

复习计划Day1

2020年7月27日 23:08

《数据库系统概论》

基础概念

一、绪论

1. **四个基本概念**：数据，数据库，数据库管理系统，数据库系统

2. **数据库管理系统**，位于用户与操作系统之间。

主要功能：数据定义；数据组织、储存和管理；数据操纵功能；数据库的事务管理和运行管理；数据库的建立和维护；其他功能（与网络中其他软件系统通信）

3. **数据库系统**：数据库+数据库管理系统+应用程序+数据库管理员

特点：数据结构化；共享性高、冗余低且易扩充；数据独立性高（物理独立性+逻辑独立性）；由数据库管理系统统一管理控制（安全性保护+完整性保护+并发控制+数据库恢复）

4. **数据模型**：对现实世界数据特征的抽象。

概念模型/信息模型；逻辑模型和物理模型

5. **概念模型**：

实体 (entity)

属性 (attribute)

码 (key)

实体型 (entity type)

实体集 (entity set)

联系 (relationship) 一对一 一对多 多对多

实体-联系方法 (Entity-Relationship approach) : E-R模型

6. **数据模型**：数据结构+动态特征+完整性约束条件

1) 层次模型：根节点+只有一个双亲结点 查询效率高

2) 网状模型：无双亲+可以有多个双亲 存取效率高

3) 关系模型：关系必须规范化 关系（表）元组（行）属性（列）码 域 分量（元组的一个属性值，唯一）

7. **数据库系统结构**：外模式、模式和内模式

模式（逻辑模式）全体数据的逻辑结构和特征，公共数据视图

外模式（子模式）用户模式

内模式（存储模式）唯一，物理结构和储存方式的描述

二级映像：外模式/模式，模式/内模式

二、关系数据库

关键词：域，笛卡尔积

1. **关系**： $D_1 \times D_2 \times \dots \times D_n$, $R(D_1, D_2, \dots, D_n)$, R 为关系的名字, n 为关系的目或者度

单元关系，二元关系

2. **候选码**：某组属性唯一标识

多个候选码中选定主码

主属性：候选码的各个属性 非主属性/非码属性

全码：所有属性

3. **基本关系/基本表/基表**：实际存在的表

查询表：查询结果

视图表：虚表不对应实际数据

4. **关系模式** $R(U,D,DOM,F)$

R 关系名

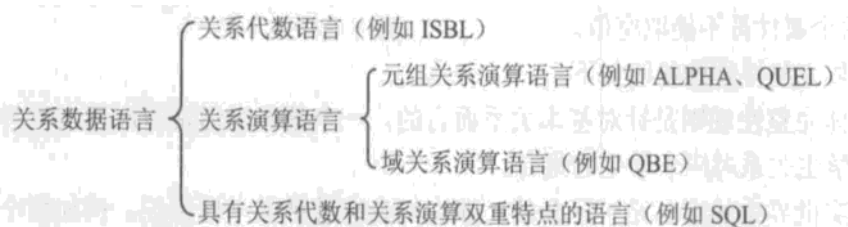
U 属性名集合

D U中属性的域

DOM 属性向域的映像集合

F 属性间数据的依赖关系集合

5. **基本关系操作**：选择，投影，并，差，笛卡尔积



6. **关系的完整性**：实体完整性，参照完整性，用户定义完整性

实体完整性：主属性不为空

参照完整性：外键，要么为空，要么是参照属性的一个值

7. **关系代数**

并，差，交，笛卡尔积

查询，投影

连接：非等值连接，等值连接，自然连接（舍弃 悬浮数组）

外连接：左外连接，右外连接

除

关系演算：元组关系演算语言（ALPHA）元组关系演算（QBE）

三、关系数据库标准语言SQL

1. **模式的定义与删除**

定义模式

Create Schema<模式名> Authorization<用户名>

删除模式

Drop Schema<模式名> <Cascde | Restrict>

级联删除和限制删除

2. **基本表定义，删除与修改**

定义基本表

Create Table<表名>([<列名> <数据类型> [列级完整性约束条件]], [<列名> <数据类

型>[列级完整性约束条件]],``,[<表级完整性约束条件>])

数据类型

数据类型	含义
CHAR(<i>n</i>), CHARACTER(<i>n</i>)	长度为 <i>n</i> 的定长字符串
VARCHAR(<i>n</i>), CHARACTERVARYING(<i>n</i>)	最大长度为 <i>n</i> 的变长字符串
CLOB	字符串大对象
BLOB	二进制大对象
INT, INTEGER	长整数 (4 字节)
SMALLINT	短整数 (2 字节)
BIGINT	大整数 (8 字节)

数据类型	含义
NUMERIC(<i>p</i> , <i>d</i>)	定点数, 由 <i>p</i> 位数字 (不包括符号、小数点) 组成, 小数点后面有 <i>d</i> 位数字
DECIMAL(<i>p</i> , <i>d</i>), DEC(<i>p</i> , <i>d</i>)	同 NUMERIC
REAL	取决于机器精度的单精度浮点数
DOUBLE PRECISION	取决于机器精度的双精度浮点数
FLOAT(<i>n</i>)	可选精度的浮点数, 精度至少为 <i>n</i> 位数字
BOOLEAN	逻辑布尔量
DATE	日期, 包含年、月、日, 格式为 YYYY-MM-DD
TIME	时间, 包含一日的时、分、秒, 格式为 HH:MM:SS
TIMESTAMP	时间戳类型
INTERVAL	时间间隔类型

一个模式内含有多个基本表

修改基本表

```
ALTER TABLE <表名>
[ADD [COLUMN] <新列名><数据类型> [完整性约束]]
[ADD <表级完整性约束>]
[DROP [COLUMN] <列名> [CASCADE|RESTRICT]]
[DROP CONSTRAINT<完整性约束名> [RESTRICT|CASCADE ]]
[ALTER COLUMN <列名><数据类型>];
```

删除基本表

Drop Table<模式名><Restrict | Cascde >

3. 索引的建立与删除

建立索引 Creat Index

```
CREATE [UNIQUE] [CLUSTER] INDEX <索引名>
ON <表名>(<列名> [<次序>] [, <列名> [<次序>]] ...);
```

修改

Drop Index<索引名>

4. 数据查询 Select

```
SELECT [ALL|DISTINCT] <目标列表表达式> [,<目标列表表达式>] ...
FROM <表名或视图名> [,<表名或视图名>...] | (<SELECT 语句>) [AS] <别名>
[WHERE <条件表达式>]
[GROUP BY <列名 1> [HAVING <条件表达式>]]
[ORDER BY <列名 2> [ASC|DESC]] ;
```

Select Distinct 消除重复行

where 查询范围 (Not) Between `` And `` 限定范围; (Not) In 确定集合; (Not) Like 字符匹配 %任意长度 _单个字符; Escape 转义字符; Is Null 空值查询; 多重条件 And Or

Order by 排序 ASC 升序 DASC降序

聚集函数

COUNT(*)	统计元组个数
COUNT([DISTINCT ALL] <列名>)	统计一列中值的个数
SUM([DISTINCT ALL] <列名>)	计算一列值的总和 (此列必须是数值型)
AVG([DISTINCT ALL] <列名>)	计算一列值的平均值 (此列必须是数值型)
MAX([DISTINCT ALL] <列名>)	求一列值中的最大值
MIN([DISTINCT ALL] <列名>)	求一列值中的最小值

WHERE 子句中不能使用聚集函数作为条件表达式, 聚集函数只能用于SELECT子句和 GROUP BY中的HAVING子句

GROUP BY

WHERE 子句与 HAVING 短语的区别在于作用对象不同。WHERE 子句作用于基本表或视图, 从中选择满足条件的元组。HAVING 短语作用于组, 从中选择满足条件的组。

连接查询 多表查询

等值连接 自身连接 外连接 多表连接

嵌套查询

嵌套在WHERE子句或HAVING子句的条件中的查询

ANY 和 ALL 区别

EXISTS 存在两次, 返回true or false

集合查询

交INTERSECT 并UNION 差EXCEPT

派生表查询 即生成的临时表进行查询, 可以改名 As

5. 数据更新 Insert Update Delete

插入元组

```
INSERT INTO <表名> [(<属性列 1> [,<属性列 2>] ...)]
VALUES (<常量 1> [,<常量 2>] ...);
```

插入子查询结果

```
INSERT INTO <表名> [(<属性列 1> [,<属性列 2>] ...)]
子查询;
```

修改数据

```
UPDATE <表名>  
SET <列名>=<表达式> [,<列名>=<表达式>] ...  
[WHERE <条件>];
```

删除数据

```
DELETE  
FROM <表名>  
[WHERE <条件>];
```

6. 空值的处理

属性应有值，但是目前不知道

属性不应有值

不便于填写

IS (NOT) NULL

空值与另一个值的算术运算结果为空值，空值与另一个值的比较运算结果是

UNKNOWN

7. 视图

视图是从一个或几个基本表（或视图）导出的表，虚表。

定义视图

```
CREATE VIEW <视图名> [(<列名> [,<列名>] ...)]  
AS <子查询>  
[WITH CHECK OPTION];
```

删除视图

Drop View <索引名>[CASCADE]

查询视图 同上

关系数据库管理系统执行对视图的查询时，首先进行有效性检查，检查查询中涉及的表、视图等是否存在。如果存在，则从数据字典中取出视图的定义，把定义中的子查询和用户的查询结合起来，转换成等价的对基本表的查询，然后再执行修正了的查询。这一转换过程称为视图消解（view resolution）。

更新视图 最后要转换为对基本表的操作

视图作用：简化操作 多角度看数据 逻辑独立性 安全保护 清晰的表达查询

四、数据库安全性

1. 数据库安全性概述

TCSEC/TDI系统

表 4.1 TCSEC/TDI 安全级别划分

安全级别	定义
A1	验证设计 (verified design)
B3	安全域 (security domains)
B2	结构化保护 (structural protection)
B1	标记安全保护 (labeled security protection)
C2	受控的存取保护 (controlled access protection)
C1	自主安全保护 (discretionary security protection)
D	最小保护 (minimal protection)

CC安全性说明

评估保证级	定义	TCSEC 安全级别 (近似相当)
EAL1	功能测试 (functionally tested)	
EAL2	结构测试 (structurally tested)	C1
EAL3	系统地测试和检查 (methodically tested and checked)	C2
EAL4	系统地设计、测试和复查 (methodically designed, tested and reviewed)	B1
EAL5	半形式化设计和测试 (semiformally designed and tested)	B2
EAL6	半形式化验证的设计和测试 (semiformally verified design and tested)	B3
EAL7	形式化验证的设计和测试 (formally verified design and tested)	A1

2. 数据安全性控制

用户身份鉴别

静态口令鉴别, 动态口令鉴别, 生物特征鉴别, 智能卡鉴别

存取控制 定义用户权限 合法权限检查

自主存取控制DAC 强制存取控制MAC

3. 自助存取控制

授权语句

GRANT 语句的一般格式为

```
GRANT <权限> [, <权限>] ...
ON <对象类型> <对象名> [, <对象类型> <对象名>] ...
TO <用户> [, <用户>] ...
[WITH GRANT OPTION];
```

收回权限

授予用户的权限可以由数据库管理员或其他授权者用 REVOKE 语句收回，REVOKE 语句的一般格式为

```
REVOKE <权限> [,<权限>]...  
ON <对象类型> <对象名> [,<对象类型><对象名>]...  
FROM <用户> [,<用户>]...[CASCADE|RESTRICT];
```

角色创建

Create Role <角色名>

角色授权

```
GRANT <权限> [,<权限>]...  
ON <对象类型>对象名  
TO <角色> [,<角色>]...
```

授权给他人

```
GRANT <角色 1> [,<角色 2>]...  
TO <角色 3> [,<用户 1>]...  
[WITH ADMIN OPTION]
```

收回角色授权

```
REVOKE <权限> [,<权限>]...  
ON <对象类型> <对象名>  
FROM <角色> [,<角色>]...
```

4. 强制存取控制

敏感度标记 $TS \geq S \geq C \geq P$

5. 视图机制

6. 审计

设法打破控制。审计功能把用户对数据库的所有操作自动记录下来放入审计日志（audit log）中。审计员可以利用审计日志监控数据库中的各种行为，重现导致数据库现有状况的一系列事件，找出非法存取数据的人、时间和内容等。还可以通过对审计日志分析，对潜

AUDIT设置审计功能 NOAUDIT取消审计功能

7. 数据加密

储存加密 传输加密

8. 其他安全性保护

推理控制 强制存取控制未解决的问题 利用能访问的数据推知更高机密的数据

隐蔽通道 强制存取控制未解决的问题 利用未被钱之光存取控制的SQL执行后反馈信息进行间接信息传递

数据隐私

五、数据库安全性

正确性 相容性

提供定义完整性约束的机制

提供完整性检查的方法

进行违约处理

1. 实体完整性 PRIMARY KEY

主键值是否唯一；主码各个属性是否为空

2. 参照完整性 REFERENCE

被参照表（例如 Student）	参照表（例如 SC）	违约处理
可能破坏参照完整性 ←	插入元组	拒绝
可能破坏参照完整性 ←	修改外码值	拒绝
删除元组 →	可能破坏参照完整性	拒绝/级联删除/设置为空值
修改主码值 →	可能破坏参照完整性	拒绝/级联修改/设置为空值

拒绝（NOACTION）执行级联（CASCADE）操作 设置为空

3. 用户定义完整性

属性约束条件

非空 NOT NULL 列值唯一 UNIQUE 满足某个条件表达式 CHECK

元组约束条件 CHECK

4. 完整性约束命名字句

```
CONSTRAINT <完整性约束条件名> <完整性约束条件>
```

<完整性约束条件>包括 NOT NULL、UNIQUE、PRIMARY KEY、FOREIGN KEY、CHECK 短语等。

ALTER TABLE 修改完整性限制

5. 断言

涉及关系的操作都会触发关系数据库对断言的检查

```
CREATE ASSERTION <断言名> <CHECK 子句>
```

每个断言都被赋予一个名字，<CHECK 子句>中的约束条件与 WHERE 子句的条件表达式类似。

```
DROP ASSERTION <断言名>
```

6. 触发器

事件-条件-动作规则

```
SQL 使用 CREATE TRIGGER 命令建立触发器，其一般格式为
```

```
CREATE TRIGGER <触发器名> /*每当触发事件发生时，该触发器被激活*/
```

```
{BEFORE | AFTER} <触发事件> ON <表名>
```

```
/*指明触发器激活的时间是在执行触发事件前或后*/
```

```
REFERENCING NEW|OLD ROW AS<变量> /*REFERENCING 指出引用的变量*/
```

```
FOR EACH {ROW | STATEMENT}
```

```
/*定义触发器的类型，指明动作体执行的频率*/
```

```
[WHEN <触发条件>] <触发动作体> /*仅当触发条件为真时才执行触发动作体*/
```

同一模式下触发器名唯一，触发器名和表名在同一模式下

只能定义在基本表上

触发事件可以是INSERT,DELETE和UPDATE及其组合 AFTER/BEFORE是触发时机

行级触发器（FOR EACH ROW）和语句级触发器（FOR EACH STATEMENT）

例子：


```

CREATE TRIGGER SC_T          /*SC_T 是触发器的名字*/
AFTER UPDATE OF Grade ON SC  /*UPDATE OF Grade ON SC 是触发事件，*/
/* AFTER 是触发的时机，表示当对 SC 的 Grade 属性修改完后再次触发下面的规则*/
REFERENCING
    OLDROW AS OldTuple,
    NEWROW AS NewTuple
FOR EACH ROW                /*行级触发器，即每执行一次 Grade 的更新，下面的规则就执行一次*/
WHEN (NewTuple.Grade >= 1.1 * OldTuple.Grade) /*触发条件，只有该条件为真时才执行*/
INSERT INTO SC_U (Sno,Cno,OldGrade,NewGrade) /*下面的 insert 操作*/
VALUES(OldTuple.Sno,OldTuple.Cno,OldTuple.Grade,NewTuple.Grade)

```

DROP TRIGGER <触发器名> ON <表名>

设计与应用开发

六、关系数据库的规范化理论

最基本条件：每个分量必须是不可分的数据项 第一范式 1NF

数据依赖：一个关系内部属性和属性之间的一种约束关系

函数依赖 FD 多值依赖 MVD

存在问题：数据冗余 更新/插入/删除异常

1. 规范化

函数依赖

术语：

- $X \rightarrow Y$ ，但 $Y \not\subseteq X$ ，则称 $X \rightarrow Y$ 是非平凡的函数依赖。

- $X \rightarrow Y$ ，但 $Y \subseteq X$ ，则称 $X \rightarrow Y$ 是平凡的函数依赖。对于任一关系模式，平凡函数依赖都是必然成立的，它不反映新的语义。若不特别声明，总是讨论非平凡的函数依赖。
- 若 $X \rightarrow Y$ ，则 X 称为这个函数依赖的决定属性组，也称为决定因素（determinant）。
- 若 $X \rightarrow Y$ ， $Y \rightarrow X$ ，则记作 $X \leftrightarrow Y$ 。
- 若 Y 不函数依赖于 X ，则记作 $X \nrightarrow Y$ 。

完全函数依赖 部分函数依赖 传递函数依赖

码

2. 一个低一级范式的关系模式通过模式分解可以转换为若干个高一级范式的关系模式的集合，这个过程就叫规范化

若 R 为 1NF，且每个非主属性完全函数依赖于任何一个候选码，则 R 属于 2NF

2NF 在 2NF 的基础之上，消除了非主属性对于码的传递函数依赖

3. BCNF

BC 范式在 3NF 的基础上消除主属性对于码的部分与传递函数依赖

定义

定义 6.8 关系模式 $R<U, F> \in 1NF$ ，若 $X \rightarrow Y$ 且 $Y \not\subseteq X$ 时 X 必含有码，则 $R<U, F> \in BCNF$ 。

也就是说，关系模式 $R<U, F>$ 中，若每一个决定因素都包含码，则 $R<U, F> \in BCNF$ 。

特征

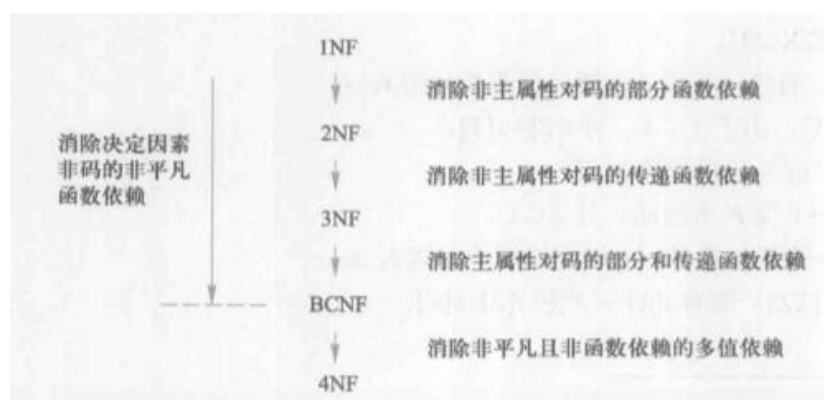
- 所有非主属性对每一个码都是完全函数依赖。
- 所有主属性对每一个不包含它的码也是完全函数依赖。
- 没有任何属性完全函数依赖于非码的任何一组属性。

4. 多值依赖

一个关系，至少存在三个属性（A、B、C），才能存在这种关系。对于每一个A值，有一组确定的B值和C值，并且这组B的值独立于这组C的值。

第四范式的定义很简单：已经是BC范式，并且不包含多值依赖关系。

除了第四范式外，我们还有更高级的第五范式和域键范式（DKNF），第五范式处理的是无损连接问题，这个范式基本没有实际意义，因为无损连接很少出现，而且难以察觉。而域键范式试图定义一个终极范式，该范式考虑所有的依赖和约束类型，但是实用价值也是最小的，只存在理论研究中。



5. Armstrong公理系统

Armstrong 公理系统 (Armstrong's axiom) 设 U 为属性集总体， F 是 U 上的一组函数依赖，于是有关系模式 $R<U, F>$ ，对 $R<U, F>$ 来说有以下的推理规则：

- A1 自反律 (reflexivity rule): 若 $Y \subseteq X \subseteq U$ ，则 $X \rightarrow Y$ 为 F 所蕴涵。
 - A2 增广律 (augmentation rule): 若 $X \rightarrow Y$ 为 F 所蕴涵，且 $Z \subseteq U$ ，则 $XZ \rightarrow YZ$ 为 F 所蕴涵。
 - A3 传递律 (transitivity rule): 若 $X \rightarrow Y$ 及 $Y \rightarrow Z$ 为 F 所蕴涵，则 $X \rightarrow Z$ 为 F 所蕴涵。
- 注意：由自反律所得到的函数依赖均是平凡的函数依赖，自反律的使用并不依赖于 F 。

最小依赖

定义 6.15 如果函数依赖集 F 满足下列条件，则称 F 为一个极小函数依赖集，亦称为最小依赖集或最小覆盖 (minimal cover)。


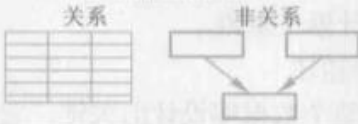
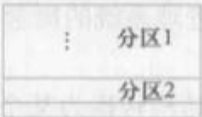
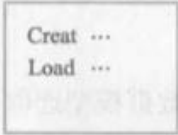
- (1) F 中任一函数依赖的右部仅含有一个属性。
- (2) F 中不存在这样的函数依赖 $X \rightarrow A$ ，使得 F 与 $F - \{X \rightarrow A\}$ 等价。
- (3) F 中不存在这样的函数依赖 $X \rightarrow A$ ， X 有真子集 Z 使得 $F - \{X \rightarrow A\} \cup \{Z \rightarrow A\}$ 与 F 等价。

6. 模式分解算法 (先略过了)

七、数据库设计

1. 六个阶段：

需求分析 概念结构设计 逻辑结构设计 物理结构设计 数据实施 数据库运行和维护

设计阶段	设计描述
需求分析	数据字典、全系统中数据项、数据结构、数据流、数据存储的描述
概念结构设计	概念模型 (E-R 图)  数据字典
逻辑结构设计	某种数据模型 
物理结构设计	存储安排 存取方法选择 存取路径建立 
数据库实施	创建数据库模式 装入数据 数据库试运行 
数据库运行和维护	性能监测、转储/恢复、数据库重组和重构

2. 需求分析

数据字典 数据项 数据结构 数据流 数据存储 处理过程

3. 概念结构设计

E-R模型 一对一 一对多 多对多

实体矩形 属性椭圆形 联系菱形

UML语言

4. 逻辑结构设计

5. 物理结构设计 B+树 hash索引

6. 数据实施和维护

八、数据库编程

1. 嵌入式SQL

主变量 游标 (数据缓冲区) 建立关闭数据库连接

2. 动态SQL

3. ODBC 类比一下 JDBC

九、关系查询处理和查询优化 (略过 根据之后的题目有需要再补充)

十、数据库恢复技术

1. 事务 数据库操作序列

原子性Atomicity 一致性Consistency 隔离性Isolation 持续性Durability ACID特性

2. 事务内部故障 事务撤销 UNDO

系统故障 重启 事务重做REDO

介质故障

计算机病毒

3. 数据转储 登记日志文件
4. 检查点 动态维护日志
5. 数据库镜像

十一、并发控制

1. 事务是并发的基本单位
为了保证事务的隔离性和一致性
2. 丢失修改
不可重复读
读“脏”数据

T ₁	T ₂	T ₁	T ₂	T ₁	T ₂
① R(A)=16		① R(A)=50 R(B)=100 求和=150		① R(C)=100 C=C*2 W(C)=200	
②	R(A)=16	②	R(B)=100 B=B*2 W(B)=200	②	R(C)=200
③ A=A-1 W(A)=15		③ R(A)=50 R(B)=200 求和=250 (验算不对)		④ ROLLBACK C恢复为100	
④	A=A-1 W(A)=15				
(a) 丢失修改		(b) 不可重复读		(c) 读“脏”数据	

3. 封锁
排他锁 写锁 X
共享锁 读锁 S

T ₁ \ T ₂	X	S	-
X	N	N	Y
S	N	Y	Y
-	Y	Y	Y

Y=Yes, 相容的请求
N=No, 不相容的请求

封锁协议

- 一级 修改数据之前必须加X锁，事务结束释放
 - 二级 读取数据前必须加S锁，读完释放 + 一级
 - 三级 读取数据前必须加S锁，事务结束释放 + 一级
- 协议越高，一致性程度越高

表 11.1 不同级别的封锁协议和一致性保证

	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
一级封锁协议		✓			✓		
二级封锁协议		✓	✓		✓	✓	
三级封锁协议		✓		✓	✓	✓	✓

4. 活锁 先来先服务
死锁

一次封锁 将所有要用的数据都加锁

顺序封锁

超时法

等待图法

5. 可串行化 多个事务并发执行正确，当且仅当其结果与某一次序串行地执行这些事务的结果相同

充分条件 冲突可串行化

充分条件 两段锁协议 2PL

6. 多粒度封锁协议

意向锁

IS锁 IX锁 SIX锁

$T_1 \backslash T_2$	S	X	IS	IX	SIX	-
S	Y	N	Y	N	N	Y
X	N	N	N	N	N	Y
IS	Y	N	Y	Y	Y	Y
IX	N	N	Y	Y	N	Y
SIX	N	N	Y	N	N	Y
-	Y	Y	Y	Y	Y	Y

Y=Yes, 表示相容的请求 N=No, 表示不相容的请求

十二、数据库管理系统（略过）

之后是新技术领域 具体技术根据面试题准备