数据库冲刺考点

考点一 绪论

一、 基本概念

（1）数据：描述事物的符号记录称为数据。数据的种类有数字、文字、图形、图像、声音等。数据与其语义是不可分的。

（2）数据库：数据库是长期储存在计算机内的、有组织的、可共享的数据集合。

（3）数据库管理系统：DBMS 是位于用户与操作系统之间的一层数据管理软件，用于科学地组织和存储数据、高效地获取数据和维护数据。DBMS 的主要功能：数据定义功能、数据操纵功能、数据库的运行管理功能、数据库的建立和维护功能。

（4）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构成，一般由数据库、数据库管理系统（及其开发工具）、应用系统、数据库管理员构成。

（5）数据库系统的特点：数据结构化；数据共享性高，冗余度低，易扩充；数据独立性高

二、数据管理技术的发展过程

1．数据库管理技术发展的三个阶段：人工管理阶段、文件管理阶段和数据库系统阶段.

2.文件系统与数据库系统区别和联系：

a.数据库系统实现整体数据的结构化，这是数据库的主要特征之一，也是数据库系统与文件系统的本质区别。

b.数据库系统存储数据的方式灵活，可以存储数据库中的某一个数据项，一组数据项，一组记录或一个纪录，而文件系统中数据的存取单位是记录。

c.数据库系统的数据共享性高，易扩充；而文件系统中的文件是为某一特定的应用服务的，系统也不宜扩充。

三、数据模型

1. 定义：数据模型也是一种模型，它是对现实世界的一种 抽象。也就是说，数据模型是用来描述数据、组织数据和对数据进行操作的。

2.组成三要素：数据结构、数据操作和完整性约束。

（1）数据结构：是所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）允许进行的操作集合，包括操作及有关的操作规则，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以及状态的变化，以保证数据的正确、有效、相容。

3.概念模式中常见术语：

（1）实体（Entity）：指客观存在并可相互区别的事物。比如学生、学生的一次选课、学生与系的关系等。

（2）实体型（Entity type）：指用实体名及其属性集合来抽象和刻画同类实体。如部门（部门号、部门名称）就是一个实体型。

（3）实体集（Entity set）： 指同类实体的集合。如全体部门就是一个实体集。

（4）属性（Attribute）：指实体所具有的某一特性。若干属性可以刻画一个实体，例如部门实体可以由部门号、部门名称等属性组成。

（5）码（Key）：指唯一标识实体的属性集。比如部门号是部门实体的码。

（6）域（Domain）：指某一属性的取值范围。如部门名称的域为字符串集合。

（7）实体—联系图（E-R）：是一种用来在数据库设计过程中彼时数据库系统结构的方法，是一种可视化图形方法。

（8） 联系（Relationship）：反映为实体（型）内部的联系和实体（型）之间的联系。

两个实体型之间的联系分为三类：

（1）一对一（1：1）； （2）一对多（1：n）； （3）多对多（m：n）

4．概念模型的表示方法（实体-联系方法）

数据库的总体概念结构可以用 E-R 模型中的 E-R 图来表示。E-R 图由以下基本元素构成：

（1）矩形：代表实体集；　（2）椭圆：代表属性；　（3）菱形：代表实体间的联系集；

（4）线段：将属性与实体集相连或将实体集与联系集相连。除线段以外，每个元素上都标有它所代表的实体、属性或联系。

5．常用的数据库模型：层次、网状、关系模型

层次模型与网状模型的优缺点：

层次数据模型的优点是：1.层次 数据结构比较简单清晰。2.层次数据库的查询效率高。3.层次数据模型提供了良好的完整性支持。缺点主要有：1.现实世界中很多联系是非层次性的，如结点之间具有多对多联系。2.一个结点具有多个双亲等 ，层次模型表示这类联系的方法很笨拙，只能通过引入冗余数据或创建非自然的数据结构来解决。对插入和删除操作的限制比较多，因此应用程序的编写比较复杂。3.查询子女结点必须通过双亲结点。4.由于结构严密，层次命令趋于程序化。可见用层次模型对具有一对多的层次联系的部门描述非常自然，直观容易理解，这是层次数据库的突出优点。

网状模型的优点主要有：1.能够更为直接地描述现实世界，如一个结点可以有多个双亲。结点之间可以有多种上联第。2.具有良好的性能，存取效率较高。

缺点主要有：1.结构比较复杂，而且随着应用环境的扩大，数据库的结构就变得越来越复杂，不利于最终 用户掌握。2.网状模型的 DDL,DML 复杂，并且要嵌入某 一种高级语言中，用户不容易掌握，不容易使用。

6

四、数据库系统结构

1.数据库系统的三级模式结构：外模式、模式和内模式。

模式也称逻辑模式，是数据库中全体数据的逻辑结构和牲的描述，是所有用户的公共数据视图。

外模式也称子模式或用户模式，它是数据库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。

内模式也称存储模式 ，是一个数据库只有一个内模式。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。

2.三级模式之间的二级映像：外模式/模式映像、模式/模式映像.

3.数据库管理系统的主要功能有（1）数据库定义功能；（2）数据组织、存储和管理；（3）数据操纵功能；（4）数据库的事务管理和运行管理；（5）数据库的建立和维护功能等等

DDL：数据定义语言　　DML:数据操纵语言　DCL：数据控制语言

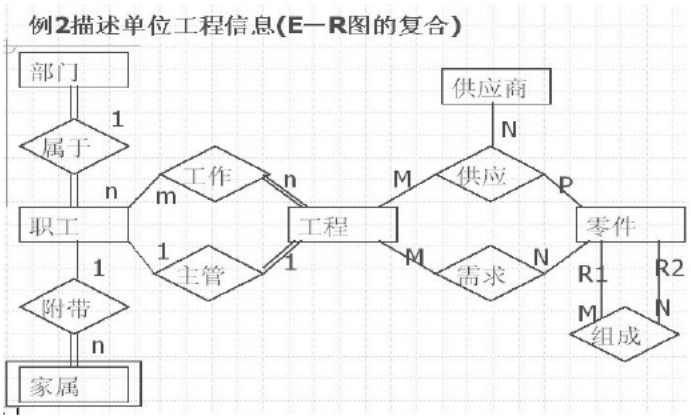
4. 什么叫数据与程序的物理独立性？什么叫数据与程序的逻辑独立性？

数据与程序的物理独立性：当数据库的存储结构改变了（例如采用了更先进的存储结构），数据库管理员修改模式 / 内模式映像，使模式保持不变，当数据库的存储结构改变时，由数据库管理员对模式/内模式映像做相应改变，以保持模式不变，从而应用程序也不必改变，保证了数据与程序的物理独立性。

数据与程序的逻辑独立性：当模式改变时，数据库管理员修改有关的外模式 / 模式映像，使外模式保持不变，应用程序是依据数据的外模式编写的，从而应用程序不必修改，保证了数据与程序的逻辑独立性，简称数据的逻辑独立性

重难点：**画 E—R 图，**实体、联系、实体联系的属性，以及实体间的对应关系

重点掌握 1、工厂物资管理 E-R 图示例：



考点二　关系数据库

一、关系模型的基本概念

1. 关系模型由关系数据结构、关系操作集合、关系完整性约束三部分组成。

2.基本术语：

域：域是一组具有相同数据类型的值的集合。

元组：关系中的每个元素是关系中的元组。

属性：关系也是一个二维表，表的每行对应一个元组，表的每列对应一个域。由于域可以相同，为了加以区分，必须对每列起一个名字，称为属性（ Attribute）。

候选码：若关系中的某一属性组的值能唯一地标识一个元组，则称该属性组为候选码

主码：若一个关系有多个候选码，则选定其中一个为主码（ Primary key）。

外部码：设 F 是基本关系 R 的一个或一组属性，但不是关系 R 的码，如果 F 与基本关系 S 的主码 Ks 相对应，则称 F 是基本关系 R 的外部码（Foreign key），简称外码。

关系模式：关系的描述称为关系模式）。

关系：在域 D1，D2，…，Dn 上笛卡尔积 D1×D2×…×Dn 的子集称为关系。

关系是关系模式在某一时刻的状态或内容。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。

关系数据库：关系数据库也有型和值之分。关系数据库的型也称为关系数据库模式，是对关系数据库的描述，它包括若干域的定义以及在这些域上定义的若干关系模式。关系数据库的值是这些关系模式在某一时刻对应的关系的集合，通常就称为关系数据库。

3. 关系的**三类完整性约束**

（1）实体完整性；（2）参照完整性；（3）用户定义的完整性。

二、关系代数

1. 关系代数是一种抽象的查询语言，它是用关系的运算来表达查询的。

2. 运算符种类：集合运算符（∪ ─ ∩）、专门的关系运算符（× σ ∏ ∞ ÷）、算术比较符（＞≥＜≤＝≠）和逻辑运算符（¬ ∧ ∨）。

3.关系代数的运算按运算符的不同可分为传统的集合运算和专门的关系运算两类。

4.传统的集合运算是二目运算，包括交、并、差和笛卡尔积四种运算。

并（Union）：R∪S={t|t∈R∨t∈S}，t 属于 R 或 t 属于 S。

差（Difference）：R-S={t|t∈R∧t 不属于 S}。

交（Intersection）：R∩S={t|t∈R∧t∈S}。

广义笛卡尔积: r1×r2

专门的关系运算包括选择、投影、连接、除运算等。

1）选择（Selection）

选择运算给出满足给定谓词(条件)的元组。用小写希腊字母σ来表示选择运算，而将谓词写作σ的下标，并在σ后的括号中给出作为参数的关系。记作：

σF(R)={t|t∈R∧F(t)='True'}

例如，我们要找出职员表中所有广东省的员工，则写作：σ籍贯='广东'（职员表）。

通常允许在选择谓词中进行比较，使用的比较运算符是＝、≠、<、≤、>和≥。另外，还可以用连词 and(∧)和 or(∨)将多个谓词合并成一个较大的谓词。

2、投影（Projection）

关系 R 上的投影是从 R 中选择出若干属性列组成新的关系，投影后会取消原有关系中的某些列或玩组。记作： ΠA（R）={t[A]|t∈R}

例如，查询职员所在的姓名和部门：Π姓名，部门（职员表）。

3、连接（Join）

从两个关系的笛卡尔积中选取属性间满足一定条件的元组。常用的有等值连接和自然连接两种。这里要记住，自然连接需取消重复列，其它连接操作只须考虑行。

等值连接与自然连接的区别和联系:

等值连接是从关系 R 与 S 的笛卡尔积中选取 A、B 属性值相等的那些元组。

自然连接是一种特殊的等值连接，它要求两个关系到中进行比较的分量必须是相同的属性组，并且在结果中把重复的属性列去掉。

4. 关系代数的基本运算有哪些？

在八种关系代数运算中：并、差、笛卡尔积、投影、选择五种运算为基本的运算。其他三种运算，即交、连接和除均可用这五种基本运算来表达。

三、关系演算

1.关系演算分为元组关系演算和域关系演算。

考点三　系数据库标准语言 SQL

一、SQL 概貌,特点及其相关基本概念

1.SQL(Structured Query Language)，即结构化查询语言，是关系数据库的标准语言。

2.SQL 包括：数据查询、数据操纵、数据定义和数据控制。

3.SQL 语言的特点：（1）综合统一（2）高度非过程化 （3）面向集合的操作方式 （4）以同一种语法结构提供两种使用方式 （5）语言简洁，易学易用。

4. 基本表是本身独立存在的表，在 SQL 中一个关系就对应一个表。

视图是从一个或几个基本表导出的表。

视图本身不独立存储在数据库中，是一个虚表。即数据库中只存放视图的定义而不存放视图对应的数据，这些数据仍存放在导出视图的基本表中。视图在概念上与基本表等同，用户可如同基本表那样使用视图，可以在视图上再定义视图。

在数据库的三级模式中，外模式对应于视图和部分基本表，模式对应于基本表，内模式对应于存储文件。

二 SQL 数据定义功能

1. SQL 的数据定义功能包括定义表、定义视图和定义索引。

SQL 语言使用 CREATE TABLE 语句建立基本表，ALTER TABLE 语句修改基本表，DROP TABLE 语句删除基本表；使用 CREATE INDEX 语句建立索引，DROP INDEX 语句删除索引；使用 CREATE VIEW 语句建立视图，DROP VIEW 语句删除视图。

三、SQL 数据查询功能

1.SQL 提供了 SELECT 语句进行数据库的查询。有以下几种形式：

(1).无条件查询：获取表中的全部信息

(2).防止选取重复行(用 distinct 消除重复行)和使用别名(可用 as)

(3).条件查询—WHERE 子句，比较运算符(</>/=/!=),逻辑运算符(NOT/ AND/ OR)

特殊运算符(between / in / like / % / _ / null)

(4).查询结果排序 Order by 列名 ASC 升/DESC 降 默认 ASC

(5).信息汇总——特殊函数、group by ，having 子句

特殊函数有：COUNT AVG SUM MIN MAX

一般 COUNT 是记录元组的行数或个数；

在有 group by 时 COUNT*表示记录每个小组的个数而不是全部的个数

2.连接查询

连接查询是关系数据库中最主要的查询，包括广义笛卡尔积、等值连接查询(含自然连接查询)、非等值查询、自身连接查询、外连接查询和符合条件查询等。

3. 嵌套查询

(1).一个 SELECT—FROM—WHERE 语句成为一个查询块。将一个语句块嵌套在另一个查询快的 WHERE 子句或 HAVING 短语中的查询称为嵌套查询。

(2).嵌套查询的使用场合：条件未知和存在依赖数据。

(3).在嵌套查询中，子查询的 SELECT 语句不能有 ORDER BY 子句，ORDER BY 只能对最终查询进行排序。

(4).不关子查询(如带有 IN 的)、相关子查询(如带有比较运算符的子查询)、含 ANY / ALL / EXISTS 谓词的子查询

4.集合查询

集合操作主要包括并操作 UNION、交操作 INTERSECT 和差操作 EXCEPT。

四、数据更新

插入数据 INSERT 、修改数据 UPDATE 、删除数据 DELETE

五、视图的定义和作用

1.视图是从一个或几个基本表（或视图）导出的表。它与基本表不同，是一个虚拟表。数据库中只存放视图的定义，而不存放试图对应的数据。

2.创建视图：CREATE VIEW

3.删除视图：DROP VIEW　　注意：如果所删除的视图还导出了其他视图，则使用 CASCADE 级联删除语句，把该视图和有它导出的视图一起删除。

4.更新视图：INSERT / DELETE / UPDATE

注意：对视图的更新最终都转化成对基本表的更新。

WITH CHECK OPTION 表示对试图进行更新时，保证了所作的更新满足视图定义中的条件，若不满足则拒绝执行该操作。

5.在关系数据库中，并不是所有的视图都可以更新。因为有些视图的更新不能唯一有意义地转换成对相应基本表的更新，例如我们要修改某学生的平均成绩，必须修改各科成绩，而我们无法知道哪些课程成绩的变化导致了平均成绩的变化。

6. 如果视图是从多个基本表使用联接操作导出的，则不允许更新。

如果导出的视图使用了分组和聚合操作，也不允许更新。

如果视图是从单个基本表使用选择和投影操作导出的，并且包括了基本表的主键或某个候选键，则可以执行操作。

常见问题分析：

1、在确定用实体集还是联系集时，一个可采用的原则是什么？

在确定用实体集还是联系集时一个可采用的原则就是：当描述发生在实体间的行为时最好采用联系集。

2 、关系和表是一回事吗？

严格地说，关系和表不一样。关系是一系列域上的笛卡尔积的子集，是一个集合。既然关系是集合就不允许在集合中有重复的元组。按照课件的解释表可以看成是由一行一行的内容组成的一个格式，每个表有多个列，每列有唯一的列名，而每个表也有一个唯一的名字。值得注意的是在关系数据库系统中，表中允许有重复的行存在，即允许有重复的记录。

3、NULL 是表示一个属性值非法吗？

不对，NULL 是一个合法的属性值。它表示该属性的值为空值，空值的含义有两层意思，一是表示属性没有值；二是表示属性的值未知。当属性值未知时还有两种情况，一是值缺失，表示属性有值，且占用一个空格，但是我们目前还没有该值的信息；二是不知道该属性有没有值。

4、 在关系代数和 SQL 中对查询结果的重复行的处理一样吗？

不一样。关系代数的运算结果仍然是一个关系，绝对不允许重复的元组在关系代数的结果中出现。而在 SQL 中，去掉查询结果中的重复行是一件非常耗时的操作，因此允许在 SQL 的查询结果中保留重复的行。可以通过 DISTINCT 选项说明 SQL 的查询结果是否要保留重复的行。

5 、数据库中的视图和关系有什么区别？

视图是虚关系，在 DBMS 中只保存定义视图的 SQL 语句，而不保存视图的具体数据。当在查询中用到视图时，DBMS 会根据其保存的视图的定义（即 SQL 语句）来得到有关的数据。如果在 DBMS 中保存视图的数据，这样的视图称之为实体化视图。而关系在 DBMS 中不仅存储有其定义，而且还存储有实实在在的数据

典型例题：

SQL 查询语句涉及 SQL 语句的主要功能。数据定义（DDL）：create drop alter;数据操纵（DML）:insert update delete，数据控制（DCL）:revoke graint

1 建立一个“学生”表 Student，它由学号 Sno、姓名 Sname、性别 Ssex、年龄 Sage、所在系 Sdept 五个属性组成。其中学号不能为空，值是唯一的，并且姓名取值也唯一。

```
CREATE TABLE Student (Sno   CHAR(5)   NOT NULL   UNIQUE，
        Sname   CHAR(20)   UNIQUE，
        Ssex     CHAR(1) ，
        Sage     INT，
        Sdept    CHAR(15) )；
```

2　建立一个“学生选课”表 SC，它由学号 Sno、课程号 Cno，修课成绩 Grade 组成，其中(Sno, Cno)为主码。

```
CREATE TABLE SC(   Sno CHAR(5) , Cno CHAR(3) ,Grade    int, Primary key (Sno, Cno),   FOREIGN KEY(Sno) REFERENCES S(Sno),
```

FOREIGN KEY(Cno) REFERENCES C(Cno));

注意：常用约束的写法，留意其中关键字的间隔次序，以及标点符号的使用

Foreign key 和 unique 的区别：**Primary key** 在建立的时候会默认地建立此 field 的索引，且此 **primary key** 可以作为作为另外的表的 foreign **key**。

Primary key 一定是 not null，而 **unique** 则没有此限制

3 单表查询：查询所有列，部分列，使用聚合函数，对结果排序，对结果分组

查询信息系（IS）、数学系（MA）和计算机科学系（CS）学生的姓名和性别。

SELECT Sname，Ssex FROM Student

WHERE Sdept IN ('IS'，'MA'，'CS');

关于ＬＩＫＥ语句的使用类似于正则表达式，其中的％表示任意多的字符“１％２”表示以“1 开头 2 结尾的一个字符串”，但是若被匹配的 语句中 有“%”，则进行转义。“_”表示任何一个字符 ，“_”则表示一个下划线 。

逻辑运算中 and 的优先级高于 or

5 个重要的集合函数：ALL 为默认值

计数 COUNT（[DISTINCT|ALL] *）COUNT（[DISTINCT|ALL] <列名>）

计算总和 SUM（[DISTINCT|ALL] <列名>）

计算平均值 AVG（[DISTINCT|ALL] <列名>）

求最大值 MAX（[DISTINCT|ALL] <列名>）

求最小值 MIN（[DISTINCT|ALL] <列名>）

Group by 的使用，作用在查询中间结果

多表查询

连接查询，嵌套查询，集合查询。

嵌套查询，返回数值只有一个时，可以使用逻辑运算符，返回值为多个时使用 any，all,exists ,(not) in

另外 exist 与 in 同义 all 与 not in 同义

Select * from 学生 where 性别='女' and 年龄< all(select 年龄 from from 学生 where 性别='男')

查询结果为年龄小于所有男生的女生

设职工-社团数据库有三个基本表：

职工（职工号，姓名，年龄，性别）；

社会团体（编号，名称，负责人号，活动地点）

参加（职工号，编号，参加日期）

查找没有参加任何团体的职工情况；

Select * from 职工 where not exists(select * from 参加 where 参加.职工号=职工.职工号)；

查找参加了全部社会团体的职工情况；

Select * from 职工 where not exists(select * from 社会团体

where not exists(select * from 参加 where 参加.职工号=职工.职工号 and 参加.编号=社会团体.编号))；

查找参加了职工号为 1001 的职工所参加的全部社会团体的职工的职工号。

Select 职工号 from 职工 where not exists(select * from 参加 1

where 1.职工号='1001'and not exists(select * from 参加 2 where 2.编号=1.编号 and 2.职工号=职工.职工号))；

相关子查询

Select 姓名 from 学生 s，成绩 g，where s.学号=g.学号 and 课程号='C01'

Select 姓名 from 学生 s where exists (select * from 成绩 where 学号=s.学号

And 课程号='C01')

两者效果一样.

集合运算：

Union 并运算

查询所有男学生和男教师的姓名年龄

Select Name,Age from Teacher where Sex=’male’

Union

Select Name,Age from Student where Sex=’male’

Intersect 交运算

Select Name,Age from Teacher

Intersect

Select Name,Age from Student

查询既是学生也是老师的人员的姓名和年龄

Minus 差运算

注意两个语句的前后顺序

查询不是学生的老师。对调次序后则是不是老师的学生

Select Name,Age from Teacher

Minus Select Name,Age from Student

集合运算中 order by 的使用

- ORDER BY 子句只能用于对最终查询结果排序，不能对中间结果排序
- 任何情况下，ORDER BY 子句只能出现在最后
- 对集合操作结果排序时，ORDER BY 子句中用数字指定排序属性

错误写法

```
SELECT *
    FROM Student
    WHERE Sdept= 'CS'
    ORDER BY Sno （不能对中间结果排序）
    UNION
    SELECT *
    FROM Student
    WHERE Sage<=19
    ORDER BY Sno;（不能对中间结果排序）
```

正确写法

```
SELECT *
    FROM Student
    WHERE Sdept= 'CS'
    UNION
    SELECT *
    FROM Student
    WHERE Sage<=19
    ORDER BY 1; //其中的 1 表示按照第一列的属性进行排序，使用数字指示，且本句必须在最后
```

视图中的查询：

查询有 3 门以上课程是 90 分以上的学生的学号及（90 分以上的）课程数

```
SELECT Sno, COUNT(*) FROM SC WHERE Grade>=90 GROUP BY Sno HAVING COUNT(*)>=3;
```

在 S_G 视图中查询平均成绩在 90 分以上的学生学号和平均成绩

SELECT * FROM S_G WHERE Gavg>=90；

S_G 视图定义： CREATE VIEW S_G (Sno，Gavg) AS

SELECT Sno，AVG(Grade) FROM SC GROUP BY Sno；

错误：

SELECT Sno，AVG(Grade)

FROM SC

WHERE AVG(Grade)>=90

GROUP BY Sno；

正确：

SELECT Sno，AVG(Grade)

FROM SC

GROUP BY Sno

HAVING AVG(Grade)>=90；

考点四 数据库安全性

1.**数据库的安全性**是指保护数据库以防止不合法的使用所造成的数据泄露、更改或破坏。

2.计算机安全标准级别划分:A1、B3、B2、B1、C2、C1、D (级别由高到底排列且高级兼容低级)

3. 实现数据库安全性控制的**常用方法**有：用户标识和鉴别、存储控制、视图机制、审计加密等。

4.存取控制：包括两部分定义用户权限和合法权限检测

自主存取控制 DAC 方法：定义各个用户对不同数据对象的存取权限。当用户对数据库访问时首先检查用户的存取权限。防止了不合法用户对数据库的存取。

强制存取控制 MAC 方法：每一个数据对象被（强制地）标以一定的密级，每一个用户也被（强制地）授予某一个级别的许可证。系统规定只有具有某一许可证级别的用户才能存取某一个密级的数据对象。

5.自主存取控制采用 GRANT 语句向用户授予权限，用 REVOKE 语句收回权限的授予。

GRANT 授予的权限有：SELECT ,INSERT,UPDATE,DELETE,ALL PRIVILEGES 等。 用 WITH GRANT OPTION 则表示，获得某种权限的用户还可以把这种权限再授予其他用户。

GRANT <权限> ON <用户> [WITH GRANT OPTION]

REVOKE <权限> ON <对象名> FROM <用户>

收回权限时，若该用户已将权限授予其他用户则必须级联(CASCADE)收回

6. 视图机制：通过视图机制把要保密的数据对无权存取的用户隐藏起来，从而自动地对数据提供一定程度的安全保护。

7. 审计功能把用户对数据库的所有操作自动的记录下来放入审计日志中。

8. 数据加密是防止数据库中数据在存储和传输中失密的有效手段，有替换和置换两种加密方法。

考点五 数据库完整性

1. **数据库的完整性**是指数据的正确性、有效性和相容性。

2. 数据的完整性和安全性是两个不同的概念，但是有一定的联系。

完整性是为了防止数据库中存在不符合语义的数据，防止错误信息的输入和输出，即所谓垃圾进垃圾出所造成的无效操作和错误结果。安全性是保护数据库防止恶意的破坏和非法的存取。即安全性措施的防范对象是非法用户和非法操作，完整性措施防范对象是不合语义的数据。

3. 数据库完整性约束条件包括实体完整性、参照完整性和用户定义完整性

违约处理：

违反实体完整性约束时，拒绝执行操作；

违反参照完整性约束时，可以拒绝执行、级联操作或设置为空；

违反用户定义完整性约束时，可以拒绝执行该操作。

4．触发器是用户定义在关系表上的一类由事件驱动的特殊过程。

考点六　关系数据理论

一、基本概念

函数依赖：设 R (U)是一个关系模式，U 是 R 的属性集合，X 和 Y 是 U 的子集。对于 R (U)的任意一个可能的关系 r，如果 r 中不存在两个元组，它们在 X 上的属性值相同， 而在 Y 上的属性值不同， 则称“X 函数确定 Y”或“Y 函数依赖于 X”，记作 X→Y。

完全函数依赖、部分函数依赖：在 R(U)中，如果 X→Y，并且对于 X 的任何一个真子集 X，都有 X′→Y，则称 Y 对 X 完全函数依赖；若 X→Y，但 Y 不完全函数依赖于 X，则称 Y 对 X 部分函数依赖。

传递依赖：在关系 R (U)中，如果 X→Y（Y⊄X），Y→X，Y→Z，则称 Z 对 X 传递函数依赖。

候选码、主码：设 K 为 R(U，F)中的属性或属性组合，若 K→U 则 K 为 R 的候选码。若候选码多于一个，则选定其中的一个为主码。

外码：关系模式 R 中属性或属性组 X 并非 R 的码，但 X 是另一个关系模式的码，则称 X 是 R 的外部码也称外码。

全码：整个属性组是码，称为全码（ All-key）。

1NF：如果一个关系模式 R 的所有属性都是不可分的基本数据项，则 R∈1NF

2NF：若关系模式 R∈1NF，并且每一个非主属性都完全函数依赖于 R 的关键字，则 R∈2NF。

3NF：关系模式 R<U,F> 中若不存在这样的关键字 X、属性组 Y 及非主属性 Z(Z⊄Y)，使得 X→Y，Y→X，Y→Z 成立，则称 R<U,F>∈3NF。

BCNF：设关系模式 R<U,F>∈1NF，如果对于 R 的每个函数依赖 X→Y，若 Y⊄X，则 X 必含有候选关键字，那么 R∈BCNF。

二、关系规范化

1．规范化：把一个低一级的关系模式分解为高一级关系模式的过程。

2．范式：关系模式满足的确定约束条件，由低到高分为 1NF、2NF、3NF、BCNF、4NF、5NF 等。

3．1NF、2NF、3NF 和 BCNF 的内涵可概括为：

（1）每一个属性都是不可再分解的属性（1NF 的要求）

（2）非主属性完全函数依赖于码（2NF 的要求）

（3）非主属性不传递依赖于任何一个候选码（3NF 的要求）

（4）主属性对不含它的码完全函数依赖（BCNF 的要求）

（5）没有属性完全函数依赖于一组非主属性（BCNF 的要求）

三、数据依赖

数据依赖是通过一个关系中属性间值的相等与否体现出来的数据间的相互关系，是现实世界属性间相互联系的抽象，是数据内在的性质，是语义的体现。现在人们已经提出了许多种类型的数据依赖，其中最重要的是函数依赖和多值依赖。

四、关系模式规范化的基本步骤如图所示。

①对 1NF 关系进行投影，消除原关系中非主属性对码的函数依赖，将 1NF 关系转换为若干个 2NF 关系。

②对 2NF 关系进行投影，消除原关系中非主属性对码的传递函数依赖，从而产生一组 3NF 关系。

③对 3NF 关系进行投影，消除原关系中主属性对码的部分函数依赖和传递函数依赖（也就是说，使决定属性都成为投影的候选码），得到一组 BCNF 关系。

消除决定属性集非码的非平凡函数依赖

1NF

消除非主属性对码的部分函数依赖

2NF

消除非主属性对码的传递函数依赖

3NF

消除主属性对码的部分和传递函数依赖

BCNF

消除非平凡且非函数依赖的多值依赖

4NF

消除不是由候选码所蕴含的连接依赖

5NF

以上三步也可以合并为一步：对原关系进行投影，消除决定属性不是候选码的任何函数依赖。1NF/2NF/3NF 存在的问题：①插入异常 ②删除异常 ③数据冗余度大 ④修改复杂

BCNF 问题:①数据冗余度大 ②增加操作复杂 ③删除操作复杂 ④修改操作复杂

典型例题：

写出 3 个关系模式分别满足：

1） 是 1NF，不是 2NF；

2） 是 2NF，不是 3NF；

3） 是 3NF，也是 BCNF；

各用两句话分别说明你所写的关系模式是前者，不是（或也是）后者。

参考答案 1）学生选课（学号，姓名，课程号，成绩）

属性不可分，是 1NF；存在非主属性对键码的部分依赖（学号，课程号 姓名），不是 2NF。

2）学生（学号，姓名，系别，系主任）

键码为单属性，不存在部分依赖，是 2NF；存在非主属性对键码的传递依赖（学号→姓名，系别；系别 学号；系别→系主任；学号 系主任），不是 3NF。

3） 学生（学号，姓名，年龄）

非主属性（姓名，年龄）对键码不存在部分依赖和传递依赖，是 3NF；

主属性（学号）对键码也不存在部分依赖和传递依赖，是 BCNF。

1)键码:{SNo,CN}和{SNo,TN}

函数依赖:SNo ⟶ SN,SA （BC 范式违例）

TN ⟶ CN （BC 范式违例）

SNo,CN ⟶ TN,G

a) $SNo,CN \xrightarrow{P} SN,SA$

SNo,TN ⟶ G

b) $SNo,TN \xrightarrow{P} CN$

c) $SNo,TN \xrightarrow{P} SN,SA$ （a,b,c 为部分依赖,可不写）

2)STC1(SNo,SN,SA)

STC2(TN,CN)

STC3(SNo,TN,G)

3、涉及到学生、教师和课程的关系模式 STC(SNo，SN，SA，TN，CN，G)，其中 6 个属性分别为学生的学号、姓名、年龄、教师的姓名、课程名以及学生的成绩。假设学生有重名，课程名也可能有重名。又假设每个教师只教一门课，但一门课可有几个教师开设。当某个学生选定某门课后，其上课教师就固定了。1)写出键码和函数依赖； 2)分解关系模式使之属于 BC 范式。

考点七 数据库设计概述

一、数据库的**设计过程：**

（1）需求分析；（2）概念结构设计；（3）逻辑结构设计；（4）数据库物理设计；（5）数据库实施；（6）数据库运行和维护。

二、需求分析

1. 需求分析阶段的设计目标是通过详细调查现实世界要处理的对象（组织、部门、企业等），充分了解原系统（手工系统或计算机系统）工作概况，明确用户的各种需求，然后在此基础上确定新系统的功能。

2. 数据字典是系统中各类数据描述的集合。数据字典的内容包括：

（1）数据项；（2）数据结构；（3）数据流；（4）数据存储；（5）处理过程。

其中数据项是数据的最小组成单位，若干个数据项可以组成一个数据结构。数据字典通过对数据项和数据结构的定义来描述数据流和数据存储的逻辑内容。

三、概念设计结构

1.概念结构是信息世界的结构，即概念模型，

2.重要性：数据库概念设计是整个数据库设计的关键，将在需求分析阶段所得到的应用需求首先抽象为概念结构，以此作为各种数据模型的共同基础，从而能更好地、更准确地用某一种 DBMS 实现这些需求。

3.设计步骤：概念结构的设计方法有多种，其中最经常采用的策略是自底向上方法，该方法的设计步骤通常分为两步：第 1 步是抽象数据并设计局部视图，第 2 步是集成局部视图，得到全局的概念结构

4. 视图集成需两步：合并分 E—R 图，生成初步 E—R 图；修改和重构，消除不必要的冗余，生成基本 E—R 图。

合并 E—R 图时会存在许多不一致的地方，即冲突。**冲突种类**有：属性冲突、命名冲突和结构冲突。处理属性冲突和命名冲突可以通过讨论和协商，而处理结构冲突则需要根据语义对实体联系的类型进行综合调整如求并集。

四、逻辑结构设计

1. 数据库的逻辑结构设计就是把概念结构设计阶段设计好的基本 E-R 图转换为与选用的 DBMS 产品所支持的数据模型相符合的逻辑结构。即 E-R 图转化成关系模型。

2.注意**实体和联系**都得转化为关系模式：

一个实体转化成一个关系模式。实体的属性就是关系的属性，实体的码就是关系的码。

实体间的联系转化成关系模式则有以下几种情况（1）1：1 的联系转化，联系可以转化成一个独立的关系模式，也可以与任何一个实体对应的关系模式合并。如果联系单独转化成一个关系模式，，则与该联系相连的各实体的码以及联系本身的属性均转化成关系的属性，每个实体的码均是关系的候选码。如果联系归并到实体的某一端则需要在该关系模式的属性中加入另一个关系模式的码和联系本身的属性。

1：n 的联系可以转化成独立的关系模式，也可以与 n 端合并。独立转化时，各实体的码以及联系本身的属性都是关系的属性，而关系的码是 n 端实体的码

m：n 的联系只可以转化成一个独立的关系模式，联系不能和任意一端合并。各实体的码以及联系本身的属性都是关系的属性，各实体的码共同组成关系的码

考点八 关系查询处理和查询优化

1. 关系数据库查询优化的总目标是：选择有效的策略，求得给定关系表达式的值。

2. 查询优化的一般准则：（1）选择运算应尽可能先做。 （2）在执行连接前对关系适当地预处理。分为两种：在连接属性上建立索引和对关系排序，然后执行连接。（3）把投影运算和选择运算同时进行。（4）把投影同其前或其后的双目运算结合起来执行。（5）把某些选择同在它前

面要执行的笛卡尔积结合起来成为一个连接运算。（6）找出公共子表达式。

3.优化查询就是选择一个高效执行的查询处理策略。按优化层次可分为代数优化和物理优化。代数优化是指按照一定的规则，改变代数表达式中操作的次序和组合，使查询执行更高效；物理优化则是指存取路径和底层操作算法的选择。

4. 关系代数等价变换: 所谓关系代数表达式的等价是指用相同的关系代替两个表达式中相应的关系所得到的结果是相同的。

5. 优化的一般步骤:（1）把查询转换成某种内部表示，通常用的内部表示是语法树。 （2）把语法树转换成标准(优化)形式。 即利用优化算法，把原始的语法树转换成优化的形式。 （3）选择低层的存取路径。 （4）生成查询计划，选择代价最小的.

6.代数优化查询树：（1）生成原始查询树时，以 SELECT 对应投影操作，FROM 对应笛卡尔乘积，WHERE 对应选择。（2）一般先做选择和投影操作，靠近树的叶子结点。（3）连接时先做小关系连接再做大关系连接。

考点九 数据库恢复技术

1.所谓事务是用户定义的一个数据库操作序列，这些操作要么全做要么全不做，是一个不可分割的工作单位。

2.**事务的四个特性**：原子性、一致性、隔离性和持续性。

3. 数据库系统中可能发生各种各样的故障，大致可分为：**事务内部故障；系统故障；介质故障**。

事务故障、系统故障和介质故障都会影响事务的正常执行；而只有介质故障会破坏数据库数据。

4.数据库恢复的基本技术：数据转储和登录日志文件。

数据转储是数据库恢复中采用的基本技术。所谓转储即 DBA 定期地将数据库复制到磁带或另一个磁盘上保存起来的过程。当数据库遭到破坏后可以将后备副本重新装入，将数据库恢复到转储时的状态。

转储可以分为静态转储(转储期间不允许对数据库进行任何存取、修改活动)和动态转储(转储期间允许对数据进行存取或修改)，也可分为海量转储(每次转储全部数据库)和增量转储(每次只转储上次转储后更新过的数据)。

最好每晚进行一次动态增量转储，每周进行一次动态海量转储，每月进行一次静态海量转储。

5. 不同故障的恢复方法：事务故障和系统故障由系统自动恢复，介质故障需重装数据库重做已完成的事物。

考点十 并发控制

1. 当多个事务并发地存取数据库时就会产生同时读取和 /或修改同一数据的情况。若对并发操作不加控制就可能会存取和存储不正确的数据，破坏数据库的一致性。所以数据库管理系统必须提供并发控制机制。事务是并发控制的基本单位，并发控制技术保证事务的一致性，隔离性。

2.并发操作带来的数据不一致性包括三类：**丢失修改、不可重复读和读“脏”数据。**

3.封锁是实现并发控制的重要技术，封锁类型有**排它锁 X(写锁)和共享锁 S(读锁)**

4.封锁可能引起活锁和死锁等问题。**活锁**是指当若干事务要对同一数据项加锁时，造成一些事务的永远等待，得不到控制权的现象，避免

活锁的方法是采用先来先服务的策略。死锁是指两个以上事务集合中的每个事务都在等待加锁当前已被另一事务加锁的数据项，从而造成相互等待的现象。死锁的诊断方法有:超时法和等待图法。解除死锁的方法是选择一个处理死锁代价最小的事务，将其撤销，并释放此事务持有的所有的锁。

5.运用 X 和 S 锁是常用的封锁协议是三级封锁协议。

	X锁		S锁		一致性保证		
	操作结束释放	事务结束释放	操作结束释放	事务结束释放	不丢失修改	不读"脏"数据	可重复读
1级封锁协议		√			√		
2级封锁协议		√	√		√	√	
3级封锁协议		√		√	√	√	√

6.**可串行化的调度**：如果几个事务并行（交错）执行的结果和按次序串行执行的结果相同，则称该并行执行结果是正确的。这样的调度称为可串行化的调度。

7. 两段锁协议是指所有事务必须分两个阶段对数据库项加锁和解锁：(1)在对任何数据进行读、写操作之前，首先要申请并获得对该数据的封锁；(2)在释放一个封锁之后，事务不再申请和获得任何其他锁。

若遵循两段锁协议则调度正确，而调度正确不一定遵循两段锁协议。

8.封锁对象的大小称为封锁粒度，多粒度封锁需要加意向锁，**意向锁**有意向共享锁 IS、意向排它锁 IX 和共享意向排它锁 SIX。多粒度封锁中对数据树申请加锁是从上往下的，释放锁是从下往上的。