数据库系统原理

第一章节：数据库系统基本概念

1．文件系统阶段的数据管理有些什么缺陷？试举例说明。

三个：数据冗余；数据不一致性；数据联系弱。

2．与文件结构相比，数据库结构有些什么不同？

有下面三点不同：

数据结构化：文件由记录组成，文件间无联系。数据库逻辑结构用数据模型来描述，数据模型不仅描述数据本身的特点，还要描述数据之间的联系。

数据独立性：文件只有设备独立性，而数据库还具有逻辑独立性和物理独立性。

访问数据的单位：文件以记录为单位；数据库以字段为单位，增加了系统的灵活性。

3．试解释 DB、DBMS、DBS 三个术语。

ＤＢ是存储在一起、统一管理的相关数据的集合。

ＤＢＭＳ是位于用户与ＯＳ之间的一层数据管理软件，为用户或应用程序提供访问ＤＢ的方法。

ＤＢＳ是采用了数据库技术的计算机系统。

4．试叙述数据库系统中物理存储介质层次。

高速缓存、主存、快闪存、磁盘、光盘和磁带。

5．各种结构数据模型的数据结构有什么特点？

（１）在层次、网状模型中，数据之间联系是用指针实现的。这种方法查询速度较快，但带来一个缺点，查询语言是过程性的，应用程序的编写比较复杂。

（２）关系模型中，数据之间的联系通过关键码来体现。这种方法，查询过程是非过程性的，编程比较简单，但速度较慢。

6．数据之间联系在各种结构数据模型中是怎么实现的？

（１）在层次、网状模型中，数据之间联系是用指针实现的。

（２）关系模型中，数据之间的联系通过外键和主键间联系来体现。

7．试述概念模式在数据库结构中的重要地位。

数据库按外模式的描述是提供给用户，按内模式的描述是存储在磁盘中，而逻辑模式提供了连接这二级的相对稳定的中间观点，并使得二级的任何一级的改变都不受另一级的牵制。

8．为什么要在三级模式结构之间提供两个层次的映像？

由于三级模式之间（在结构、类型、长度等方面）往往差别很大，为了实现这三个抽象级别的联系和转换，ＤＢＭＳ在三级模式之间提供了二个层次的映像。

9．数据独立性与数据联系这两个概念有什么区别？

独立性是指应用程序和ＤＢ的数据之间相互独立，不受影响，对系统的要求是“数据独立性提高”

数据库联系是指记录之间的联系，对系统的要求是“数据联系密切”

10．什么是数据冗余？及其可能导致的后果？

数据冗余是指相同的数据被存储多次。后果是：浪费空间，造成数据操作异常，可能造成数据完整性的缺陷和数据不一致

11．何谓数据库语言？它主要包括哪两类？

数据库语言包括：数据定义（或描述）语言和数据操纵语言

数据描述语言（DDL）负责定义和描述数据库的各种特性，对数据库的逻辑设计和物理设计中所得到的数据模式进行定义和描述及数据库三级结构的描述。

数据操纵语言（DML）实现对数据库的操作，包括：检索、插入、删除和修改等操作。

12．什么是数据库（DB）？

数据库是统一管理的相关数据的集合。数据库是存储在计算机内的、有组织的、可共享的数据集合，具有较小的冗余度和较高的数据独立性和易扩展性。

13．什么是数据之间的联系，在关系数据库中，用什么方法实现数据之间的联系？

数据库不仅能存储数据，而且能存储数据之间的联系。在关系数据库中，通过表与表之间所所包含的公共属性实现数据之间的联系。利用这种联系能够将数据冗余度限定在最小范围之内，实现数据完整性约束和数据一致性控制。

14．什么是“DB 的系统缓冲区”？

在应用程序运行时，DBMS 在内存为其开辟一个 DB 的系统缓冲区，用于“数据的传输和格式的转换”

15．在 DBS 中数据字典（DD）起着什么作用？

DD 是存储三级结构的描述（即元数据）。DBMS 的所有工作都要以 DD 中的元数据为依据，也就是所有工作都要通过 DD 系统访问 DD。

16．数据库系统与文件系统相比怎样减少数据 冗余？

在文件管理系统中，数据文件之间没有联系，**数据冗余大。**

数据库系统以数据库方式管理大量共享的数据。数据库系统由许多单独文件组成的，文件内部具有完备的结构，但它更注重文件之间的联系。数据库系统中的数据具有共享性。数据库系统是面向整个系统的数据共享而建立的，各个应用的数据集中存储，共同使用，数据库文件之间联系密切，避免了数据的重复使用，减少和控制了数据的冗余。

17．什么是数据独立性？数据库系统如何实现数据独立性？数据独立性可带来什么好处？

数据独立性是指应用程序和数据之间相互独立、不受影响，即数据结构的修改不会引起应用程序的修改。

包括：物理数据独立性和逻辑数据独立性。

物理数据独立性是指数据库物理结构改变时不必修改现有的应用程序。

逻辑数据独立性是指数据库逻辑结构改变时不用改变应用程序。

数据独立性是由 DBMS 的二级映像功能来实现的。

当整个系统要求改变模式时（增加记录类型、增加数据项），由 DBMS 对各个外模式/模式的映像作相应改变，从而保证了数据的逻辑独立性。

当数据的存储结构改变时，由 DBMS 对模式/内模式的映像作相应改变，从而保证了数据的物理独立性。

数据独立性的好处是：

减轻了应用程序的维护工作量；

对同一数据库的逻辑模式，可以建立不同的用户模式，从而提高数据共享性，使数据库系统具有较好的可扩充性，给 DBA 维护、改变数据库的物理存储提供了方便。

18．数据库系统与数据库管理系统的主要区别是什么？

数据库系统是指在计算机系统中引入数据库后的系统构成，一般由**数据库、数据库管理系统、应用系统、数据库管理员和用户**构成。

数据库管理系统是位于用户与操作系统之间的一层数据管理软件，是数据库系统的一个重要组成部分。

第二章节 数据库设计和 ER 模型

1. 数据库系统的生存期分成哪几个阶段？数据库结构的设计在生存期中的地位如何？

分为七个阶段： 规划阶段、 需求分析
概念设计、 逻辑设计、 物理设计
实现阶段、 运行和维护阶段

数据库结构的设计是数据库应用系统设计的基础，它的好坏直接影响数据库的效率和质量，是数据库生存期中的一个非常重要的阶段。

2. 数据库设计的规划阶段应做哪些事情？

A、进行建立数据库的必要性及可行性分析。
B、确定数据库系统在组织中和信息系统中的地位。
C、以及各个数据库之间的联系。

3. 数据库设计的需求分析阶段是如何实现的？目标是什么？

这一阶段是计算机人员（系统分析员）和用户双方共同收集数据库所需要的信息内容和用户对处理的需求。并以需求说明书的形式确定下来，作为以后系统开发的指南和系统验证的依据。

需求分析的工作主要由下面四步组成：

A、 分析用户活动，产生业务流程图。
B、确定系统范围，产生系统关联图。
C、 分析用户活动涉及的数据，产生数据流图。
D、 分析系统数据，产生数据字典。

4. 数据字典的内容和作用是什么？

数据字典通常包括：数据项、数据流、数据结构、数据存储和处理过程五个部分。

数据字典是系统中各类数据描述的集合，是一系列二维表格，是进行详细的数据收集和数据分析所获得的主要成果。数据字典在数据库设计中占有很重要的地位。

5. 试叙述概念设计的步骤。

分三步完成：

A、 进行数据抽象，设计局部概念模式。
B、将局部概念模式综合成全局概念模式。
C、 对全局概念模式进行评审和确认。

6. 什么是 ER 图？构成 ER 图的基本要素是什么？

描述现实世界概念结构模型的有效方法称为 ER 方法，用 ER 方法建立的概念结构模型称为 ER 模型，或称为 ER 图。ER 图是由实体、实体的属性和实体之间的联系三个要素组成的。

7. 试述采用 ER 方法的数据库概念设计的过程。

A、设计局部 ER 模式：确定局部结构范围，实体定义，联系定义，属性分配。
B、设计全局 ER 模式：确定公共实体类型，ER 模式的合并，冲突的消除。
C、全局 ER 模式的优化：实体类型的合并，冗余属性的消除，冗余联系的消除。

8. 叙述 ER 图转换为关系模型的转换规则。

A、每一个实体分别转换成一个关系，实体的属性即是关系的属性，实体的键就是关系的键。
B、实体联系是一对一联系，将一个关系的键作为外键放在另一个关系中。
C、实体联系是一对多联系，则将“一”端的关系的键作为外键放在“多”端的关系中。
D、如果实体是多对多联系，则将联系单独转换成一个关系，该关系又称为交叉关系。这个关系的键

由与联系相关联的实体的键组合而成，联系的属性成为这个交叉关系的属性。

9．DBS 的运行维护工作，由谁担当？有哪些维护工作？

DBS 的运行维护工作由 DBA 担当，主要工作有四种：

A、DBS 的转储、恢复工作。

B、DBS 的安全性、完整性控制。

C、DBS 性能的监督、分析和改进。

D、DBS 的重组织和重构造。

10．什么是弱实体？什么是超类和子类？

一个实体的存在以其他实体存在为前提，此时称前者为“**弱实体**”，后者为“父实体”。父实体与弱实体的联系类型只能是：1:1 或者 1:N

在数据抽象的“概括”联系中，我们把下层概念称为子类，上层概念称为超类。此时，子类实体继承了超类实体的所有属性，但子类实体本身还可包含自己特有的属性。

11．评审在数据库设计中有什么重要的作用？为什么允许设计过程中有多次回溯与反复？

评审的作用在于确认某一阶段的任务是否全部完成，避免重大的错误。

为管理人员缺乏对计算机的了解，另一方面，数据库设计者缺乏对管理对象的了解，所以在评审时可能导致设计过程的回溯与反复。

12．在 DBD 中，为什么必须要有一个概念设计阶段？

如果将现实世界中的客观对象直接转换为机器世界中的对象，就会感到非常不方便，注意力往往被牵扯到更多的细节限制方面，而不能集中在最重要的信息的组织结构和处理模式上，因此往往是将客观对象先抽象为不依赖于任何具体机器的信息结构，这种信息结构不是 DBMS 支持的数据模型，而是概念模型。概念模型可以看成是现实世界到机器世界的一个过渡的中间层次，降低了 DBD 的复杂度。

13．在数据库设计中，对概念模型有些什么要求？

有以下四个要求：

A、 应有丰富的语义表达能力，能真实反映用户的各种要求。

B、 应简洁、明晰、独立于机器、容易理解。

C、 易于变动，容易修改和补充。

D、 应很容易向关系模型、层次模型、网状模型等转换。

14．叙述数据库的运行过程。

数据库运行过程中物理存储会不断变化，对数据库的设计进行评价、调整、修改是一个长期的任务，也是设计工作的继续和提高。维护工作由 DBA 来完成，维护工作包括：

(1)数据库的转储和恢复。

(2)数据库的安全性和完整性控制

(3)数据库的性能监督、分析和改造。

(4)数据库的重组织与重构造

15．什么是数据库的重组织和重构造？为什么要进行数据库的重组织与重构造？

数据库运行一段时间后，由于记录不断增、删、改会使数据库的物理存储情况变坏，降低了数据的存储效率，这时可由 DBA 对数据库进行重组织，按原设计要求重新安排存储位置、回收垃圾、减少指针，以提高效率。它不改变数据库的逻辑结构和物理结构。

由于数据库应用环境的变化，使原有的数据库设计不能满足新的要求，需要调整数据库的模式和内模式，这被称为数据库的重构造。（如：在表中增加或删除某此数据项，改变数据项的类型，增加或删除某个表），如果应用变化太大，可能就要重新设计新的数据库应用系统了。这个工作由 DBA 来完成。

第三章节 关系模式设计理论

1、试叙述函数依赖的定义

在关系模式 R（U）中，X，Y 是 U 的子集，r 是 R 的任一具体关系，如果对 r 的任意二个元组 t_1 、t_2 ，都有 $t_1[X]=t_2[X]$ 蕴涵 $t_1[Y]=t_2[Y]$ ，那么称 函数依赖 X → Y 在模式 R 上成立。

2、什么是二个函数依赖集的等价？

如果二个函数依赖集 F 和 G ，满足 $F^+=G^+$，则称 F 和 G 等价。此时，F 中每个 FD 都在 G^+中，而 G 中每个 FD 也都在 F^+中。

3、什么是平凡的 FD？平凡的 FD 可根据哪一条推理规则推出？

对于 FD X → Y，如果 Y ⊆ X ，那么称 X → Y 是一个平凡的 FD。平凡的 FD 根据自反性规则就可推出。

4、为什么要进行关系模式的分解？分解的依据是什么？

由于数据之间存在着联系和约束，在关系模式的关系中可能会存在数据冗余和操作异常现象，因此需把关系模式进行分解，以消除冗余和异常现象。

分解的依据是数据依赖和模式的标准（范式）。

5、简述多值依赖的定义。

给定关系模式 R 及其属性或属性组 A，B，当且仅当对于一给定的 A 值，与之相应有一组 B 值，且这组 B 值与关系中的其他属性值无关，则称多值依赖于 A，记为 A →→ B 。

6、什么是“无损分解”？有什么意义？

设关系模式 R 相对于 FD 集 F 分解成数据模式 *p*，如果对于 R 中每个关系 r 都有 $r = m_p(r)$ ，那么称 *p* 是无损分解.。 （此处 $m_p(r)$为 r 在 *p* 中每个模式上投影的自然连接）

无损分解，可以保证数据在投影以后通过自然连接仍能恢复回来。

7、什么是“保持 FD 的分解”？有什么意义？

设关系模式 R 相对于 FD 集 F 分解成数据模式 *p*，如果 F 在 *p* 中每个模式上的投影的并集，仍能逻辑蕴涵原来的 F，那么称分解 *p* 保持 FD 集。

8、关系模式的分解有什么优缺点？

二个优点：

A、消除冗余和异常。

B、在分解了的关系中可存储悬挂元组。

二个缺点：

A、可能分解了的关系不存在泛关系。

B、做查询操作，需做连接操作，增加了查询时间。

第四章节 关系运算

1. 为什么要对关系代数表达式进行优化？

关系代数表达式由关系代数操作组合而成。操作中，以笛卡尔积和连接操作最费时间，并生成大量的中间结果。如果直接按用户书写的顺序执行，很可能效率低。因此在执行前，先由 DBMS 的查询子系统对关系代数表达式进行检查，尽可能先执行选择和投影操作，以减少中间结果，并节省时间。

优化工作是由 DBMS 做的，用户在书写关系代数表达式时不必关心优化这件事，仍以简捷的连接形式书写。

2. 为什么关系中元组没有先后顺序？

由于关系定义为元组的集合，而集合中的元素是没有顺序的，因此关系中的元组也就没有先后的顺序（对用户而言）。这样既能减少逻辑排序，又便于在关系数据库中引进集合论的理论。

3. 试叙述两个关系代数表达式等价的定义。

二个关系代数表达式等价是指用同样的关系实例代替二个表达式中相应关系时所得到的结果是一样的。也就是相同的属性集和相同的元组集，但元组中属性的顺序可能不一致。

4. 试述查询优化在关系数据库系统中的重要性和可能性。

关系数据语言具有非过程化的特点。用户只要指出“做什么”，“怎样做”是由系统实现的。这给用户带来了方便，使对数据的操作简便，但却加重了系统的负担。系统需要自行选择存取路径，而存取路径选择的好坏是影响查询效率的关键所在。因此查询优化就成了关系系统需要解决的一个重要问题。

利用优化的一般策略和关系代数表达式的等价变换规则，就可以对关系代数表达式进行优化。经过优化的关系代数表达式可能不是所有等价的关系代数表达式中最优的，但一般情况下，能使查询效率大大提高。

5. 为什么关系中不允许有重复元组？

每个关系模式都有一个主键，在关系中主健值是不允许重复的。如果关系中有重复元组，那么其主键值肯定相等，起不了唯一标识作用，因此关系中不允许有重复元组。

6. 笛卡儿积、等值连接、自然连接三者之间有什么区别？

笛卡儿积是一个基本操作，而等值连接和自然连接是组合操作。

设关系 R 的元数为 r，元组个数为 m；

关系 S 的元数为 s，元组个数为 n 。

那么 R × S 的元数为 r+s ，元组个数为 m × n

$R \underset{i=j}{\bowtie} S$ 的元数为 r+s ，元组个数为小于等于 m × n

$R \bowtie S$ 的元数为小于等于 r+s ，元组个数为 小于等于 m × n

7、 元组演算表达式中这些符号的意义（ $\exists$、$\forall$、=>、∧、∨、¬ ）

$\exists$：存在量词　　$\forall$：全称量词

P1=>P2：若 P1 为真，则 P2 必然为真　　∧ ：交

∨ ：并　　¬ ：否

第五章节 SQL 语言

1．试叙述 SQL 的关系代数特点和元组演算特点。

SQL 的 SELECT 语句的基本句法来自于关系代数表达式$\prod_L(Q_F(R1\times\ldots\times Rm))$，并且 SQL 中有并（UNION）、交（INTERSECT）和差（EXCEPT）等操作，因此 SQL 具有关系代数特点。

SELECT 语句中出现的基本表名，都应该理解成基本表中的元组变量，而列名应理解成元组分量，这样ＳＱＬ就具有了元组演算的特点。

2．SQL 语言对于“查询结果是否允许存在重复元组”是如何实现的？

对于 SELECT 语句中 SELECT 子句，若用“SELECT　DISTINCT” 形式，则查询结果中不允许有重复元组；若不写成 DISTINCT　字样，则查询结果中允许出现重复元组。

3．嵌入式 SQL 的预处理方式是如何实现的？这种方式有什么重要意义？

预处理方式是先用预处理程序对源程序进行扫描，识别出ＳＱＬ语句，并处理成主语言的函数调用形式；然后再用主语言的编译程序把源程序编译成目标程序。

这种方法的重要意义在于不必改动主语言的编译程序，这样，ＳＱＬ的编译程序和主语言的编译程序之间就可独立，互不影响。

4．视图的优点是什么？

A、视图对于数据库的重构造提供了一定程序的逻辑独立性。

Ｂ、简化了用户观点。

Ｃ、视图机制使不同的用户能以不同的方式看待同一数据集合。

D、视图机制对机密文件数据提供了自动的安全保护功能。

5．建索引的目的是什么？是否索引建得越多越好？

建立索引的目的是为了快速查询，建索引的个数不是越多越好，如果数据增、删、改频繁，系统会花费许多时间来维护索引（因为当关系中的元组进行增、删、改时，索引由系统进行自动修改），解决的办法是删除不必要的索引。

6．基本表与视图两者的区别和联系是什么？

基本表是实际存在物理存储介质上的实表，而视图是个虚表，它在实际的存储介质上不存在，只是在数据字典中存储了视图结构的定义，它是由一个或几个基本表或其他视图导出的表。用户对基本表和视图的查询是相同的，对视图的更新有一些限制。

7．在嵌入式 SQL 中是如何解决数据库工作单位与源程序工作单位之间的通讯的？

在嵌入式ＳＱＬ语句中引用主语言程序变量，简称共享变量，完成数据库工作单元与源程序工作单元之间的通讯。主变量前加“:”，以区别于数据库的字段名，共享变量在使用前应预先用ＳＱＬ的ＤＥＣＬＡＲＥ语句声明。

8．解释游标的概念。

在主语言程序中，SELECT 命令在操作数据集合上定义一个游标时，系统将分配一个临时缓冲区保存该元组数据集合，游标是在程序中设定的一个临时指针，它指向 SELECT 命令当前要处理的元组。

9．SQL 的数据定义主要包括哪些操作？

SQL 的数据定义主要包括对ＳＱＬ模式（即数据库）、基本表、视图和索引的创建和撤消。

10．对视图的更新操作，有些什么限制？

视图中只有“行列子集视图”是允许更新的。如果视图是从单个基本表只使用选择、投影操作导出的，并

且包含了基本表的主键，那么这种视图称为“行列子集视图”。其他方式构成的视图都不允许更新。

另外，允许更新的视图在定义时，必须加上“WITH CHECK OPTION ”短语。

11．在宿主语言的程序中使用 SQL 语句有哪些规定？

A、分清 SQL 语句和主语言语句。所有 SQL 语句必须加上前缀“EXEC SQL”和结束标志“END_EXEC”

B、允许嵌入的 SQL 语句引用主语言的程序变量（引用时加冒号作为前缀），但不允许主语言引用 SQL 数据库的表名和列名。

C、SQL 的集合处理方式与主语言单记录处理方式之间用游标机制协调。

12．SQL 的集合处理方式与主语言单记录处理方式之间如何协调？

ＳＱＬ和主语言之间用游标机制协调。

游标有二层含义：游标关系和游标指针。

游标是与某一查询结果相联系的符号名。实际上把查询结果看成一个“文件”。如同ＯＳ一样，对“文件”的操作有定义、打开、推进、关闭等四个。游标推进相当于“读”操作，读一个元组值到共享变量中，让主语言的程序去处理。

13．嵌入式 SQL 的 DML 语句何时不必涉及到游标？何时必须涉及到游标？

不必涉及到游标的ＤＭＬ语句有下面二种情况：

Ａ、insert/delete/update 语句，只要加上前缀和结束标志，就能嵌入在主语言程序中使用。

B、对于 select 语句，如果已知查询结果肯定是单元组，也可不必涉及游标操作。

涉及到游标的ＤＭＬ语句有下面二种情况：

A、当 select 语句查询结果是多个元组时，必须用游标机制把多个元组一次一个地传递给主程序处理。

B、对游标指向元组进行修改或删除操作时，也涉及到游标。

14．为什么将 SQL 中视图称为“虚表”？

在 SQL 中创建一个视图时，系统只是将视图的定义存放在数据字典中，并不存储视图对应的数据，在用户使用视图时才去求对应的数据。因此，我们将视图称为“虚表”。这样处理的目的是为了节约存储空间，因为视图对应的数据都可从相应的基本表中获得。

15．所有的视图是否都可以更新？为什么？

由于视图是不实际存储数据的虚表，因此对视图的更新最终要转换为对基本表的更新。

在 SQL 中，允许更新的视图在定义时，必须加上“WITH CHECK OPTION”子句。

16．预处理方式对于嵌入式 SQL 的实现有什么重要意义？

预处理方式是先用预处理程序对程序进行扫描，识别 SQL 语句，并处理主语言的函数调用形式；然后再用主语言的编译程序把源程序编译成目标程序。

17．试说明 SQL 语言支持的关系数据库三级结构。

SQL 语言支持的数据库体系结构基本上也是三级结构，但术语与传统的关系模型术语有所不同。在 SQL 中，关系模式称为基本表，存储模式称为“存储文件”，子模式称为“视图”，元组称为“行”，属性称为“列”。

18．实现嵌入式 SQL 需要解决的关键技术问题是什么？

A、嵌入式 SQL 语言与宿主语言的识别问题。在嵌入式 SQL 语句前加“EXEC SQL”，在嵌入式 SQL 语句后加“END_EXEC“

B、数据交换。共享变量和游标来实现两种语言的匹配问题。

第六章节 数据库管理

1. 什么是封锁机制？

封锁是实现并发控制的一个非常重要的技术。所谓封锁就是事务 T 在对某个数据对象如表、记录等操作之前，先向系统发出请求，对其加锁。加锁后事务 T 就对该数据对象有了一定的控制，在 T 释放它的锁之前，其他的事务不能更新此数据对象。

2. 什么是数据库的安全性？什么是数据的完整性？两者有什么区别和联系？

数据库的安全性是指保护数据库，防止不合法、未经授权的使用，以免数据的泄露、非法更改和破坏。

数据库的完整性是指避免非法的不合语义的错误数据的输入和输出，造成无效操作和错误结果。

数据库的完整性是指尽可能避免无意滥用；数据库的安全性是指尽可能避免恶意滥用数据库数据。当然完整性和安全性是密切相关的。特别是从系统实现方法来看，某一种机制常常既可以用于安全性保护也可用于完整性保护。

3. 什么叫封锁粒度？封锁粒度的大小与并发性、系统开销之间的关系如何？

封锁的数据目标可以是这样一些逻辑单元：数据库、表、记录、字段等，封锁数据目标的大小叫封锁粒度。封锁的粒度小，并发度高，但封锁机构复杂，系统开销大。反之，封锁的粒度大，并发度小，但封锁机构简单，系统开销小。

4. 简述数据库的典型恢复策略。

数据库的恢复，意味着要把数据库恢复到最近一次故障前的一致性状态。典型的数据恢复策略为：

A、周期性地对整个数据库进行复制或转储、建立日志文件。

B、一旦数据库发生故障，则分二种情况处理：

a)如果数据库遇到灾难性故障，就必须装入最近一次拷贝的数据库备份到新的磁盘，然后利用日志库执行“重做”已提交的事务，把数据库恢复到故障前的状态。

b)如果数据库未遭到物理性破坏，只是某些数据不正确，只需利用日志库“撤销”所有不可靠的修改，再利用日志库执行“重做”已提交的、但对数据库的更新可能还留在内存缓冲区的事务，就可以把数据库恢复到正确的状态。

5. 试解释权限的转授和回收。

用户对自己建立的基本表和视图拥有全部的操作权限，而且可用 GRANT 语句把其中的部分或全部的权限授予别的用户。被授予的用户如果有“继续授权”(WITH GRANT OPTION 子句)的许可，还可以把获得的权限对其他用户继续授权。

可用 REVOKE 语句收回所有授予出去的权限。

第七章节 SQL Server 简介及应用

1、简述 SQL Server 2000 的 4 个版本

企业版：作为生产数据库服务器使用。

标准版：作为小工作组或部门的数据库服务器使用。

个人版：供移动的用户使用，这些用户有时从网络上断开，但所运行的应用程序需要 SQL Server 数据存储。

开发版：供程序员用来开发将 SQL Server 2000 用作数据存储的应用程序。

2、简述 SQL Server 2000 的企业管理器

企业管理器是用户管理 SQL Server 的主要管理工具和图形界面。企业管理器可以完成几乎所有的管理工作，如管理登录账号、数据库用户的权限，创建和管理数据库、创建和管理表、视图、存储过程等。在企业管理器中，用户以图形化的方法来进行操作，企业管理器向 SQL Server 发送 T－SQL 命令，同时它也检查 SQL Server ，显示相关的数据和配置信息以便查看。

3、简述存储过程及其优点

存储过程是存放在服务器上的预先编译好的 SQL 语句组并在服务器端运行。存储过程在第一次执行时进行语法检查和编译，编译好的存储过程在高速缓存中用于后续调用，这样就可以使存储过程执行时更加迅速、高效。

存储过程的优点是提供了安全机制、改进了执行性能、减少了网络流量和增强了代码的重用性。

4、简述 SQL Server 触发器

触发器是 SQL Server 提供给程序员和数据分析员来保证数据完整性的一种方法，它是与表事件相关的特殊的存储过程。触发器不能被直接执行，它只能被表上的 Insert 、Update 、Delete 事件所触发。

5、什么是角色？其作用是什么？

在数据库中，把组织在一起的一组具有相同权限的用户称为角色。有了角色就不用直接管理每个具体的数据库用户的权限，而只需将数据库用户放置到合适的角色即可。当工作发生变化时，只要更改角色的权限，无需更改角色中的成员。

第八章节 PowerBuilder 9.0 简介及应用

1. 简述 PB9.0 的特点。

PB 作为基于客户端/服务器应用的快速数据库前端开发工具，它的特点可以概括变如下几点：

A、采用面向对象的编程方法和事件驱动的工作原理。

B、支持跨平台开发，具有开放的数据库连接系统。

C、无须记住各种语句格式即可轻松编写语句，PB 9.0 提供了几乎所有常用的编程语句模板，使用它可以轻松快速插入各种语句。

D、人性化设计，为用户提供方便快捷的语句块剪切板工具，大大提高了程序员的编程效率。

E、独特的库文件画板和层次清晰的系统资源树。

F、系统提供 11 种不同“显示风格”的数据窗口，以满足不同的用户需求。

G、提供规范化的 Workspace (工作空间)对象，让程序员更方便地管理创建的应用对象。

H、还实现对 XML、JSP、.NET 以及 WEB Service 的支持，不仅可以满足企业级应用的需要，而且可以实现对所持设备的应用开发。

2. 简述 PB 开发空间的三个层次。

开发空间有三个层次：工作空间、目标、库文件。

工作空间是开发应用系统的“容器”，一次只能打开一个工作空间，对应扩展名为“. pbw”

目标用于描述加入到工作空间的应用。一个工作空间中可以建立多个目标，一个目标对应一个“.pbt”文件。在一个空间中，可以在多个目标中同时打开多个对象进行编辑，因此，可以同时开发多个应用。

每个目标同时对应一个或多个扩展名为“.pbl”的库文件。

3. 简述 PB 的两组工具栏。

PowerBar：主控菜单工具栏。可以完成很多操作，如：生成新的对象和应用程序，也可以打开存在的对象或调试运行应用程序等。

PainterBar：操作画板组件工具栏。它能够完成当前画板中的任务。

4. 简述 PB 的数据库画板的功能。

PB 通过数据库画板实现了用统一的界面来处理不同数据库，与数据库相关的任务绝大部分可以在数据库画板中完成，实现了方便地对数据库进行管理和维护。

5. 简述 PB 的数据窗口对象。

数据窗口是 PB 的一大特色，PB 作为一种数据库前端开发工具，操作的核心是数据库中的数据。而数据窗口是一个对象，它包含了对数据库中的数据进行特定操作的信息。只要定义好一个数据窗口对象，以后就可以在多个应用程序中使用这个数据窗口对象。

数据窗口对象是封装了对数据库中数据操作的对象，它极大地方便了应用程序对数据库的使用。

6. 简述 PB 连接数据库的步骤。

PB 在访问数据库之前，必须首先与要使用的数据库建立连接。PB 9.0 连接数据库有三个步骤：

(1) 建立 ODBC 数据源；

(2) 建立数据库描述文件；

(3) 用“Connect ”命令连接数据库。

第九章节 数据库技术的发展

1、简述传统数据库的局限性？

一般 来说，人们把第一代（网状、层次）数据库和第二代（关系）数据库称为传统数据库。传统数据库相对于文件系统而言，具有数据结构化、冗余度小、共享程度高、独立性好的特点。

在传统的数据库技术中，ER 模型里的多值属性都要设法转换成单属性或一个弱实体。另外，关系模型中的属性值都是原子值（即 1NF 性质），不允许是元组值或关系值。这就限制了关系模型的适用范围。也就是传统数据库的数据结构比较简单，不能支持复合数据类型和嵌套、递归等数据结构。

2、与 ER 图相比，对象联系图有哪些修改和扩充？

A、实体类型改称为“对象类型”，用椭圆表示。

B、小圆圈表示属性，并且是基本数据类型。对象类型与属性之间联系可以是单箭头（表示单值）或双箭头（表示多值）。

C、对象类型之间的连线表示指针方式的引用，连线也可以是单箭头（表示单值，即元组）或双箭头（表示多值，即集合）。

D、子类与超类的联系图用双线箭头表示。

3、面向对象技术中的数据类型系统是由哪几个部分组成的？

由基本类型、复合类型和引用类型三部分组成。

基本数据类型是指整型、浮点型、字符、字符串、布尔型和枚举类型。

复合类型有行类型、数组类型、列表类型、包类型和集合类型五种。

引用类型相当于程序设计中指针的概念，可以避免数据结构的无穷嵌套问题。

4、简述 UML 的发展历程。

在 20 世纪 90 年代中期，Booch 、 Rumbaugh 和 Jacobson 三位专家设计了一个标准的建立模型语言。他们把这个成果称为“统一建模语言”简写为 UML，并把 UML 版本交给 OMG 组织，经过修改后在 1997 的推出 UML1.0 和 UML1.1 版，确定 UML 为面向对象开发的行业标准语言。

5、简述类图中的基本成分。

类图描述了系统的静态结构，包括类和类间的联系。

类图中的基本成分是类和关联：

(1) 类由三个部分组成：类的名称、类的属性、一些可以用到的关于类的对象的操作。

(2) 关联是对类的实例之间联系的命名，相当于 ER 模型中的联系类型。与关联有关的内容有：

A. 关联元数：与关联有关的类的个数。

B. 关联角色：关联的端部，也就是与关联相连的类。

C. 重复度：指在一个给定的联系中有多少对象参与。

6、简述 ODBC 技术的特点。

ODBC 技术以 C/S 结构为设计基础，它使得应用程序与 DBMS 之间在逻辑上可以分离，使应用程序具有数据库无关性。ODBC 定义了一个 API，每个应用程序利用相同的源代码就可访问不同的数据库系统，存取多个数据库中的数据。与嵌入式 SQL 相比，ODBC 一个最显著的优点是用它生成的应用程序与数据库或数据库引擎无关。

ODBC 使应用程序具良好的互用性和可移植性，并且具备同时访问多种 DBS 的能力，从而克服了传统数据库应用程序的缺陷。

7、ODBC 的体系结构由哪几部分构成，并简述各部分的功能。

ODBC 是一个分层体系结构，其各部分的结构和功能如下：

A. ODBC 数据库应用程序：用宿主语言和 ODBC 函数编写的应用程序用于访问数据库。

B. 驱动程序管理器：为应用程序加载、调用和卸载 DB 驱动程序。

C. DB 驱动程序：处理 ODBC 函数，向数据源提交用户请求执行 SQL 语句。

D. 数据源：是 DB 驱动程序与 DBS 之间连接的命名。

8、简述驱动程序管理器及其主要功能。

驱动程序管理器是一个动态链接库（DLL），用于连接各种 DBS 的 DBMS 驱动程序（如 Oracle 、 FoxPro 等驱动程序），管理应用程序和 DBMS 驱动程序之间的交互作用（通信）。

驱动程序管理器的主要功能如下：

A. 为应用程序加载 DBMS 驱动程序；

B. 检查 ODBC 调用参数的合法性和记录 ODBC 函数的调用；

C. 为不同驱动程序的 ODBC 函数提供单一的入口；

D. 调用正确的 DBMS 驱动程序；

E. 提供驱动程序信息。

9、简述 SQL/CLI

在介绍 SQL/CLI 之前，我们有必要简述一下 API 的概念。API 是一个公共的、与数据库无关的应用程序设计接口，它为开发者提供单一的编程接口，这样同一个应用程序就可以访问不同的数据库服务器。

API 是微软公司 ODBC 技术的术语，是非 SQL 名称。在 SQL 标准中，称为调用层接口，即 CLI。SQL/CLI 是 ODBC 技术的延续，它定义了一套可以从宿主语言调用的公共函数，每一个函数具有一个名称和一个要求的算法。 使用 SQL/CLI 时，函数调用 SQL 语句将作为字符串参数动态地创建和传递。