_no=1 for xml auto,elements;**

****

1. **for xml raw**

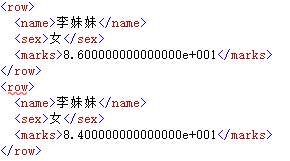
**返回数据行为元素，每一列（所有表）的值作为元素的属性，并以ROW为节点名称**

**select name,sex,marks from student inner join StudentMarks on Student.student\_no=StudentMarks.StudentCode where Student.student\_no=1 for xml raw;**

****

1. **for xml raw,elements;以row为根节点名称**

**select name,sex,marks from student inner join StudentMarks on Student.student\_no=StudentMarks.StudentCode where Student.student\_no=1 for xml raw,elements;**

****

**(十)XML类型的数据存储:**

**1、 xml**

**用于存储xml格式的文档，最大可达2GB，支持128级层次。**

**CREATE TABLE Person(Id int,Info xml)**

**--插入3条测试数据**

**INSERT Person VALUES(1,'<Person><ID>1</ID><Name>刘备</Name><job>经理</job></Person>')**

**INSERT Person VALUES(2,'<Person><ID>2</ID><Name>关羽</Name><job>战将</job></Person>')**

**INSERT Person VALUES(3,'<Person><ID>3</ID><Name>张飞</Name><job>大将</job></Person>‘)**

**--查询值value()方法**

**1、SELECT Id,Info.value('(/Person/Name)[1]','VARCHAR(50)') FROM Person WHERE ID = 2**

**2、SELECT \* FROM Person WHERE Info.value('(/Person/Name)[1]','VARCHAR(50)') = '张飞'**

**2、geography 地理数据**

**应用在椭圆体（或圆球）系统中，比如经纬度。**

**3、geometry 几何数据**

**应用于平面（或平面球）系统中，比如多边形。**