第一章

第一题：

第一小问：DB，DBMS，DBS 和 IS 的关系

相关定义：

1、DB 是 database 的缩写也就是数据库，数据库是存储数据的一个集合，数据库中通常使用数据表等组成，而数据表是数据的字段和数据的值等信息组成。

2、DBS 是 Database System 的缩写也就是数据库系统，数据库系统是指在计算机系统中引入数据库后的系统。一般由数据库、数据库管理系统（及其开发工具）、应用系统、数据库管理员和用户构成。应当指出的是，数据库的建立、使用和维护等工作只靠一个 DBMS 远远不够，还要有专门的人员来完成，这些人被称为数据库管理员。

3、DBMS 是 Database Management System 的缩写，它是操作数据库和管理数据库的一个系统，比如 mysql、sqlserver 等都是属于数据库管理软件，人们通过这些系统或者工具来管理数据库内的数据。

4、信息系统（Information system）是由计算机硬件、网络和通信设备、计算机软件、信息资源、信息用户和规章制度组成的以处理信息流为目的的人机一体化系统。主要有五个基本功能，即对信息的输入、存储、处理、输出和控制。

关系

1、DBMS 和 DB 的关系：DBMS 数据库管理系统(databasemanagementsystem)是一种操纵和管理数据库的大型软件，是用于建立、使用和维护数据库（DB）。它对数据库进行统一的管理和控制，以保证数据库的安全性和完整性。用户通过 DBMS 访问数据库（DB）中的数据。

2、DBS 和 DB 的关系：数据库系统 DBS（Data Base System，简称 DBS）是一个实际可运行的存储、维护和应用系统提供数据的软件系统，是存储介质、处理对象和管理系统的集合体。它通常由软件、数据库（DB）和数据管理员组成。

3、IS 与 DB 的关系：信息系统是对提供服务的计算机应用，需要数据的存储与处理，数据库是信息系统实现的基石。

第二小问：信息模型与数据模型的区别

信息模型是现实世界到机器世界的一个中间层次，是反映所研究的信息的实体集及实体集之间联系的一个抽象的模型，是易于人理解的，不依赖具体的计算机系统又不受某一 DBMS 所左右的模型。

数据模型是描述数据特征及数据之间联系的模型，它是在信息模型的基础上建立的一个适合计算机表示的数据层的模型。可以看成信息模型到机器中的实现形式。

第三小问：模式、子模式和内模式的不同

子模式又称外模式，对应于用户级。它是某个或某几个用户所看到的数据库的数据视图，是与某一应用有关的数据的逻辑表示。子模式是从模式导出的一个子集，包含模式中允许特定用户使用的那部分数据。用户可以通过外模式描述语言来描述、定义对应于用户的数据记录(外模式)，也可以利用数据操纵语言(DML)对这些数据记录进行。子模式反映了数据库的用户观。

内模式又称存储模式，对应于物理级，它是数据库中全体数据的内部表示或底层描述，是数据库最低一级的逻辑描述，它描述了数据在存储介质上的存储方式翱物理结构，对应着实际存储在外存储介质上的数据库。内模式由内模式描述语言来描述、定义，它是数据库的存储观。

模式又称概念模式或逻辑模式，对应于概念级。它是由数据库设计者综合所有用户的数据，按照统一的观点构造的全局逻辑结构，是对数据库中全部数据的逻辑结构和特征的总体描述，是所有用户的公共数据视图(全局视图)。它是由数据库管理系统提供的数据模式描述语言(DDL)来描述、定义的，体现、反映了数据库系统的整体观。

用户级对应外模式，概念级对应模式，物理级对应内模式

详细课本：P18-P19 页，也可以看看 P27 的本章小结部分关于 3 级模式结构的描述。

第二题

人工管理：程序员在程序中不仅要规定数据的逻辑结构，还要设计其物理结构，包括存储结构、存取方法、输入输出方式等。当数据的物理组织或存储设备改变时，用户程序就必须重新编制。由于数据的组织面向应用，不同的计算程序之间不能共享数据，使得不同的应用之间存在大量的重复数据，很难维护应用程序之

间数据的一致性。

这一阶段的主要特征可归纳为如下几点：

* 计算机中没有支持数据管理的软件。

* 数据组织面向应用，数据不能共享，数据重复。

在**文件系统**阶段，数据以文件为单位存储在外存，且由操作系统统一管理。操作系统为用户使用文件提供了友好界面。文件的逻辑结构与物理结构脱钩，程序和数据分离，使数据与程序有了一定的独立性。用户的程序与数据可分别存放在外存储器上，各个应用程序可以共享一组数据，实现了以文件为单位的数据共享。但由于数据的组织仍然是面向程序，所以存在大量的数据冗余。而且数据的逻辑结构不能方便地修改和扩充，数据逻辑结构的每一点微小改变都会影响到应用程序。由于文件之间互相独立，因而它们不能反映现实世界中事物之间的联系，操作系统不负责维护文件之间的联系信息。如果文件之间有内容上的联系，那也只能由应用程序去处理。

特点：

数据的管理者：文件系统

数据面向的对象：某一应用程序

数据的共享程度：共享性差,冗余度大

数据的独立性：独立性差

数据的结构化：记录内有结构，整体无结构

数据控制能力：应用程序自己控制

数据库系统阶段:特点

数据的管理者：数据库管理系统

数据面向的对象：整个应用系统

数据的共享程度：共享性高，冗余度小

数据的独立性：具有高度的物理独立性和逻辑独立性

数据的结构化：整体结构化，用数据模型描述

数据控制能力：由数据库管理系统提供数据安全性、完整性、并发控制和恢复能力

第三题

数据冗余是指数据之间的重复，也可以说是同一数据存储在不同数据文件中的现象。

在文件管理系统中数据被组织在一个个独立的数据文件中，每个文件都有完整的体系结构，对数据的操作是按文件名访问的，数据文件之间没有联系，数据文件是面向应用程序的。每个应用都拥有并使用自己的数据文件，各数据文件中难免有许多数据相互重复，数据的冗余度比较大。

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，文件内部具有完整的结构，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，因而尽可能地避免了数据的重复存储，减少和控制了数据的冗余。

第四题

数据模型是数据库中用来对现实世界进行抽象的工具，是数据库中用于提供信息表示和操作手段的形式架构。一般地讲，数据模型是严格定义的概念的集合。这些概念精确描述了系统的静态特性，动态特性和完整性约束条件。因此数据模型通常由数据结构，数据操作和完整性约束三部分组成

（1）数据结构

是研究的对象类型的集合，是对系统静态特性的描述。

（2）数据操作

是指对数据库中各种对象（型）的实例（值）允许进行的操作的集合，包括操作及由关的操作规则，是对系统动态特性的描述。

（3）数据的约束条件

是一组完整性规则的集合。完整性规则是给定的数据模型中数据及其联系所具有的制约和依存规则，用以限定符合数据模型的数据库状态及状态的变化，以保证数据的正确、有效相容。解析数据模型是数据库系统中重要的概念之一。要通过

学习真正掌握数据模型的概念和作用。数据模型是数据库系统的基础。任何一个DBMS 都以某一个数据模型为基础，或者说支持某一个数据模型。数据库系统中，模型有不同的层次。根据模型应用的不同目的，可以将模型分成两类或者说两个层次：一类是概念模型，是按用户的观点来对数据和信息建模，用于信息世界的建模，强调语义表达能力，概念简单清晰，另一类是数据模型，是按照计算机系统的观点对数据进行建模，用于机器世界，人们可以用它定义，操纵数据中的数据，一般需要严格的形式化定义个一组严格定义了语法和语义的语言，并有一些规定和限制，便于在机器上实现。

第五题

划分：数据模型是现实世界中各种实体之间存在着联系的客观反映，是用记录描述实体信息的基本结构，它要求实体和记录一一对应；同一记录类型描述同一类实体且必须是同质的。目前应用在数据库技术中的模型有关系、网状和层次模型，它们是依据描述实体与实体之间联系的不同方式来划分的；用二维表格来表示实体和实体之间联系的模型叫做关系模型；用图结构来表示实体和实体之间联系的模型叫做网状模型；用树结构来表示实体和实体之间联系的模型叫做层次模型。

优缺点：

层次模型

优点是：结构清晰，表示各结点之间的联系简单；容易表示如“家族关系”等现实世界的层次结构的事物及其之间的联系。

缺点是：不能表示两个以上实体型之间的复杂联系和实体型之间的多对多联系；严格的层次顺序使数据插入和删除操作变得复杂，如父结点的删除导致子结点的删除。

网状模型

优点是：能够表示实体之间的多种复杂联系。

缺点是：网状模型比较复杂，需要程序员熟悉数据库的逻辑结构；在重新组织数据库时容易失去数据独立性。

关系模型

优点是：使用表的概念，简单直观；直接表示实体之间的多对多联系；具有更好

的数据独立性；具有坚实的理论基础。

缺点是：关系模型的联结等操作开销较大，需要较高性能计算机的支持。

第六题

（1）外模式

外模式又称子模式，对应于用户级。它是某个或某几个用户所看到的数据库的数据视图，是与某一应用有关的数据的逻辑表示。外模式是从模式导出的一个子集，包含模式中允许特定用户使用的那部分数据。用户可以通过外模式描述语言来描述、定义对应于用户的数据记录(外模式)，也可以利用数据操纵语言(DML)对这些数据记录进行。外模式反映了数据库的用户观。

（2）内模式

内模式又称存储模式，对应于物理级，它是数据库中全体数据的内部表示或底层描述，是数据库最低一级的逻辑描述，它描述了数据在存储介质上的存储方式翱物理结构，对应着实际存储在外存储介质上的数据库。内模式由内模式描述语言来描述、定义，它是数据库的存储观。

（3）模式.

模式又称概念模式或逻辑模式，对应于概念级。它是由数据库设计者综合所有用户的数据，按照统一的观点构造的全局逻辑结构，是对数据库中全部数据的逻辑结构和特征的总体描述，是所有用户的公共数据视图(全局视图)。它是由数据库管理系统提供的数据模式描述语言(DDL)来描述、定义的，体现、反映了数据库系统的整体观。

三级模式间的映射

数据库的三级模式是数据库在三个级别 (层次)上的抽象，使用户能够逻辑地、抽象地处理数据而不必关心数据在计算机中的物理表示和存储。实际上，对于一个数据库系统而言一有物理级数据库是客观存在的，它是进行数据库操作的基础，概念级数据库中不过是物理数据库的一种逻辑的、抽象的描述(即模式)，用户级数据库则是用户与数据库的接口，它是概念级数据库的一个子集(外模式)。

用户应用程序根据外模式进行数据操作，通过外模式一模式映射，定义和建立某个外模式与模式间的对应关系，将外模式与模式联系起来，当模式发生改变时，只要改变其映射，就可以使外模式保持不变，对应的应用程序也可保持不变；另一方面，通过模式一内模式映射，定义建立数据的逻辑结构(模式)与存储结构(内模式)间的对应关系，当数据的存储结构发生变化时，只需改变模式一内模式映射，就能保持模式不变，因此应用程序也可以保持不变。

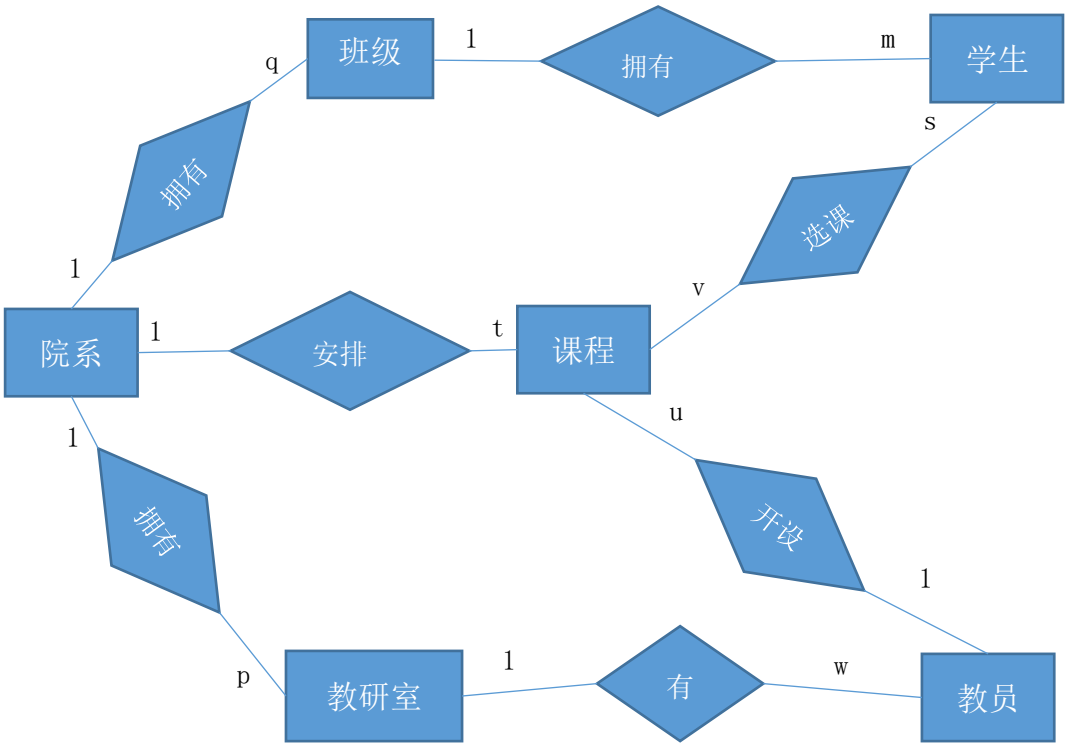
第 7 题

见 P25-P26 页

第 8 题

参考第九题

第 9 题



第 10 题

第二章

第一题

关系实际上就是关系模式在某一时刻的状态或内容。也就是说，关系模式是型，关系是它的值。关系模式是静态的、稳定的，而关系是动态的、随时间不断变化的，因为关系操作在不断地更新着数据库中的数据。但在实际当中，常常把关系模式和关系统称为关系，读者可以从上下文中加以区别。 关系模型是 1970 年由 E.F.Codd 提出的,是关系模式设计的理论。能够在一定程度上支持关系模型的数据库管理系统是关系系统，它支持关系数据库。关系数据库是按照关系模型建立的数据库。由此可见，没有关系模型，便没有关系数据库和关系系统。

第四题

（1） RUS

X	Y
A	d
b	a
c	c
d	a
d	c

（2） R∩S

X	Y
b	a

（3） RXS

R_x	R_y	S_x	S_y
a	d	d	a
a	d	b	a
a	d	d	c
b	a	d	a

b	a	b	a
b	a	d	c
c	c	d	a
c	c	b	a
c	c	d	C

（4） U÷V

X	Y
a	b
c	a

（5） R 与 T 的外部并

X	Y	Z
a	d	NULL
b	a	NULL
c	c	NULL
NULL	c	d
NULL	b	b
NULL	b	e

（6） U 与 T 的外连接、左外连接及右外连接

X	Y	Z	W
a	b	c	d
a	b	e	f
c	a	c	d
NULL	b	b	NULL
NULL	c	d	NULL

X	Y	Z	W
a	b	c	d
a	b	e	f
c	a	c	d

X	Y	Z	W
NULL	b	b	NULL
a	b	e	f
NULL	c	d	NULL

（7） T ⋈ S、S 半连接 T

Y	Z	X
c	d	d

X	Y
d	c

第五题

（1） 查询“张景林”老师所授课程号和课程名。

$$\Pi_{cno,cname}(\sigma_{teacher=\text{“张景林”}}(C))$$

（2） 查询选修课程名为“C 语言”或者“数据库”的学生学号。

$$\Pi_{sno}(\sigma_{cname=\text{数据库”Vcname=“C 语言“}}(C) \bowtie SC)$$

（3） 查询“高晓灵“同学所选修课程的课程号及课程名

$$\Pi_{cno}(\sigma_{sname=\text{“高晓灵“}}(S) \bowtie SC) \bowtie \Pi_{cno,cname}(C)$$

（4） 查询至少选修两门课程的学生学号

$$\Pi_{sno}(\sigma_{sno=sno' \wedge \ cno!=cno'}(\Pi_{sno,cno}(SC)X\Pi_{sno,cno}(SC)))$$

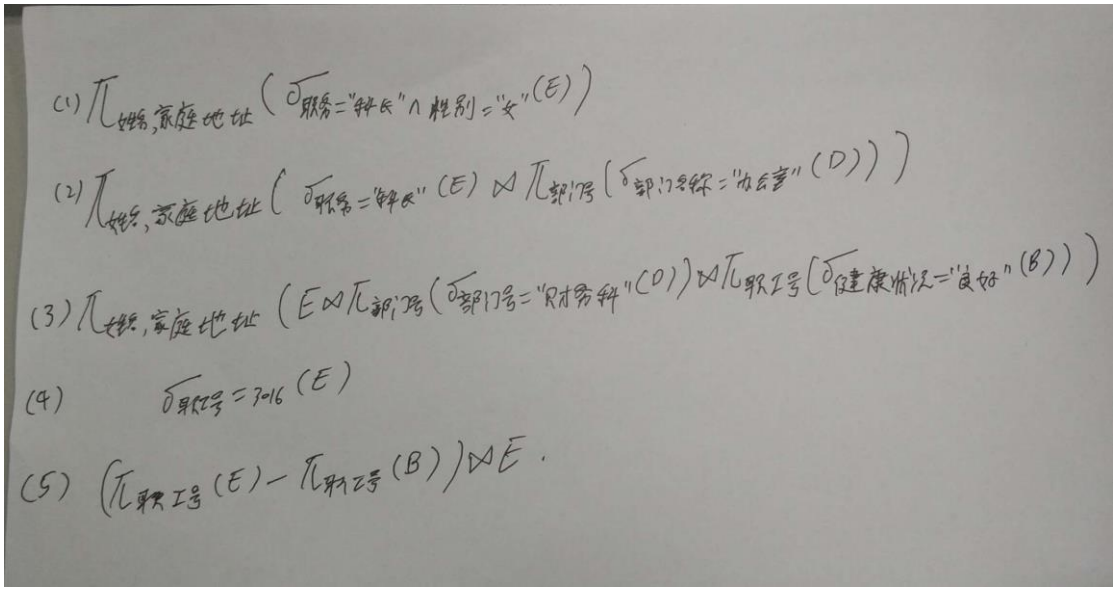
（5） 查询全部学生都选修课程的课程号和课程名

$$\Pi_{cno,sno}(SC) \div \Pi_{sno}(S) \bowtie \Pi_{cno,cname}(C))$$

（6） 查询至少选修“张景林”老师所授全部课程的学生姓名

$$\Pi_{sname}(S \bowtie (\Pi_{cno,sno}(SC) \div \Pi_{cno}(\sigma_{teacher="张景林"}(C)))$$

第六题



第七题

用户的查询通过相应的查询语句提交给 DBMS 执行，该查询首先要被 DBMS 解析，然后转换成内部表示，从查询的多个执行策略中进行合理选择的过程就是“查询处理过程中的优化”，简称为查询优化。

提高数据库性能，快速响应用户查询操作。

一般策略：

（1） 尽可能先做选择，投影运算。

（2） 合并笛卡尔积与其后的选择运算为连接运算

（3） 把投影运算和选择运算同步进行

（4） 让投影运算与其前后的其他运算同时进行

（5） 找出公共子表达式

1．合理使用索引

索引是数据库中重要的数据结构，它的根本目的就是为了提高查询效率。现在大多数的数据库产品都采用 IBM 最先提出的 ISAM 索引结构。索引的使用要恰到好处，其使用原则如下：

●在经常进行连接，但是没有指定为外键的列上建立索引，而不经常连接的字段则由优化器自动生成索引。

●在频繁进行排序或分组（即进行 group by 或 order by 操作）的列上建立索引。

●在条件表达式中经常用到的不同值较多的列上建立检索，在不同值少的列上不要建立索引。比如在雇员表的“性别”列上只有“男”与“女”两个不同值，因此就无必要建立索引。如果建立索引不但不会提高查询效率，反而会严重降低更新速度。

●如果待排序的列有多个，可以在这些列上建立复合索引（compound index）。

●使用系统工具。如 Informix 数据库有一个 tbcheck 工具，可以在可疑的索引上进行检查。在一些数据库服务器上，索引可能失效或者因为频繁操作而使得读取效率降低，如果一个使用索引的查询不明不白地慢下来，可以试着用 tbcheck 工具检查索引的完整性，必要时进行修复。另外，当数据库表更新大量数据后，删除并重建索引可以提高查询速度。

2．避免或简化排序

应当简化或避免对大型表进行重复的排序。当能够利用索引自动以适当的次序产生输出时，优化器就避免了排序的步骤。以下是一些影响因素：

●索引中不包括一个或几个待排序的列；

●group by 或 order by 子句中列的次序与索引的次序不一样；

●排序的列来自不同的表。

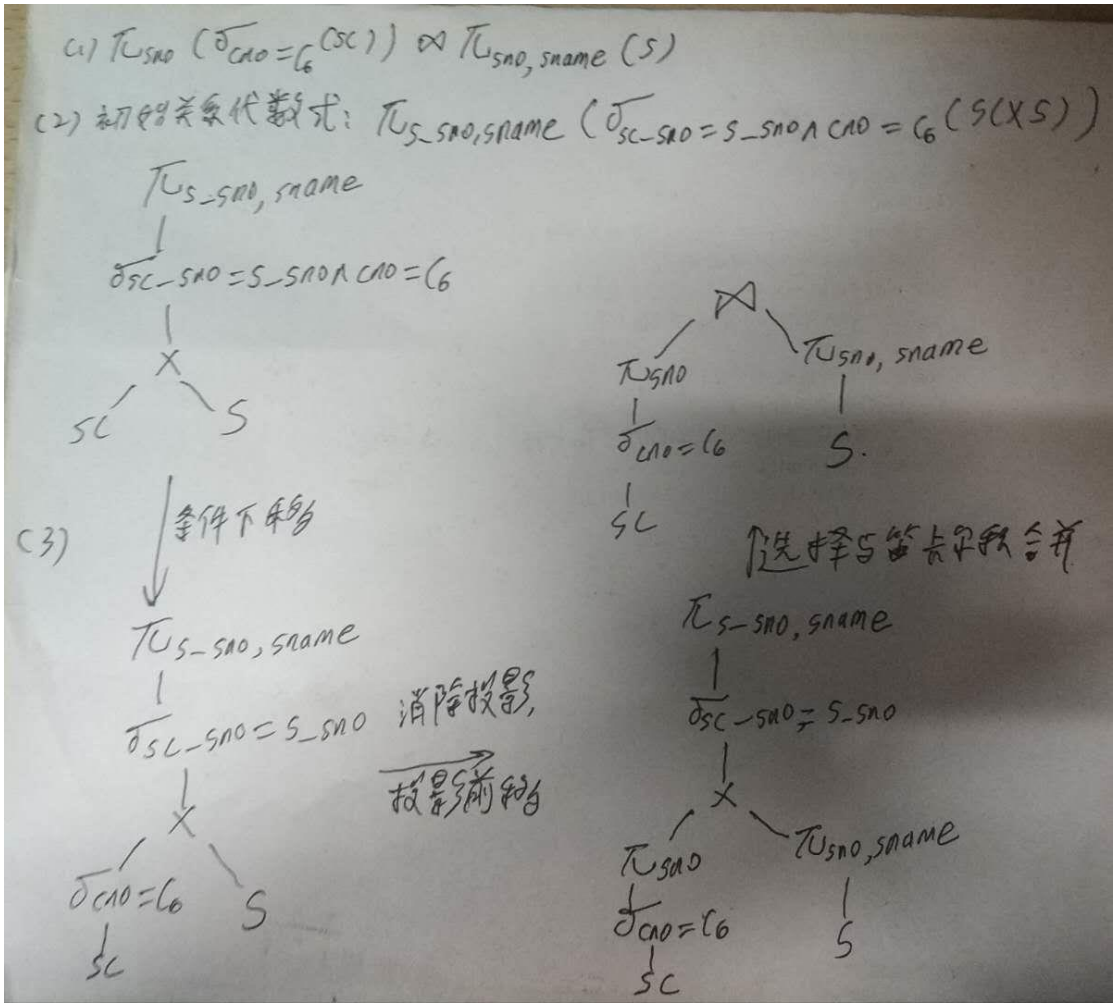
为了避免不必要的排序，就要正确地增建索引，合理地合并数据库表（尽管有时可能影响表的规范化，但相对于效率的提高是值得的）。如果排序不可避免，那么应当试图简化它，如缩小排序的列的范围等。

3．消除对大型表行数据的顺序存取

在嵌套查询中，对表的顺序存取对查询效率可能产生致命的影响。比如采用顺序存取策略，一个嵌套 3 层的查询，如果每层都查询 1000 行，那么这个查询就要

查询 10 亿行数据。避免这种情况的主要方法就是对连接的列进行索引。例如，两个表：学生表（学号、姓名、年龄……）和选课表（学号、课程号、成绩）。如果两个表要做连接，就要在“学号”这个连接字段上建立索引

第九题



第三章

第一题

数据库模式：是数据库中全体数据的逻辑结构和特征的描述，是所有用户的公共数据视图

包括子模式，模式，内模式。

子模式：是数据库用户(包括应用程序员和最终用户)能够看见和使用的局部数据的逻辑结构和特征的描述，是数据库用户的数据视图，是与某一应用有关的数据的逻辑表示。

模式：是数据库系统中全局数据逻辑结构的描述，是全体用户(应用)公共数据视图，此种描述是一种抽象的描述，它不涉及具体的硬件环境与平台，也与具体的软件环境无关。

内模式：它是数据物理结构和存储方式的描述，是数据在数据库内部的表示方式。

基本表：由行(记录)和列(字段)构成，行中的记录就是数据，列中对应字段，是行和列的集合。

视图：视图是从一个或几个基本表（或视图）中导出的虚拟的表。

相关子查询：相关子查询指的是查询中再查询,通常是以一个查询作为条件来供另一个查询使用，非相关子查询是独立于外部查询的子查询，子查询总共执行一次，执行完毕后将值传递归外部查询。相关子查询的执行依赖于外部查询的数据，外部查询执行一行，子查询就执行一次。

其他自行查找资料。

第四题

in 是子查询为驱动表，外面的表为被驱动表，故适用于子查询结果集小而外面的表结果集大的情况。

exists 是外面的表位驱动表，子查询里面的表为被驱动表，故适用于外面的表结果集小而子查询结果集大的情况。

exists 一般都是关联子查询，in 则一般都是非关联子查询。

对于关联子查询，必须先执行外层查询，接着对所有通过过滤条件的记录，执行内层查询。外层查询和内层查询相互依赖，因为外层查询会把数据传递给内层查

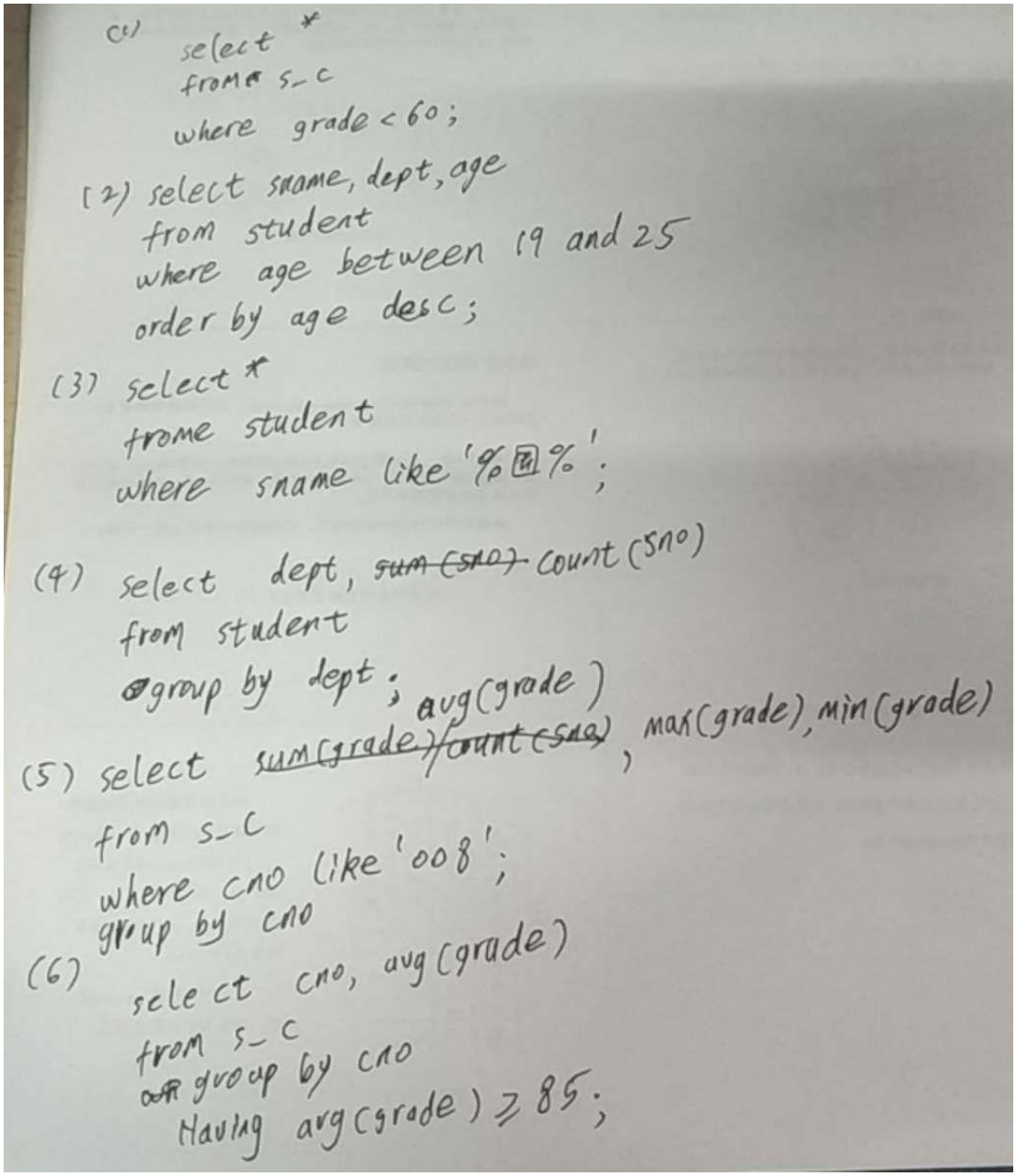
询。

非关联子查询则必须先完成内层查询之后，外层查询才能介入。

some 和 any 一个意思

is null， is not null。没有设置不能为空，就可以使用空

第六题



第 7 题

（1） 略

（2） 略

（3） 略

（4） 略

（5） select sno, sname

from S

where exists (select * from SC where cno=c5);

（6） 略

第八题

（1）（2）（3）（4）见第二章习题六，sql 语句略

（5）update B set 健康状况=‘一般‘ where 职工号 = ‘3016’;

（6）create view E_cha(职工号,姓名，性别，职务，家庭地址，部门号)
AS select E.* from E join B where B.职工号=E.职工号 and 健康状况 = ‘差’;

第九题

sql 表连接分成**外连接**、**内连接**和**交叉连接**。

外连接包括三种,分别是左外连接、右外连接、全外连接。

对应的 sql 关键字:LEFT/RIGHT/FULL OUTER JOIN,通常我们都省略 OUTER 关键字,写成 LEFT/RIGHT/FULL JOIN。

在左、右外连接中都会以一种表为基表，基表的所有行、列都会显示，外表如果和条件不匹配则所有的外表列值都为 NULL。

全外连接则所有表的行、列都会显示，条件不匹配的值皆为 NULL。

内连接是用比较运算符比较要连接的列的值的连接,不匹配的行不会被显示。sql 关键字 JOIN 或者 INNER JOIN，通常我们写成 JOIN

交叉连接：没有 where 条件的交叉连接将产生连接表所涉及的笛卡尔积。

第 10 题

把对视图的查询转化为对基本表的查询称为视图的消解

1、如果视图有一个基本关系的简单查询生成，而且它还包含了基本关系中的主关键字或是候选关键字，则可以通过这个视图进行更新操作。

2，如果更新视图的字段来自字段表达式或者是常数，则不允许对视图进行 Insert、update 操作（但可以进行 delete 操作）。因为，假设用字段表达式或者是常数对视图进行了更新，那么视图的相关行或列就与基本表中的相关行或列不一致，或者在基本表中根本找不到，对视图的更新实质上是对基本表的更新，因此和基本表的不一致的视图是毫无意义的。（不允许 Insert 操作也是同样的道理）

3，如果更新字段来自库函数，则不允许对视图进行 Insert、update 操作（但可以进行 delete 操作）。举个例，如果当前视图只是某个基本表中的一部分，而现在要用 sum 函数对视图进行更新，那么 sum 就会将基本表中满足条件的值都相加，而其实视图中只包含满足条件的一部分，这样就会出现错误。

4，如果视图的生成中涉及到聚集或是分组操作，则不允许通过这个视图进行更新。如更新字段包含 group by，distinct 等函数，这跟 2 的解释是一样的。

5，若视图来自两个及以上的基本表，则不允许更新。这是因为如果要更新这样的视图，那么就会涉及到多个基本表的同时修改，这会导致数据库的存取非常复杂，因此，在现有的关系数据库系统中，只支持来自单个表的视图的更新。

6，在一个不允许更新的视图上定义的视图不允许更新。

第 14 题

静态 SQL：静态 SQL 语句一般用于嵌入式 SQL 应用中，在程序运行前，SQL 语句必须是确定的，例如 SQL 语句中涉及的列名和表名必须是存在的。静态 SQL 语句的编译是在应用程序运行前进行的，编译的结果会存储在数据库内部。而后程序运行时，数据库将直接执行编译好的 SQL 语句，降低运行时的开销。

动态 SQL：动态 SQL 语句是在应用程序运行时被编译和执行的，例如，使用 DB2 的交互式工具 CLP 访问数据库时，用户输入的 SQL 语句是不确定的，因此 SQL 语句只能被动态地编译。动态 SQL 的应用较多，常见的 CLI 和 JDBC 应用程序都使用动态 SQL。

第 15 题

ODBC（Open Database Connectivity）是一组对数据库访问的标准 API，这些 API 通过 SQL 来完成大部分任务，而且它本身也支持 SQL 语言，支持用户发来的 SQL。ODBC 定义了访问数据库 API 的一组规范，这些 API 独立于形色各异的 DBMS 和编程语言。

也就是说，一个基于 ODBC 的应用程序，对数据库的操作不依赖任何 DBMS，不直接与 DBMS 打交道，所有的数据库操作由对应的 DBMS 的 ODBC 驱动程序完成。不论是 SQL Server、Access 还是 Oracle 数据库，均可用 ODBC API 进行访问。

由此可见，ODBC 的最大优点是能以统一的方式处理所有的数据库。

JDBC（JavaDatabase Connectivity）是 Java 与数据库的接口规范，JDBC 定义了一个支持标准 SQL 功能的通用低层 API，它由 Java 语言编写的类和接口组成，旨在让各数据库开发商为 Java 程序员提供标准的数据库 API。

JDBC API 定义了若干 Java 中的类，表示数据库连接、SQL 指令、结果集、数据库元数据等。它允许 Java 程序员发送 SQL 指令并处理结果。

共同点：

（1） JDBC 与 ODBC 都是基于 X/Open 的 SQL 调用级接口；

（2） 从结构上来讲，JDBC 的总体结构类似于 ODBC，都有四个组件：应用程序、驱动程序管理器、驱动程序和数据源，工作原 理亦大体相同；

（3）在内容交互方面，JDBC 保持了 ODBC 的基本特性，也独立于特定数据库. 而且都不是直接与数据库交互，而是通过驱动程序管理器。

区别：

（1）ODBC 几乎能在所有平台上连接几乎所有的数据库。为什么 Java 不使用 ODBC？

答案是：Java 可以使用 ODBC，但最好是以 JDBC-ODBC 桥的形式使用（Java 连接总体分为 Java 直连和 JDBC-ODBC 桥两种形式）。

那为什么还需要 JDBC?

因为 ODBC 不适合直接在 Java 中使用，因为它使用 C 语言接口。从 Java 调用本地 C 代码在安全性、实现、坚固性和程序的自动移植性方面都有许多缺点。从 ODBC C API 到 Java API 的字面翻译是不可取的。例如，Java 没有指针，而 ODBC 却对指针用得很广泛（包括很容易出错的指针"void *"）。

（2）ODBC 比较复杂，而 JDBC 尽量保证简单功能的简便性，同时在必要时允许使用高级功能。如果使用 ODBC，就必须手动地将 ODBC 驱动程序管理器和驱动程序安装在每台客户机上。如果完全用 Java 编写 JDBC 驱动程序则 JDBC 代码在所有 Java 平台上（从网络计算机到大型机）都可以自 动安装、移植并保证安全性。

总之，JDBC 在很大程度上是借鉴了 ODBC 的，从他的基础上发展而来。JDBC 保留了 ODBC 的基本设计特征，因此，熟悉 ODBC 的程序员将发现 JDBC 很容易使用。它们之间最大的区别在于：JDBC 以 Java 风格与优点为基础并进行优化，因此更加易于使用。