复旦大学计算机科学技术学院

2021-2022 第二学期《数据库引论》期末考试试卷

A 卷　共 6 页

课程代码：COMP130010.03　　考试形式：□开卷　☑闭卷　　2022 年 6 月

（本试卷答卷时间为 120 分钟，答案必须写在试卷上，做在草稿纸上无效）

专业_____学号_____姓名_____成绩_____

题号	一	二	三	四	五	六	七	八	九	十	总分
得分											

（装订线内不要答题）

一、单项选择题（每题 2 分，共 12 分）

1. 在数据库应用中，常用术语 DB、DBS 和 DBMS 三者的关系哪个是正确的：（　　）

   A、DBMS 包括 DB 和 DBS　　B、DBS 与 DB 等价，包含于 DBMS

   C、DBS 包括 DB 和 DBMS　　D、DB 包括 DBS 和 DBMS

2. 若关系模式 R 中没有非主属性，则：（　　）

   A、R 属于 2NF，但 R 不一定属于 3NF。

   B、R 属于 3NF，但 R 不一定属于 BCNF。

   C、R 属于 BCNF，但 R 不一定属于 4NF。

   D、R 属于 4NF。

3. 下面关于数据库中视图和游标的说法错误的是：（　　）

   A、视图可以提供一定程度的数据独立性。

   B、视图可以进行增改查等操作，通常视图是一个表或者多个表的行或列的子集，对视图的修改会影响到基本表。

   C、游标可以定在查询结果集的特定行，也可以从结果集的当前行检索一行或多行。

   D、使用游标的顺序一般为：游标定义（DECLARE）、游标打开（OPEN）、游标推进（FETCH）、游标关闭（CLOSE）。

4. 以下关于事务和锁的说法中错误的是：（　　）

   A、时标顺序协议能保证调度是可串行化的。

   B、若事务 T 对数据对象 A 加上 S 锁，则事务 T 可以读 A，其他事务能对 A 加 S 锁和 X 锁。

   C、封锁对象的大小称为封锁的粒度，封锁带来的“死锁”问题可以用“定期检测和解除”方式解决。

D、当执行了 COMMIT 操作后，数据不一定会即刻写入磁盘。

5. 下列关于数据库查询优化正确的策略是：（　　）

A、尽可能早地执行笛卡尔积操作。

B、尽可能早地执行并操作。

C、尽可能早地执行差操作。

D、尽可能早地执行选择操作。

6. 设一个关系有 B 页，每页有 R 条记录，读或写一页的平均时间为 D，其他时间忽略不计。rid 的长度是记录长度的 10%。对于树索引，扇出为 F，节点占满率（Occupancy）为 66.67%。假设已知查询键值不重复（最多只有一条记录满足等值查询条件），则下列说法错误的是：（　　）

A、对于堆文件（Heap File）的等值查询，平均代价为 $0.5BD$。

B、对于排序文件（Sorted File）的等值查询，平均代价为 $0.5BD$。

C、对于使用树索引的簇聚文件（Clustered File）的等值查询，平均代价为 $D\log_F 1.5B$。

D、对于带有非簇聚树索引的堆文件的等值查询，平均代价为 $D\log_F 0.15B+D$。

二、不定项选择题（下列每题的各选项中，有一个或多个选项正确。每题 4 分，共 12 分。少选给 2 分，多选或错选不给分。）

1. 以下关于计算机存储设备的说法中正确的是：（　　）

A、内存、固态硬盘是一级存储设备。

B、机械硬盘是二级存储设备。

C、光盘、磁带是三级存储设备。

D、非易失性存储设备在掉电后不会保存数据。

2. 以下关于索引的说法中正确的是：（　　）

A、ISAM 索引是静态索引，由于索引结构不会被修改，因此相比 B+树索引并发性更好。

B、线性哈希索引与 ISAM 索引一样，会因为溢出页过多导致性能退化。

C、对于可扩展哈希索引，若插入元素导致桶溢出并触发目录翻倍，重新分配（溢出）桶内元素后，不再需要溢出页。

D、相比树索引，哈希索引不能用于范围查询。

3. 以下关于数据库故障恢复的说法中正确的是：（　　）

A、使数据库具有可恢复性的基本原则是“冗余”。

B、如果数据库遇到灾难性故障，就必须利用日志库“重做”已提交的事务，把数据库恢复到故障前的状态。

C、如果数据库未遭到物理性破坏，但破坏了数据库的一致性，此时需要利用日志库“撤销”所有不可靠的修改，再利用日志库“重做”已提交的、但对数据库的更新可能还留在缓冲区

的事务，就可以把数据库恢复到正确的状态。

D、事务故障是一种常见的数据库故障，事务故障的恢复需要 DBA 配合执行。

三、解答题（76 分。其中第 1 题 9 分，第 2 题 9 分，第 3 题 12 分，第 4 题 12 分，第 5 题 15 分，第 6 题 8 分，第 7 题 11 分）

1. 假设员工数据库中有 3 个关系：

   职工关系 EMP（E#，ENAME，AGE，SEX，ECITY）

   工作关系 WORKS（E#，C#，SALARY）

   公司关系 COMP（C#，CNAME，CITY）

   使用 **SQL 语句**写出以下操作（每题只能用一条语句作答，并给出必要文字说明）：

   (1) 请写出工作关系 WORKS 的建表语句，注意指出主键和外键。（3 分）

   (2) 请为工作关系 WORKS 增加入职时间 TIME 字段（类型为 TIMESTAMP）。（1 分）

   (3) 请为在 HUAWEI 公司工作的年龄不超过 35 岁的男性员工加薪 10%。（2 分）

   (4) 检索这样的职工工号和姓名，该职工至少在职工 E6（工号）兼职的所有公司兼职。（3 分）

2. 假设教学数据库中有 4 个关系：

   学生关系 S（S#，SNAME，AGE，SEX）

   选课关系 SC（S#，C#，SCORE）

   课程关系 C（C#，CNAME，T#）

   教师关系 T（T#，TNAME，TITLE）

   回答以下问题（每题只能用一条语句作答，并给出必要文字说明）：

   (1) 请用**元组表达式**写出：检索这样的学生学号和姓名，该学生所学课程包含学号为 s1 的同学所学所有课程。（2 分）

   (2) 请用 **SQL 语句**写出：检索这样的课程号和课程名，该课程被至少 10 位男同学选修。（2 分）

   (3) 请用 **SQL 语句**写出：检索这样的学生学号和姓名，该学生所有选修课程的得分分别高于这些课程的平均分。举例来说，对于某个选修了 A 和 B 两门课程的学生，若该学生在 A 课程上的得分高于所有选修 A 课程学生的平均得分，且该学生在 B 课程上的得分高于所有选修 B 课程学生的平均得分，则该学生满足检索条件。（5 分）

3. 查找键集合为{90, 10, 80, 20, 30, 70, 40, 60, 100, 50}，请完成以下操作（**完整**画出最终结果，并保留必要中间步骤，如节点分裂、合并前后）：

(1) 按所给的插入顺序，建立秩为 1 的 B+树（除根节点外的节点可保存 1 个或 2 个键）。（3 分）

(2) 在(1)的基础上，画出依次插入 43、47 后的 B+树。（3 分）

(3) 在(2)的基础上，画出依次删除 43、47 后的 B+树。（3 分）

(4) 在(3)的基础上，画出删除 40 后的 B+树。（3 分）

4. 下面给出了 B+树搜索操作的核心代码，*nodepointer 中含有 m 个键，键值分别是 K_1~K_m，子树指针为 P_0~P_m。假设页的大小是 4096 字节，子树指针大小为 8 字节，rid 大小为 10 字节，B+树（除根节点外的其他节点）占满率为 66.67%（所有节点的占满率均不超过 66.67%）。总共 M 条数据记录（M 的规模在 10^6 数量级），键的大小为 S 字节（$0<S\leq 2036$）。为简化问题，B+树节点中的头部信息、控制信息大小忽略不计；B+树的叶节点与非叶节点结构不同，叶节点上仅保存记录的 rid，不包含键、子树指针、兄弟指针等其他字段；对于升序遍历，每个非叶节点（平均）遍历 $m/2$ 个键即可完成定位；对于二分搜索，按最坏情况计算。（**要求给出具体的计算过程和必要的说明，仅给出答案不得分**）

```
    func tree_search(nodepointer, search key K) returns nodepointer
1   if *nodepointer is a leaf, return nodepointer;
2   else,
3     if K < K1 then return tree_search(P0, K);
4     else,
5       if K ≥ Km then return tree_search(Pm, K);
6       else,
7         find i such that Ki ≤ K < Ki+1;   // 定位操作
8         return tree_search(Pi, K);
    endfunc
```

(1) 对于第 7 行的定位操作，分别对比在以下情况中使用升序遍历与二分搜索的 I/O 开销：（6 分）

(i) 非叶节点上直接保存键 K_i 的具体值；

(ii) 非叶节点上只保存键 K_i 的 rid，即需要额外一次 I/O 才能得到 K_i 的具体值（用于比较大小）。

提示：本题结果是含有参数 M 和 S 的表达式，注意写明表达式中（向上/下）取整符号（向上取整符号：$\lceil a \rceil$表示大于等于a的最小整数；向下取整符号：$\lfloor a \rfloor$表示小于等于a的最大整数）和对数的底数。

(2) 只考虑用二分搜索进行定位，讨论键的大小为何值时，第(1)题第(ii)种方案比第(i)种方案开销更小。（6 分）

（装订线内不要答题）

5. 考虑两个关系 R 和 S，其中 R 共有 20,000 条记录，数据文件每页包含 10 条记录；S 共有 5000 条记录，数据文件每页包含 10 条记录。R 和 S 均存储于简单堆文件，且数据文件（堆文件）尽量存满。

对于连接操作$R \bowtie_{R.eno=S.b} S$（可根据需求任选 R 或 S 为外关系），其中属性 b 为 S 的主键。假设有 36 个可用的缓冲区页，R 和 S 上没有建立任何索引。

在不考虑输出查询结果所需 I/O 操作的情况下，请回答以下问题。（**要求给出具体的操作过程和必要的说明，仅给出答案不得分**）

(1) 使用“块嵌套循环算法”,“归并排序算法”进行 R 和 S 的连接操作，请分别计算它们至少需要多少次 I/O 操作，以及如果要保持开销不变，至少需要多少个缓冲区页。 （6 分）

(2) 当采用“哈希连接”（分为 Partition 阶段和 Probing 阶段）的方法进行 R 和 S 表的连接操作，请计算需要多少次 I/O 操作？要确保这样的哈希连接能成功进行至少需要多少个缓冲区页？（若为 N 页数据建立内存哈希表，需要 $f \times N$ 个缓冲区页，其中 f 称为“经验系数”或称为“数据分布的偏差参数”） （2 分）

(3) 使用任意一个连接方法进行 R 和 S 表的连接操作，让 I/O 开销达到最小，最少的 I/O 操作次数是多少？以及需要多少个缓冲区页来完成这样的连接操作？请做出简单的解释。（3 分）

(4) 若 R 是一个员工信息关系：

R(eno: char(100), ename: char(100), title: char(100), department: char(100))

每条记录的大小为 400 字节，eno 为候选键。存在以下索引（均使用方法 II，即每个数据项形如：<key, rid>，rid 大小忽略不计）：department 上的非簇聚 B+树索引，以及(title, department)上的簇聚 B+树索引。为简化问题，假设 B+树的占满率为 100%，在 B+树中单次查找适当的叶节点的 I/O 开销为 2，控制信息、指针大小均忽略不计。考虑以下查询：

select eno, ename from R where title like ‘Senior%’ and department = ‘Research & Development’ （假设 title 满足条件的有 10%，department 满足条件的有 20%，同时满足条件的有 5%）

选用何种读取路径最好？至少需要多少次 I/O？ （4 分）

6. 简要回答以下问题：

(1) 介绍一个缓冲区的常用置换策略与实现思路。（4 分）

(2) 讨论定长记录与变长记录在数据库底层存储上的区别与联系。（4 分）

7. 考虑对以下事务 T1，T2 和 T3 的并发调度。

数据库中 A 的初始值为 100；读操作（FIND）表示从数据库中读值，括号内显示读取到的

值；写操作（:=）修改并更新对象的值。

时间	事务 T1	事务 T2	事务 T3
t1	FIND A (100)		
t2		FIND A (100)	
t3	A : = A - 30		
t4		A : = A * 2	
t5			FIND A (200)
t6			FIND A (200)
t7		*ROLLBACK*	
t8			A : = A * 3
t9			*COMMIT*
t10	*COMMIT*		

(1) t10 以后，A 的值是多少？（1 分）

(2) 该数据库事务隔离级别最高是什么？请简要给出判断依据。并简述该调度引发的问题。（4 分）

(3) 考虑以下调度，已知该数据库中事务的隔离级别是读提交，请为读操作（FIND）、写操作（:=）与提交操作（*COMMIT*）设计一种可行的加锁、解锁方案。（6 分）

提示：有 S 锁（读锁）与 X 锁（写锁），加锁操作为 LOCKS A 与 LOCKX A，解锁操作为 UNLOCKS A 与 UNLOCKX A。

时间	事务 T1	事务 T2	事务 T3
t1	FIND A (100)		
t2	A : = A - 30		
t3	*COMMIT*		
t4			FIND A (70)
t5		FIND A (70)	
t6		A : = A * 2	
t7			FIND A [WAIT]
t8		*COMMIT*	
t9			FIND A (140)
t10			A : = A * 3
t11			*COMMIT*

说明：t7 时刻，事务 T3 的 FIND A 操作被阻塞，直到 t9 时刻 FIND A 返回结果。