第一节

一、相关概念

1. Data：数据，是数据库中存储的基本对象，是描述事物的符号记录。

2. Database：数据库，是长期储存在计算机内、有组织的、可共享的大量数据的集合。

3. DBMS：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，用于科学地组织、存储和管理数据、高效地获取和维护数据。

4. DBS：数据库系统，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、应用系统、数据库管理员（DBA）构成。

5. 数据模型：是用来抽象、表示和处理现实世界中的数据和信息的工具，是对现实世界的模拟，是数据库系统的核心和基础；其组成元素有数据结构、数据操作和完整性约束。

6. 概念模型：也称信息模型，是按用户的观点来对数据和信息建模，主要用于数据库设计。

7. 逻辑模型：是按计算机系统的观点对数据建模，用于 DBMS 实现。

8. 物理模型：是对数据最底层的抽象，描述数据在系统内部的表示方式和存取方法，在磁盘或磁带上的存储方式和存取方法，是面向计算机系统的。

9. 实体和属性：客观存在并可相互区别的事物称为实体。实体所具有的某一特性称为属性。

10.E-R 图：即实体-关系图，用于描述现实世界的事物及其相互关系，是数据库概念模型设计的主要工具。

11.关系模式：从用户观点看，关系模式是由一组关系组成，每个关系的数据结构是一张规范化的二维表。

12.型/值：型是对某一类数据的结构和属性的说明；值是型的一个具体赋值，是型的实例。

13.数据库模式：是对数据库中全体数据的逻辑结构（数据项的名字、类型、取值范围等）和特征（数据之间的联系以及数据有关的安全性、完整性要求）的描述。

14.数据库的三级系统结构：外模式、模式和内模式。

15.数据库内模式：又称为存储模式，是对数据库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。一个数据库只有一个内模式。

16.数据库外模式：又称为子模式或用户模式，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。通常是模式的子集。一个数据库可有多个外模式。

17.数据库的二级映像：外模式/模式映像、模式/内模式映像。

二、重点知识点

1. 数据库系统由数据库、数据库管理系统、应用系统和数据库管理员构成。

2. 数据模型的组成要素是：数据结构、数据操作、完整性约束条件。

3. 实体型之间的联系分为一对一、一对多和多对多三种类型。

4. 常见的数据模型包括：关系、层次、网状、面向对象、对象关系映射等几种。

5. 关系模型的完整性约束包括：实体完整性、参照完整性和用户定义完整性。

6. 阐述数据库三级模式、二级映象的含义及作用。

数据库三级模式反映的是数据的三个抽象层次：模式是对数据库中全体数据的逻辑结构和特征的描述。内模式又称为存储模式，是对数据库物理结构和存储方式的描述。外模式又称为子模式或用户模式，是对特定数据库用户相关的局部数据的逻辑结构和特征的描述。

数据库三级模式通过二级映象在 DBMS 内部实现这三个抽象层次的联系和转换。外模式面向应用程序， 通过外模式/模式映象与逻辑模式建立联系， 实现数据的逻辑独立性。 模式/内模式映象建立模式与内模式之间的一对一映射， 实现数据的物理独立性。

第二节

一、相关概念

1. 主键： 能够唯一地标识一个元组的属性或属性组称为关系的键或候选键。 若一个关系有多个候选键则可选其一作为主键(Primary key)。

2. 外键：如果一个关系的一个或一组属性引用(参照)了另一个关系的主键，则称这个或这组属性为外码或外键(Foreign key)。

3. 关系数据库： 依照关系模型建立的数据库称为关系数据库。 它是在某个应用领域的所有关系的集合。

4. 关系模式： 简单地说，关系模式就是对关系的型的定义， 包括关系的属性构成、各属性的数据类型、 属性间的依赖、 元组语义及完整性约束等。 关系是关系模式在某一时刻的状态或内容， 关系模型是型， 关系是值， 关系模型是静态的、 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。

5. . 实体完整性：用于标识实体的唯一性。它要求基本关系必须要有一个能够标识元组唯一性的主键，主键不能为空，也不可取重复值。

6. 参照完整性： 用于维护实体之间的引用关系。 它要求一个关系的外键要么为空， 要么取与被参照关系对应的主键值，即外键值必须是主键中已存在的值。

7. 用户定义的完整性：就是针对某一具体应用的数据必须满足的语义约束。包括非空、 唯一和布尔条件约束三种情况。

二、重要知识点

1. 关系数据库语言分为关系代数、关系演算和结构化查询语言三大类。

2. 关系的 5 种基本操作是选择、投影、并、差、笛卡尔积。

3.关系模式是对关系的描述，五元组形式化表示为：R（U，D，DOM，F），其中

R —— 关系名

U —— 组成该关系的属性名集合

D —— 属性组 U 中属性所来自的域

DOM —— 属性向域的映象集合

F —— 属性间的数据依赖关系集合

4.笛卡尔乘积，选择和投影运算如下

设R是包含k1个元组的n目关系，S是包含k2个元组的m目关系，写出R与S的**笛卡尔积**运算公式并说明其含义。

运算公式为：$R \times S = \{ \overset{\frown}{t_r t_s} \mid t_r \in R \wedge t_s \in S \}$

运算结果为一个(n+m)目关系，其每个元组的前n列是关系R的一个元组，后m列是关系S的一个元组。

运算结果包含由k1×k2个元组，由来自R和S的所有元组交叉组合而成

写出关系选择(σ)运算的公式并说明其含义。

运算公式为：$\sigma_F(R) = \{ t \mid t \in R \wedge F(t) = \text{‘真’} \}$

选择运算的含义为从关系R中找出符合条件的所有元组。式中：F为选择条件，t表示元组，它是R元组的子集，并使逻辑表达式F(t)为真。

写出关系投影(π)运算的公式并说明其含义。

运算公式为：$\pi_A(R) = \{ t[A] \mid t \in R \}$

投影运算的含义为从关系R中选择出若干属性列组成新的关系。式中A为R的部分属性组，t表示元组，t[A]表示由属性组A上的分量构成的元组。

第三节

一、相关概念

1. SQL：结构化查询语言的简称， 是关系数据库的标准语言。SQL是一种通用的、 功能极强的关系数据库语言， 是对关系数据存取的标准接口， 也是不同数据库系统之间互操作的基础。集数据查询、数据操作、数据定义、和数据控制功能于一体。

2. 数据定义：数据定义功能包括模式定义、表定义、视图和索引的定义。

3. 嵌套查询：指将一个查询块嵌套在另一个查询块的 WHERE 子句或 HAVING 短语的条件中的查询。

二、重要知识点

1. SQL 数据定义语句的操作对象有：模式、表、视图和索引。
2. SQL 数据定义语句的命令动词是：CREATE、DROP 和 ALTER。
3. RDBMS 中索引一般采用 B+树或 HASH 来实现。
4. 索引可以分为唯一索引、非唯一索引和聚簇索引三种类型。

查询条件	运算符 / 谓词
比　较	=，>，<，>=，<=，!=，<>，!>，!<
确定范围	BETWEEN AND，NOT BETWEEN AND
确定集合	IN，NOT IN
字符匹配	LIKE，NOT LIKE
空　值	IS NULL，IS NOT NULL
多重条件（逻辑运算）	AND，OR，NOT

操作对象	操作方式		
	创建	删除	修改
模式	CREATE SCHEMA	DROP SCHEMA	
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	CREATE INDEX	DROP INDEX	

6.SQL 创建表语句的一般格式为

CREATE TABLE <表名>

(<列名> <数据类型>[<列级完整性约束>]

[，<列名> <数据类型>[<列级完整性约束>]] …

[，<表级完整性约束>]) ；

其中<数据类型>可以是数据库系统支持的各种数据类型，包括长度和精度。

列级完整性约束为针对单个列(本列)的完整性约束，　包括　PRIMARY KEY、　REFERENCES 表名(列名)、UNIQUE、NOT NULL 等。

表级完整性约束可以是基于表中多列的约束，包括　PRIMARY KEY (列名列表) 、FOREIGN KEY REFERENCES 表名(列名) 等。

7. SQL 创建索引语句的一般格式为

CREATE [UNIQUE] [CLUSTER] INDEX <索引名>

ON <表名> (<列名列表>) ；

其中 UNIQUE：表示创建唯一索引，缺省为非唯一索引；

CLUSTER：表示创建聚簇索引，缺省为非聚簇索引；

<列名列表>：一个或逗号分隔的多个列名，每个列名后可跟　ASC 或　DESC，表示升/降序，缺省为升序。多列时则按为多级排序。

8. SQL 查询语句的一般格式为

SELECT [ALL | DISTINCT] <算术表达式列表> FROM <表名或视图名列表>

[WHERE <条件表达式　1>]

[GROUP BY <属性列表　1> [HAVING <条件表达式　2 >]]

[ORDER BY <属性列表　2> [ASC | DESC]] ;

其中

ALL / DISTINCT：　缺省为　ALL，　即列出所有查询结果记录，　包括重复记录。　DISTINCT 则对重复记录只列出一条。

算术表达式列表：一个或多个逗号分隔的算术表达式，表达式由常量(包括数字和字符串)、列名、函数和算术运算符构成。每个表达式后还可跟别名。也可用　*代表查询表中的所有列。

<表名或视图名列表>：　一个或多个逗号分隔的表或视图名。　表或视图名后可跟别名。

条件表达式　1：包含关系或逻辑运算符的表达式，代表查询条件。

条件表达式 2：包含关系或逻辑运算符的表达式，代表分组条件。

<属性列表　1>：一个或逗号分隔的多个列名。

<属性列表　2>：　一个或逗号分隔的多个列名，　每个列名后可跟　ASC 或　DESC，　表示升/降序，缺省为升序。

关于 SQL 语句的知识这里先作如上简略介绍，具体写法下次将专门拿出一篇来叙述。

第四节

一、相关概念和知识

1.触发器是**用户定义在基本表上的一类由事件驱动的特殊过程**。由服务器自动激活，　能执行更为复杂的检查和操作，具有更精细和更强大的数据控制能力。使用 CREATE TRIGGER 命令建立触发器。

2.计算机系统存在**技术安全、管理安全和政策法律**三类安全性问题。

3. TCSEC/TDI 标准由**安全策略、责任、保证和文档**四个方面内容构成。

4. 常用存取控制方法包括**自主存取控制(DAC)和强制存取控制(MAC)**两种。

5. 自主存取控制(DAC)的 SQL 语句包括 GRANT 和 REVOKE 两个。　用户权限由数据对象和操作类型两部分构成。

权限	可否执行的操作			
	CREATE USER	CREATE SCHEMA	CREATE TABLE	CRUD 操作
DBA	可以	可以	可以	可以
RESOURCE	不可以	不可以	可以	可以
CONNECT	不可以	不可以	不可以	可以 但必须被授权

6. 常见 SQL　自主权限控制命令和例子。

1) 把对　Student　和　Course　表的全部权限授予所有用户。

GRANT ALL PRIVILIGES ON TABLE Student，Course TO PUBLIC　;

2) 把对　Student　表的查询权和姓名修改权授予用户　U4。

GRANT SELECT，UPDATE(Sname) ON TABLE Student TO U4　;

3) 把对　SC　表的插入权限授予　U5　用户，并允许他传播该权限。

GRANT INSERT ON TABLE SC TO U5 WITH GRANT OPTION　;

4) 把用户　U5　对　SC　表的　INSERT　权限收回，同时收回被他传播出去的授权。

REVOKE INSERT ON TABLE SC FROM U5 CASCADE　;

5) 创建一个角色 R1，并使其对 Student 表具有数据查询和更新权限。

CREATE ROLE R1；

GRANT SELECT，UPDATE ON TABLE Student TO R1；

6) 对修改 Student 表结构的操作进行审计。

AUDIT ALTER ON Student ；

数据库知识总结(2)范式

一、相关概念和知识点

1.数据依赖：反映一个关系内部属性与属性之间的约束关系，是现实世界属性间相互联系的抽象，属于数据内在的性质和语义的体现。

2. 规范化理论：是用来设计良好的关系模式的基本理论。它通过分解关系模式来消除其中不合适的数据依赖，以解决插入异常、删除异常、更新异常和数据冗余问题。

3. 函数依赖：简单地说，对于关系模式的两个属性子集 X 和 Y，若 X 的任一取值能唯一确定 Y 的值，则称 Y 函数依赖于 X，记作 X→Y。

4. 非平凡函数依赖：对于关系模式的两个属性子集 X 和 Y，如果 X→Y，但 Y!⊆X，则称 X→Y 为非平凡函数依赖；如果 X→Y，但 Y⊆X，则称 X→Y 为非平凡函数依赖。

5. 完全函数依赖：对于关系模式的两个属性子集 X 和 Y，如果 X→Y，并且对于 X 的任何一个真子集 X'，都没有 X'→Y，则称 Y 对 X 完全函数依赖。

6. 范式：指符合某一种级别的关系模式的集合。在设计关系数据库时，根据满足依赖关系要求的不同定义为不同的范式。

7. 规范化：指将一个低一级范式的关系模式，通过模式分解转换为若干个高一级范式的关系模式的集合的过程。

8. 1NF：若关系模式的所有属性都是不可分的基本数据项，则该关系模式属于 1NF。

9. 2NF：1NF 关系模式如果同时满足每一个非主属性完全函数依赖于码，则该关系模式属于 2NF。

10. 3NF：若关系模式的每一个非主属性既不部分依赖于码也不传递依赖于码，则该关系模式属于 3NF。

11. BCNF：若一个关系模式的每一个决定因素都包含码，则该关系模式属于 BCNF。

12. 数据库设计：是指对于一个给定的应用环境，构造优化的数据库逻辑模式和物理结构，并据此建立数据库及其应用系统，使之能够有效地存储和管理数据，满足各种用户的应用需求，包括信息管理要求和数据操作要求。

13. 数据库设计的 6 个基本步骤：需求分析，概念结构设计，逻辑结构设计，物理结构设计，数据库实施，数据库运行和维护。

14. 概念结构设计：指将需求分析得到的用户需求抽象为信息结构即概念模型的过程。也就是通过对用户需求进行综合、归纳与抽象，形成一个独立于具体 DBMS 的概念模型。

15. 逻辑结构设计：将概念结构模型（基本 E-R 图）转换为某个 DBMS 产品所支持的数据模型相符合的逻辑结构，并对其进行优化。

16. 物理结构设计：指为一个给定的逻辑数据模型选取一个最适合应用环境的物理结构的过程。包括设计数据库的存储结构与存取方法。

17. 抽象：指对实际的人、物、事和概念进行人为处理，抽取所关心的共同特性，忽略非本质的细节，并把这些特性用各种概念精确地加以描述，这些概念组成了某种模型。

18. 数据库设计必须遵循结构设计和行为设计相结合的原则。

19. 数据字典主要包括数据项、数据结构、数据流、数据存储和处理过程五个部分。

20. 三种常用抽象方法是分类、聚集和概括。

21. 局部 E-R 图之间的冲突主要表现在属性冲突、命名冲突和结构冲突三个方面。

22. 数据库常用的存取方法包括索引方法、聚簇方法和 HASH 方法三种。

23. 确定数据存放位置和存储结构需要考虑的因素主要有： 存取时间、 存储空间利用率和维护代价等。

二、细说数据库三范式

2.1 第一范式（1NF）无重复的列

第一范式（1NF）中数据库表的每一列都是不可分割的基本数据项

同一列中不能有多个值

即实体中的某个属性不能有多个值或者不能有重复的属性。

简而言之，第一范式就是无重复的列。

在任何一个关系数据库中，第一范式（1NF）是对关系模式的基本要求，不满足第一范式（1NF）的数据库就不是关系数据库。

2.2 第二范式（2NF）属性完全依赖于主键[消除部分子函数依赖]

满足第二范式（2NF）必须先满足第一范式（1NF）。

第二范式（2NF）要求数据库表中的每个实例或行必须可以被惟一地区分。

为实现区分通常需要为表加上一个列，以存储各个实例的惟一标识。

第二范式（2NF）要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的惟一标识。简而言之，第二范式就是属性完全依赖于主键。

2.3 第三范式（3NF）属性不依赖于其它非主属性[消除传递依赖]

满足第三范式（3NF）必须先满足第二范式（2NF）。

简而言之，第三范式（3NF）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。

例如，存在一个部门信息表，其中每个部门有部门编号（dept_id）、部门名称、部门简介等信息。那么在的员工信息表中列出部门编号后就不能再将部门名称、部门简介等与部门有关的信息再加入员工信息表中。如果不存在部门信息表，则根据第三范式（3NF）也应该构建它，否则就会有大量的数据冗余。简而言之，第三范式就是属性不依赖于其它非主属性。

2.4 具体实例剖析

下面列举一个学校的学生系统的实例，以示几个范式的应用。

在设计数据库表结构之前，我们先确定一下要设计的内容包括那些。学号、学生姓名、年龄、性别、课程、课程学分、系别、学科成绩，系办地址、系办电话等信息。为了简单我们暂时只考虑这些字段信息。我们对于这些信息，说关心的问题有如下几个方面。

1）学生有那些基本信息

2）学生选了那些课，成绩是什么

3）每个课的学分是多少

4）学生属于那个系，系的基本信息是什么。

首先第一范式（1NF）：数据库表中的字段都是单一属性的，不可再分。这个单一属性由基本类型构成，包括整型、实数、字符型、逻辑型、日期型等。在当前的任何关系数据库管理系统（DBMS）中，不允许你把数据库表的一列再分成二列或多列，因此做出的都是符合第一范式的数据库。

我们再考虑第二范式，把所有这些信息放到一个表中(学号，学生姓名、年龄、性别、课程、课程学分、系别、学科成绩，系办地址、系办电话)下面存在如下的依赖关系。

1）（学号）→ (姓名, 年龄，性别，系别，系办地址、系办电话)

2）(课程名称) → (学分)

3）（学号，课程）→ (学科成绩)

根据依赖关系我们可以把选课关系表 SelectCourse 改为如下三个表：

学生：Student(学号，姓名, 年龄，性别，系别，系办地址、系办电话)；

课程：Course(课程名称, 学分)；

选课关系：SelectCourse(学号, 课程名称, 成绩)。

事实上，对照第二范式的要求，这就是满足第二范式的数据库表，若不满足第二范式，会产生如下问题

数据冗余： 同一门课程由 n 个学生选修，"学分"就重复 n-1 次；同一个学生选修了 m 门课程，姓名和年龄就重复了 m-1 次。

更新异常： 1)若调整了某门课程的学分，数据表中所有行的"学分"值都要更新，否则会出现同一门课程学分不同的情况。

2)假设要开设一门新的课程，暂时还没有人选修。这样，由于还没有"学号"关键字，课程名称和学分也无法记录入数据库。

删除异常： 假设一批学生已经完成课程的选修，这些选修记录就应该从数据库表中删除。但是，与此同时，课程名称和学分信息也被删除了。很显然，这也会导致插入异常。

我们再考虑如何将其改成满足第三范式的数据库表，接着看上面的学生表 Student(学号，姓名, 年龄，性别，系别，系办地址、系办电话)，关键字为单一关键字"学号"，因为存在如下决定关系：

（学号）→ (姓名, 年龄，性别，系别，系办地址、系办电话)

但是还存在下面的决定关系

(学号) → (所在学院)→(学院地点, 学院电话)

即存在非关键字段"学院地点"、"学院电话"对关键字段"学号"的传递函数依赖。

它也会存在数据冗余、更新异常、插入异常和删除异常的情况（这里就不具体分析了，参照第二范式中的分析）。根据第三范式把学生关系表分为如下两个表就可以满足第三范式了：

学生：(学号, 姓名, 年龄, 性别，系别)；

系别：(系别, 系办地址、系办电话)。

SQL 语句总结

SQL 语句中常用关键词及其解释如下：

1）SELECT

将资料从数据库中的表格内选出，两个关键字：从（FROM）数据库中的表格内选出（SELECT)。语法为

SELECT "栏位名" FROM "表格名"。

2）DISTINCT

在上述 SELECT 关键词后加上一个 DISTINCT 就可以去除选择出来的栏位中的重复，从而完成求得这个表格/栏位内有哪些不同的值的功能。语法为

SELECT DISTINCT "栏位名" FROM "表格名"。

3）WHERE

这个关键词可以帮助我们选择性地抓资料，而不是全取出来。语法为

SELECT "栏位名" FROM "表格名" WHERE "条件"

4）AND OR

上例中的 WHERE 指令可以被用来由表格中有条件地选取资料。这个条件可能是简单的（像上一页的例子），也可能是复杂的。复杂条件是由二或多个简单条件透过 AND 或是 OR 的连接而成。语法为：

SELECT "栏位名" FROM "表格名" WHERE "简单条件" {[AND|OR] "简单条件"}+

5）IN

在 SQL 中，在两个情况下会用到 IN 这个指令；这一页将介绍其中之一：与 WHERE 有关的那一个情况。在这个用法下，我们事先已知道至少一个我们需要的值，而我们将这些知道的值都放入 IN 这个子句。语法为：

SELECT "栏位名" FROM "表格名" WHERE "栏位名" IN ('值一', '值二', ...)

6）BETWEEN

IN 这个指令可以让我们依照一或数个不连续 (discrete)的值的限制之内抓出资料库中的值，而 BETWEEN 则是让我们可以运用一个范围 (range) 内抓出资料库中的值，语法为：

SELECT "栏位名" FROM "表格名" WHERE "栏位名" BETWEEN '值一' AND '值二'

7）LIKE

LIKE 是另一个在 WHERE 子句中会用到的指令。基本上， LIKE 能让我们依据一个模式(pattern) 来找出我们要的资料。语法为：

SELECT "栏位名" FROM "表格名" WHERE "栏位名" LIKE {模式}

8）ORDER BY

我们经常需要能够将抓出的资料做一个有系统的显示。这可能是由小往大 (ascending) 或是由大往小(descending)。在这种情况下，我们就可以运用 ORDER BY 这个指令来达到我们的目的。语法为：

SELECT "栏位名" FROM "表格名 [WHERE "条件"] ORDER BY "栏位名" [ASC, DESC]

9）函数

函数允许我们能够对这些数字的型态存在的行或者列做运算，包括 AVG (平均)、COUNT (计数)、MAX (最大值)、MIN (最小值)、SUM (总合)。语法为：

SELECT "函数名"("栏位名") FROM "表格名"

10）COUNT

这个关键词能够帮我我们统计有多少笔资料被选出来，语法为：

SELECT COUNT("栏位名") FROM "表格名"

11）GROUP BY

GROUP BY 语句用于结合合计函数，根据一个或多个列对结果集进行分组。语法为：

SELECT "栏位 1", SUM("栏位 2") FROM "表格名" GROUP BY "栏位 1"

12）HAVING

该关键词可以帮助我们对函数产生的值来设定条件。语法为：

SELECT "栏位 1", SUM("栏位 2") FROM "表格名" GROUP BY "栏位 1" HAVING (函数条件)

13）ALIAS

我们可以通过 ALIAS 为列名称和表名称指定别名，语法为：

SELECT "表格别名"."栏位 1" "栏位别名" FROM "表格名" "表格别名"

下面为一个例子，通过它我们应该能很好地掌握以上关键词的使用方法。

Student(S#,Sname,Sage,Ssex) 学生表

Course(C#,Cname,T#) 课程表

SC(S#,C#,score) 成绩表

Teacher(T#,Tname) 教师表

问题：

1、查询“001”课程比“002”课程成绩高的所有学生的学号；

```
select a.S#
from (select s#,score from SC where C#=’001’) a,
(select s#,score from SC where C#=’002’) b
where a.score>b.score and a.s#=b.s#;
```

2、查询平均成绩大于 60 分的同学的学号和平均成绩；

```
select S#,avg(score)
from sc
group by S# having avg(score) >60;
```

3、查询所有同学的学号、姓名、选课数、总成绩；

```
select Student.S#,Student.Sname,count(SC.C#),sum(score)
from Student left Outer join SC on Student.S#=SC.S#
group by Student.S#,Sname
```

4、查询姓“李”的老师的个数；

```
select count(distinct(Tname))
from Teacher
where Tname like ‘李%’;
```

5、查询没学过“叶平”老师课的同学的学号、姓名；

```
select Student.S#,Student.Sname
from Student
where S# not in (select distinct( SC.S#) from SC,Course,Teacher where SC.C#=Course.C# and Teacher.T#=Course.T#
and Teacher.Tname=’叶平’);
```

6、查询学过“001”并且也学过编号“002”课程的同学的学号、姓名；

```
select Student.S#,Student.Sname
from Student,SC

where Student.S#=SC.S# and SC.C#=’001’and exists( Select * from SC as SC_2 where SC_2.S#=SC.S# and
SC_2.C#=’002’);
```

7、查询学过“叶平”老师所教的所有课的同学的学号、姓名；

```
select S#,Sname
from Student
where S# in
(select S#
from SC ,Course ,Teacher
where SC.C#=Course.C# and Teacher.T#=Course.T# and Teacher.Tname=’叶平’ group by S# having
count(SC.C#)=(select count(C#) from Course,Teacher where Teacher.T#=Course.T# and Tname=’叶平’));
```

8、查询所有课程成绩小于 60 分的同学的学号、姓名；

```
select S#,Sname
from Student
where S# not in (select Student.S# from Student,SC where S.S#=SC.S# and score>60);
```

9、查询没有学全所有课的同学的学号、姓名；

```
select Student.S#,Student.Sname
from Student,SC
where Student.S#=SC.S#
group by Student.S#,Student.Sname having count(C#) <(select count(C#) from Course);
```

10、查询至少有一门课与学号为“1001”的同学所学相同的同学的学号和姓名；

```
select S#,Sname
from Student,SC
where Student.S#=SC.S# and C# in （select C# from SC where S#='1001'）；
```

11、删除学习“叶平”老师课的 SC 表记录；

```
Delect SC
from course ,Teacher
where Course.C#=SC.C# and Course.T#= Teacher.T# and Tname='叶平';
```

12、查询各科成绩最高和最低的分：以如下形式显示：课程 ID，最高分，最低分

```
SELECT L.C# 课程 ID,L.score 最高分,R.score 最低分
FROM SC L ,SC R
WHERE L.C# = R.C#
and
L.score = (SELECT MAX(IL.score)
FROM SC IL,Student IM
WHERE IL.C# = L.C# and IM.S#=IL.S#
GROUP BY IL.C#)
and
R.Score = (SELECT MIN(IR.score)
FROM SC IR
WHERE IR.C# = R.C#
GROUP BY IR.C# );
```

13、查询学生平均成绩及其名次

```
SELECT 1+(SELECT COUNT( distinct 平均成绩)
FROM (SELECT S#,AVG(score) 平均成绩
FROM SC
GROUP BY S# ) T1
WHERE 平均成绩 > T2.平均成绩) 名次, S# 学生学号,平均成绩
FROM (SELECT S#,AVG(score) 平均成绩 FROM SC GROUP BY S# ) T2
ORDER BY 平均成绩 desc;
```

14、查询各科成绩前三名的记录:(不考虑成绩并列情况)

```
SELECT t1.S# as 学生 ID,t1.C# as 课程 ID,Score as 分数
FROM SC t1
WHERE score IN (SELECT TOP 3 score
FROM SC
WHERE t1.C#= C#
ORDER BY score DESC)
ORDER BY t1.C#;
```

15、查询每门功成绩最好的前两名

```
SELECT t1.S# as 学生 ID,t1.C# as 课程 ID,Score as 分数
FROM SC t1
WHERE score IN (SELECT TOP 2 score
FROM SC
WHERE t1.C#= C#
ORDER BY score DESC )

ORDER BY t1.C#;
```

1.事务四大特性

原子性，要么执行，要么不执行

隔离性，所有操作全部执行完以前其它会话不能看到过程

一致性，事务前后，数据总额一致

持久性，一旦事务提交，对数据的改变就是永久的

2.数据库隔离级别

脏读：事务 B 读取事务 A 还没有提交的数据

不可重复读：两次事务读的数据不一致

幻读:事务 A 修改了数据，事务 B 也修改了数据，这时在事务 A 看来，明明修改了数据，咋不一样

3.MYSQL 的两种存储引擎区别（事务、锁级别等等），各自的适用场景

引擎	特性
MYISAM	不支持外键，表锁，插入数据时，锁定整个表，查表总行数时，不需要全表扫描
INNODB	支持外键，行锁，查表总行数时，全表扫描

4.索引有 B+索引和 hash 索引

索引	区别
Hash	hash 索引，等值查询效率高，不能排序,不能进行范围查询
B+	数据有序,范围查询

5.聚集索引和非聚集索引

索引	区别
聚集索引	数据按索引顺序存储，中子结点存储真实的物理数据
非聚集索引	存储指向真正数据行的指针

6.索引的优缺点，什么时候使用索引，什么时候不能使用索引

索引最大的好处是提高查询速度，

缺点是更新数据时效率低，因为要同时更新索引

对数据进行频繁查询进建立索引，如果要频繁更改数据不建议使用索引。

7.InnoDB 索引和 MyISAM 索引的区别

一是主索引的区别，InnoDB 的数据文件本身就是索引文件。而 MyISAM 的索引和数据是分开的。

二是辅助索引的区别：InnoDB 的辅助索引 data 域存储相应记录主

8.索引的底层实现（B+树，为何不采用红黑树，B 树）重点

树　　区别

红黑树　　增加，删除，红黑树会进行频繁的调整，来保证红黑树的性质，浪费时间

B 树也就是 B-树　B 树，查询性能不稳定，查询结果高度不致，每个结点保存指向真实数据的指针，相比 B+树每一层每屋存储的元素更多，显得更高一点。

B+树 B+树相比较于另外两种树,显得更矮更宽，查询层次更浅

9． B+树的实现

一个 m 阶的 B+树具有如下几个特征：

1.有 k 个子树的中间节点包含有 k 个元素（B 树中是 k-1 个元素），每个元素不保存数据，只用来索引，所有数据都保存在叶子节点。

2.所有的叶子结点中包含了全部元素的信息，及指向含这些元素记录的指针，且叶子结点本身依关键字的大小自小而大顺序链接。

3.所有的中间节点元素都同时存在于子节点，在子节点元素中是最大（或最小）元素

10.为什么使用 B+Tree

索引查找过程中就要产生磁盘 I/O 消耗,主要看 IO 次数，和磁盘存取原理有关。

根据 B-Tree 的定义，可知检索一次最多需要访问 h 个节点。数据库系统的设计者巧妙利用了磁盘预读原理，

将一个节点的大小设为等于一个页，这样每个节点只需要一次 I/O 就可以完全载入

局部性原理与磁盘预读

11.Sql 的优化

1.sql 尽量使用索引,而且查询要走索引

2.对 sql 语句优化

子查询变成 left join

limit 分布优化，先利用 ID 定位，再分页

or 条件优化，多个 or 条件可以用 union all 对结果进行合并（union all 结果可能重复）

不必要的排序

where 代替 having,having 检索完所有记录，才进行过滤

避免嵌套查询

对多个字段进行等值查询时，联合索引

12.索引最左前缀问题

如果对三个字段建立联合索引，如果第二个字段没有使用索引，第三个字段也使用不到索引了

13.索引分类，索引失效条件

索引类型	概念
普通索引	最基本的索引，没有任何限制
唯一索引	与"普通索引"类似，不同的就是：索引列的值必须唯一，但允许有空值。
主键索引	它是一种特殊的唯一索引，不允许有空值。
全文索引	针对较大的数据，生成全文索引很耗时好空间。
组合索引	为了更多的提高 mysql 效率可建立组合索引，遵循” 最左前缀 “原则

失效条件

条件是 or,如果还想让 or 条件生效，给 or 每个字段加个索引

like 查询，以%开发

内部函数

对索引列进行计算

is null 不会用，is not null 会用

14.数据库的主从复制

复制方式　操作

异步复制　默认异步复制，容易造成主库数据和从库不一致,一个数据库为 Master,一个数据库为 slave,通过 Binlog 日志,slave 两个线程，一个线程去读 master binlog 日志，写到自己的中继日志一个线程解析日志，执行 sql,master 启动一个线程,给 slave 传递 binlog 日志

半同步复制 只有把 master 发送的 binlog 日志写到 slave 的中继日志，这时主库,才返回操作完成的反馈，性能有一定降低

并行操作　slave 多个线程去请求 binlog 日志

15.long_query 怎么解决

设置参数，开启慢日志功能，得到耗时超过一定时间的 sql

16.varchar 和 char 的使用场景

类型	使用场景
varchar	字符长度经常变的
char	用字符长度固定的

17.数据库连接池的作用

维护一定数量的连接，减少创建连接的时间

更快的响应时间

统一的管理

19.分库分表，主从复制，读写分离

读写分离，读从库，写主库

spring 配置两个数据库，通过 AOP（面向切面编程），在写或读方法前面进行判断得到动态切换数据源。

20.数据库三范式

级别	概念
1NF	属性不可分
2NF	非主键属性，完全依赖于主键属性
3NF	非主键属性无传递依赖

21.关系型数据库和非关系型数据库区别

关系型数据库

优点

1、容易理解：二维表结构是非常贴近逻辑世界一个概念，关系模型相对网状、层次等其他模型来说更容易理解；

2、使用方便：通用的 SQL 语言使得操作关系型数据库非常方便；

3、易于维护：丰富的完整性(实体完整性、参照完整性和用户定义的完整性)大大减低了数据冗余和数据不一致的概率；

4、支持 SQL，可用于复杂的查询。

5.支持事务

缺点

1、为了维护一致性所付出的巨大代价就是其读写性能比较差；

2、固定的表结构；

3、不支持高并发读写需求；

4、不支持海量数据的高效率读写

非关系型数据库

1、使用键值对存储数据；

2、分布式；

优点

无需经过 sql 层的解析，读写性能很高

基于键值对，数据没有耦合性，容易扩展

存储数据的格式：nosql 的存储格式是 key,value 形式

缺点

不提供 sql 支持

22.数据库中 join 的 inner join, outer join, cross join

1.以 A，B 两张表为例

A left join B

选出 A 的所有记录，B 表中没有的以 null 代替

right join 同理

2.inner join

A,B 的所有记录都选出，没有的记录以 null 代替

3.cross join (笛卡尔积)

A 中的每一条记录和 B 中的每一条记录生成一条记录

例如 A 中有 4 条，B 中有 4 条，cross join 就有 16 条记录

23.有哪些锁,select 时怎么加排它锁

锁	概念
乐观锁	自己实现，通过版本号
悲观锁	共享锁，多个事务，只能读不能写，加 lock in share mode
排它锁	一个事务，只能写，for update
行锁	作用于数据行

锁	概念
表锁	作于用表

24.死锁怎么解决

找到进程号，kill 进程

25.最左匹配原则

最左匹配原则是针对索引的

举例来说：两个字段（name,age）建立联合索引，如果 where age=12 这样的话，是没有利用到索引的，

这里我们可以简单的理解为先是对 name 字段的值排序，然后对 age 的数据排序，如果直接查 age 的话，这时就没有利用到索引了，

查询条件 where name= ‘xxx’ and age=xx 这时的话，就利用到索引了，再来思考下 where age=xx and name=’ xxx ‘ 这个 sql 会利用索引吗，

按照正常的原则来讲是不会利用到的，但是优化器会进行优化，把位置交换下。这个 sql 也能利用到索引了