**概念模式**使用低级DDL描述DB的整体/全局逻辑结构。这只是一个逻辑模型。它属于机器世界。

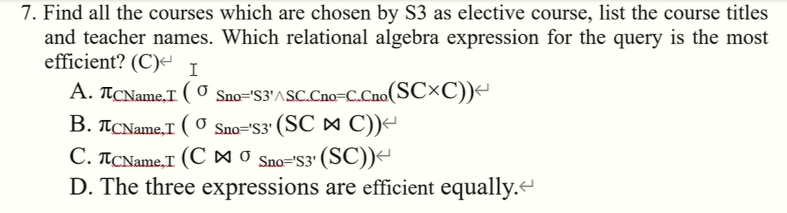
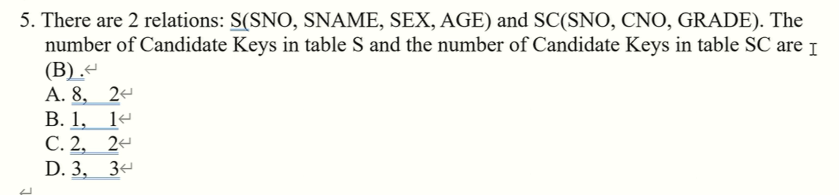
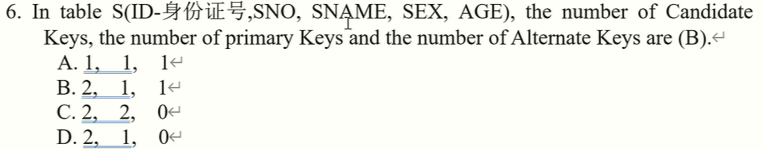
**概念模型**用一种更高级的方法描述数据库的整体/全局逻辑结构，便于用户理解。它是一种独立于计算机系统，由人脑抽象出来的模型。它属于信息世界，与特定的DBMS无关。它有明确的语义。它是系统设计者和用户之间的沟通工具。

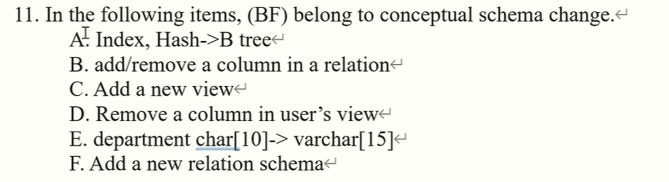
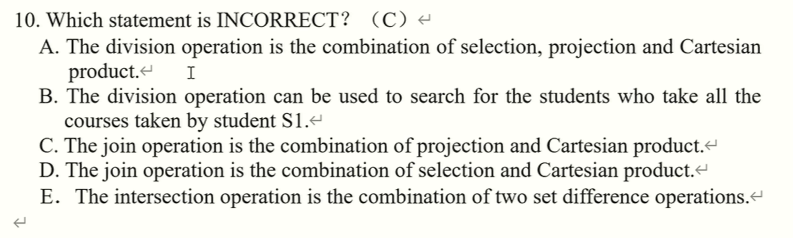
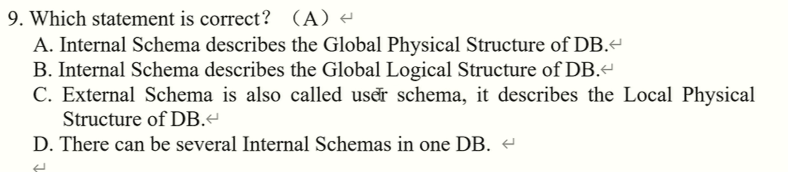
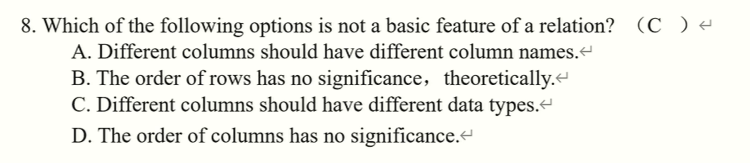
它们的共同点在于它们描述相同的对象。

不同之处在于它们的描述方法不同

## 小测屏幕截图 2024-06-16 163248屏幕截图 2024-06-16 163304屏幕截图 2024-06-16 163311

(SNO,CNO)才能作为主键

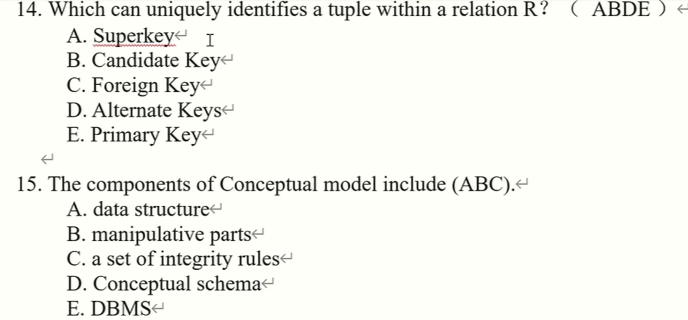


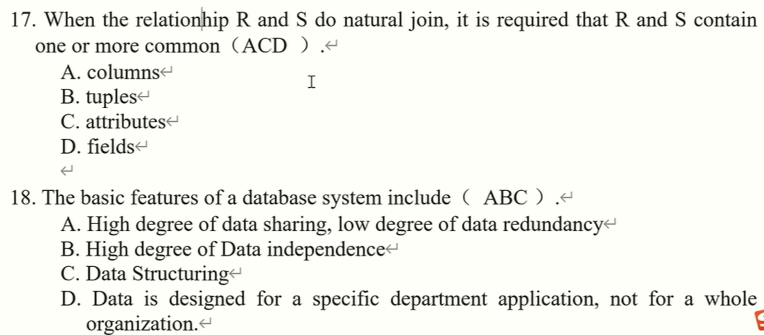
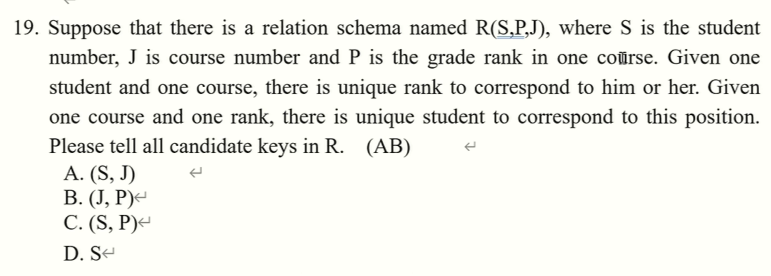
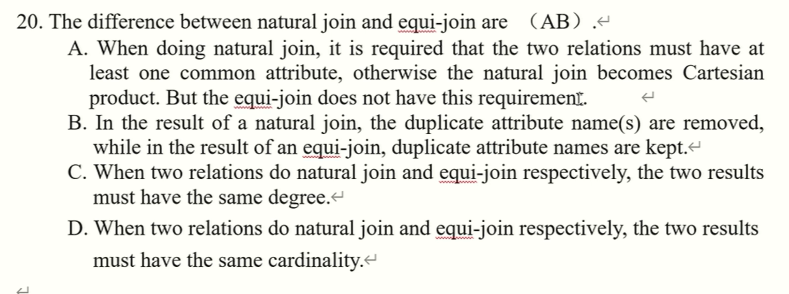
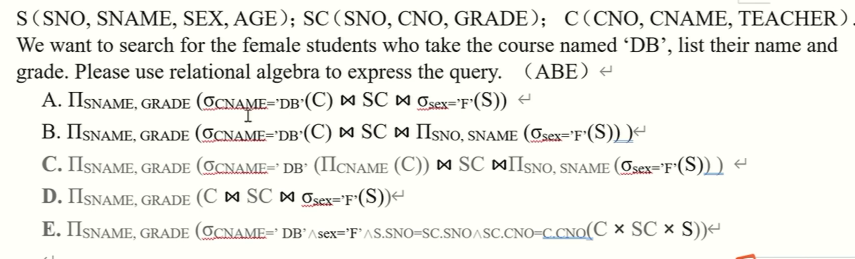


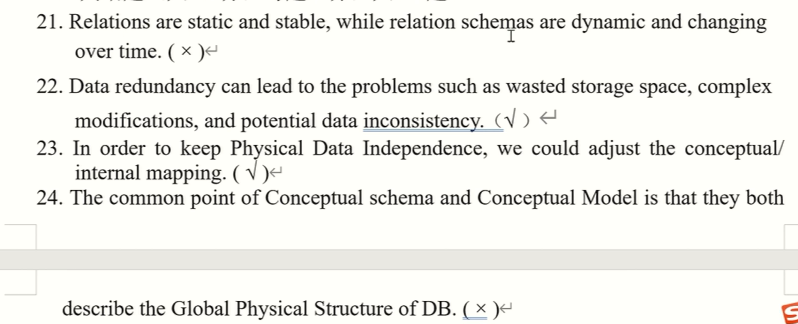
**In the following items, (BF) belong to conceptual schema change**

In the following items, (AE ) belong to internal schema change.

In the following items, (CD ) belong to external schema change

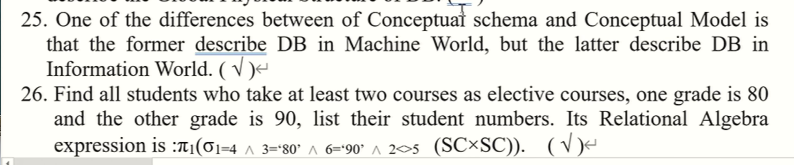


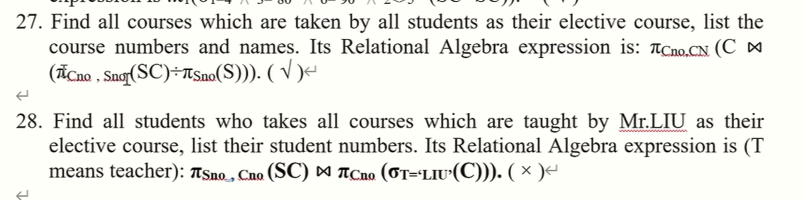


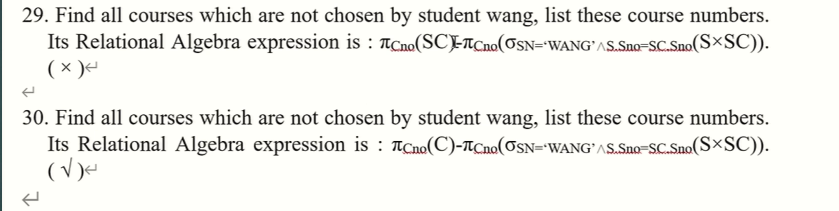


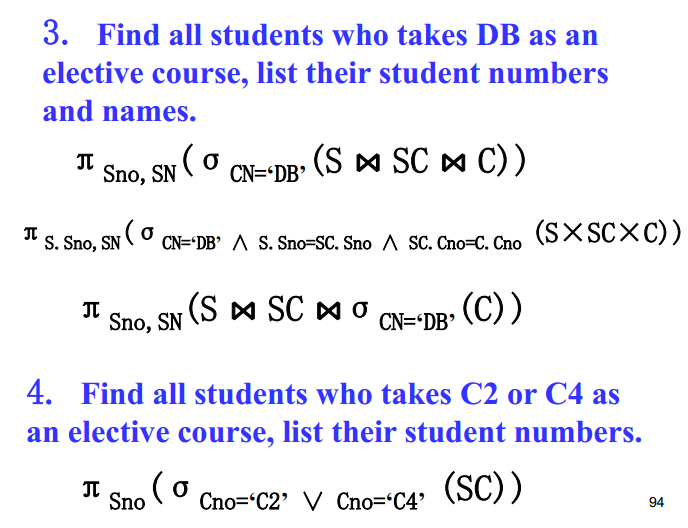
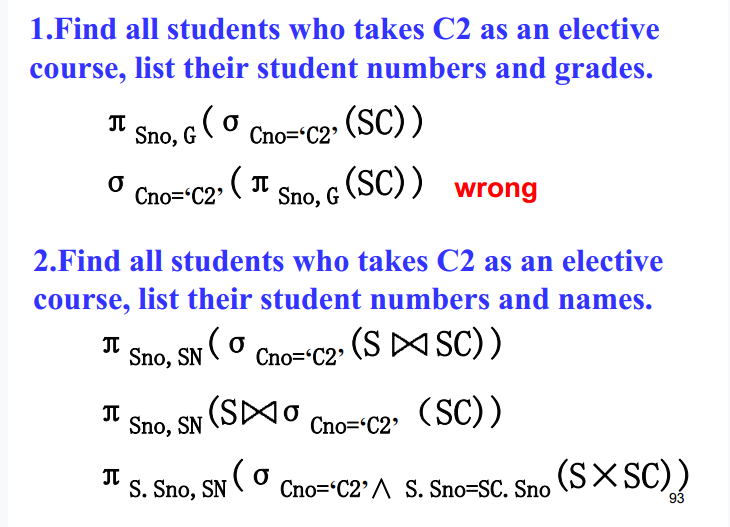
24：

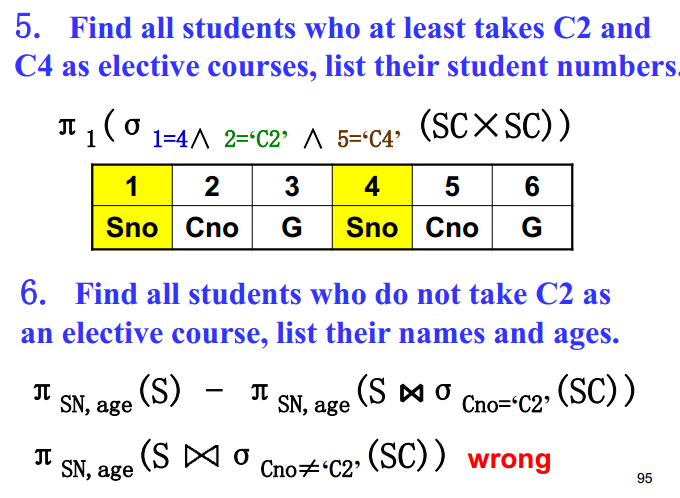
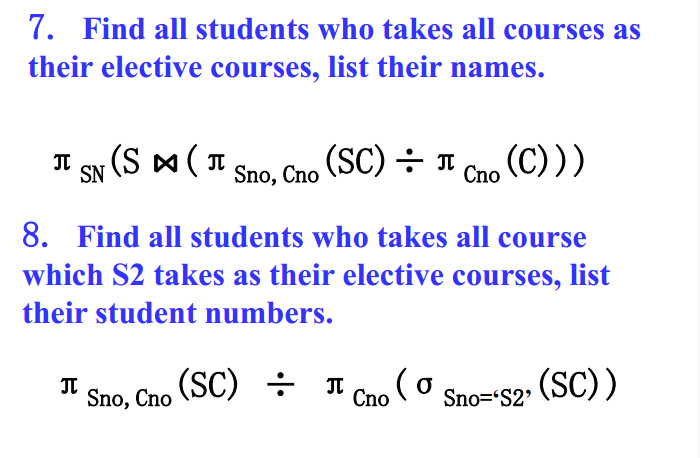
概念模式（Conceptual Schema）和概念模型（Conceptual Model）的共同点在于它们都描述了数据库的整体逻辑结构，而不是物理结构。概念模式通常指的是数据库的整体逻辑设计，它定义了数据库中的数据对象、关系以及数据的完整性约束等。而概念模型则是一个更抽象的概念，它是对现实世界中的实体、属性以及它们之间关系的抽象表示



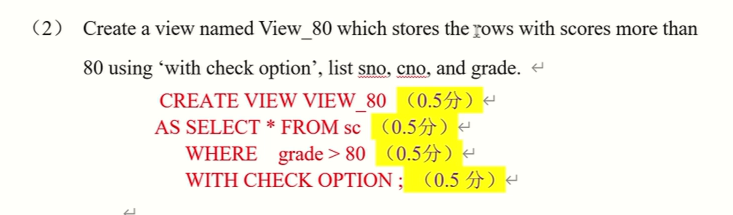
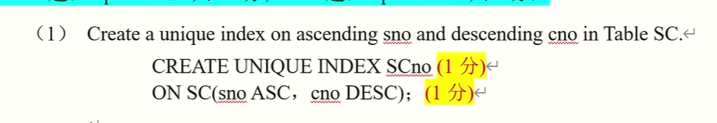


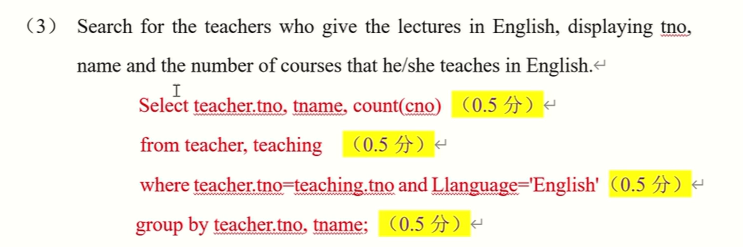


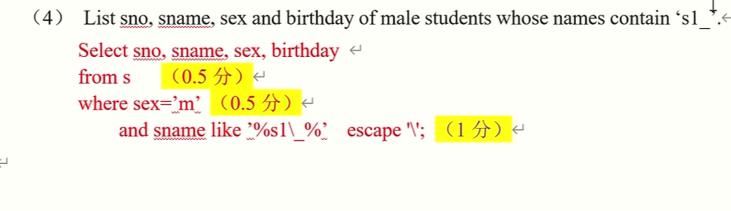


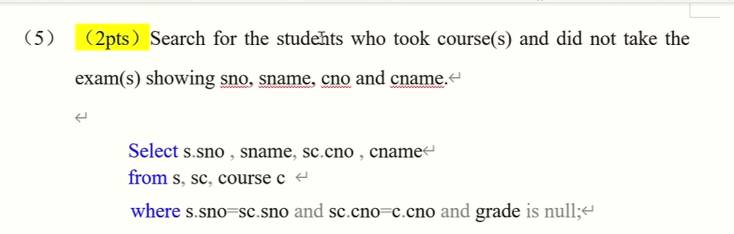


第二次小测

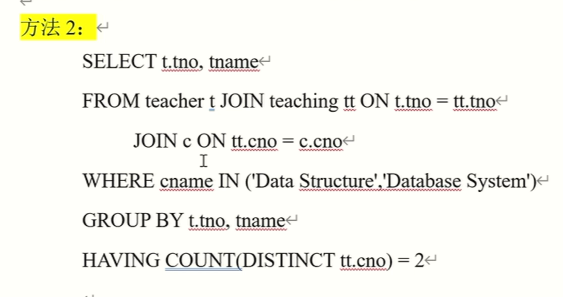
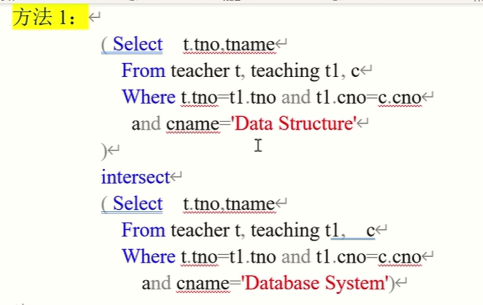


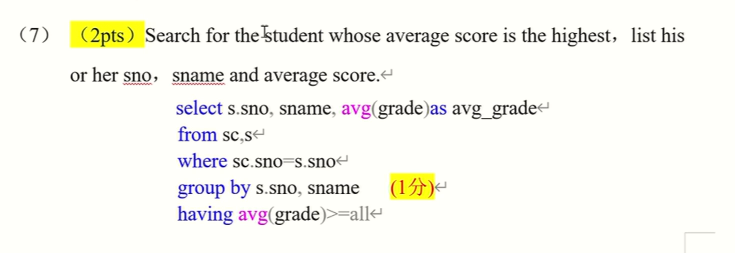


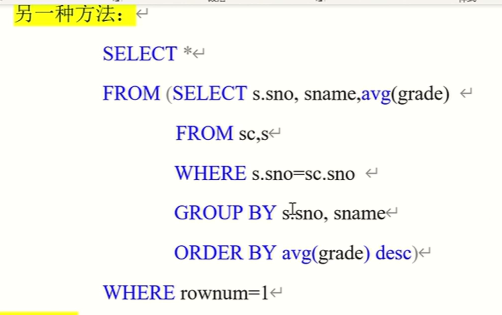


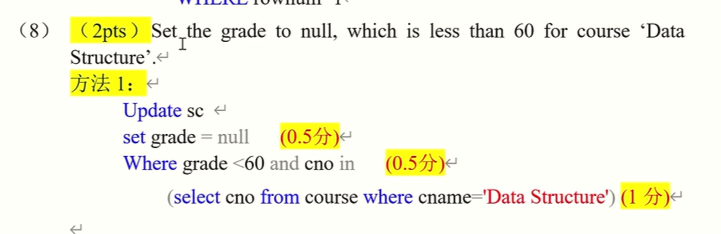


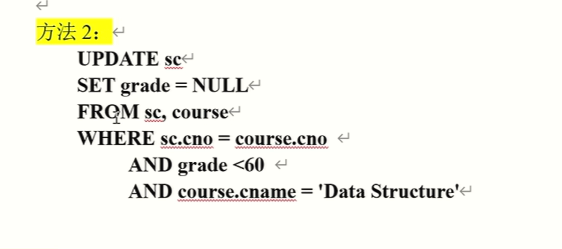


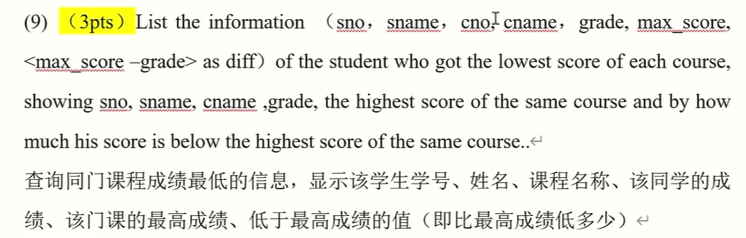


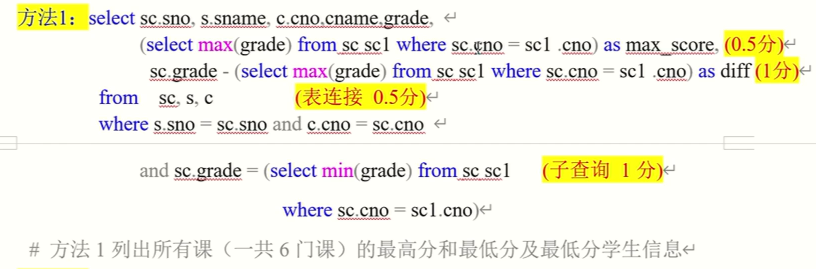


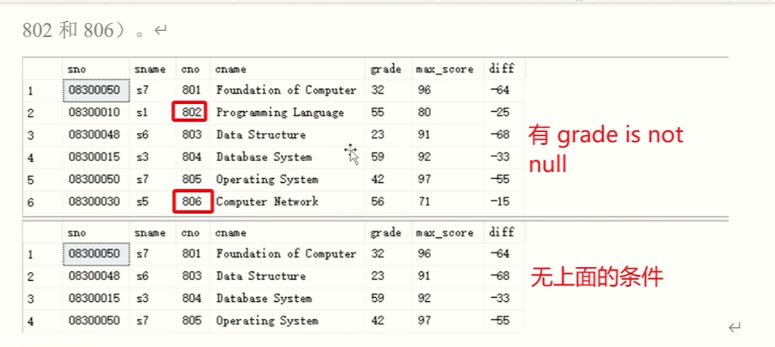


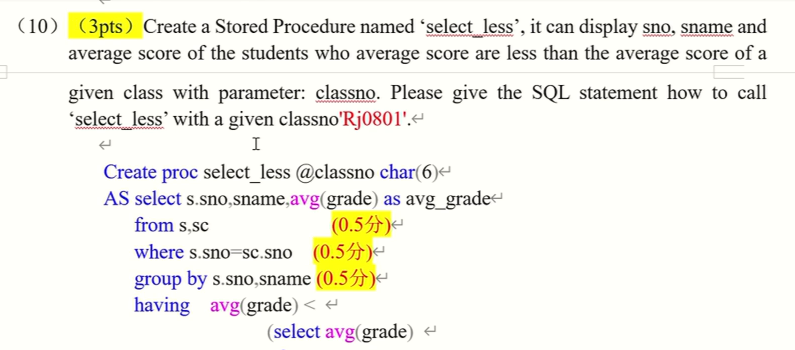
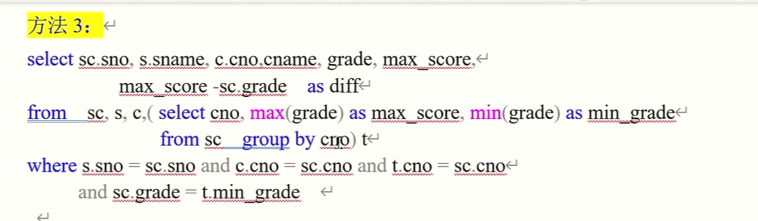


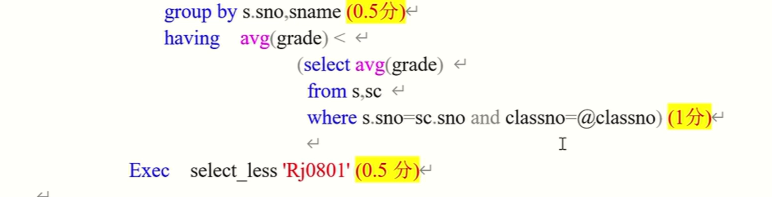


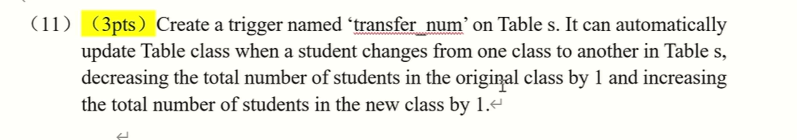


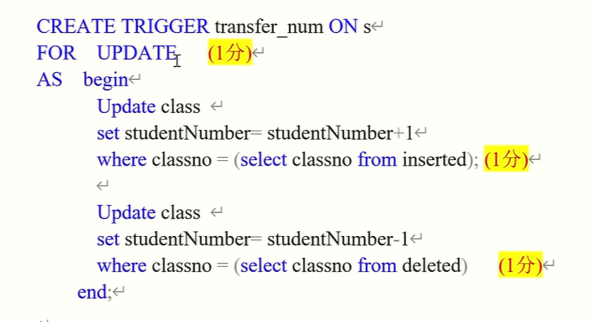


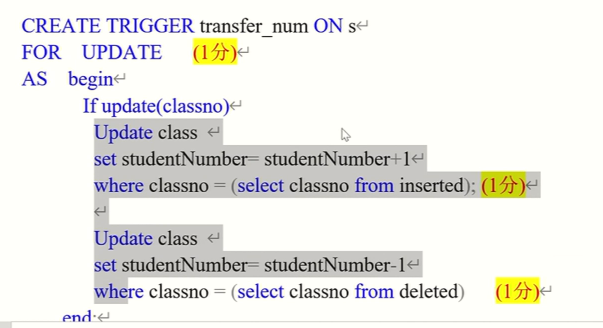


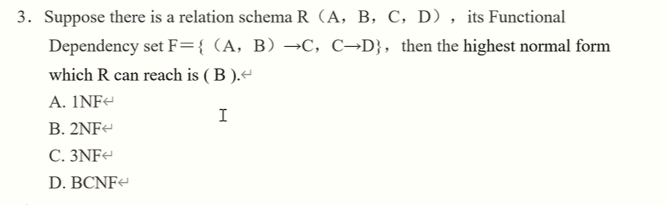
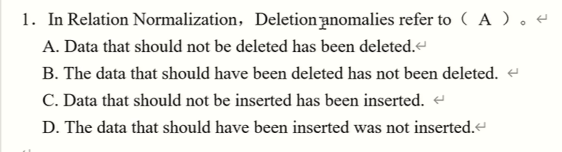


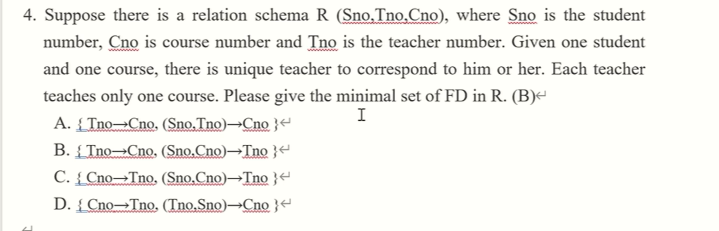


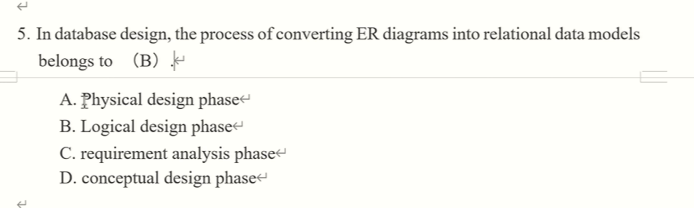


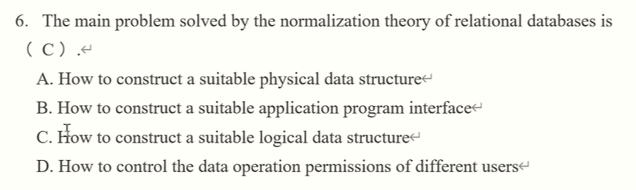


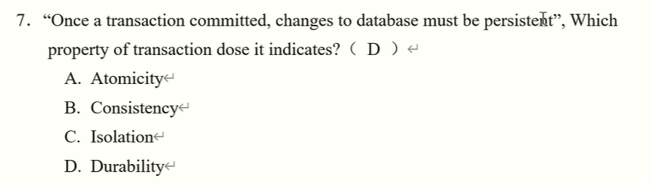


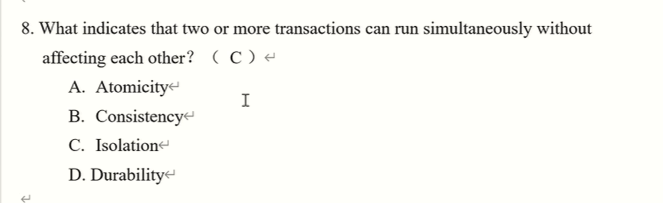


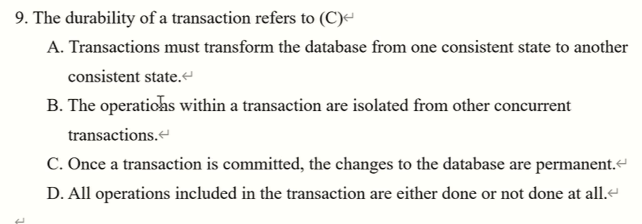


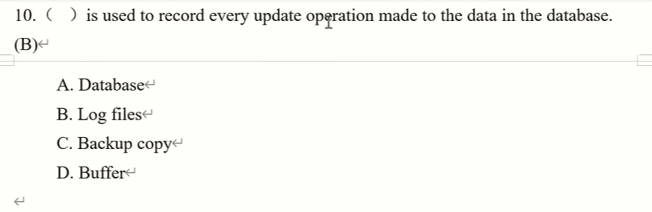


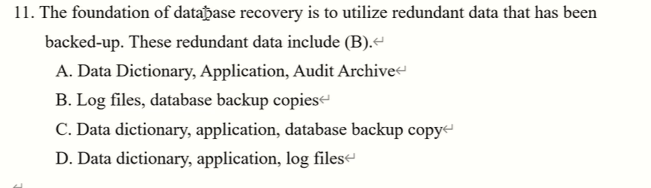


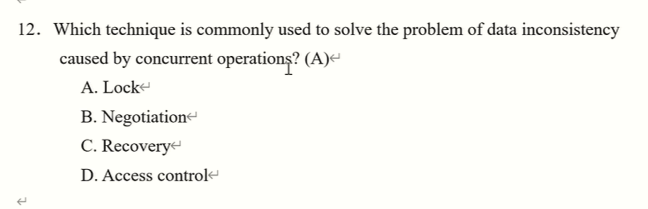


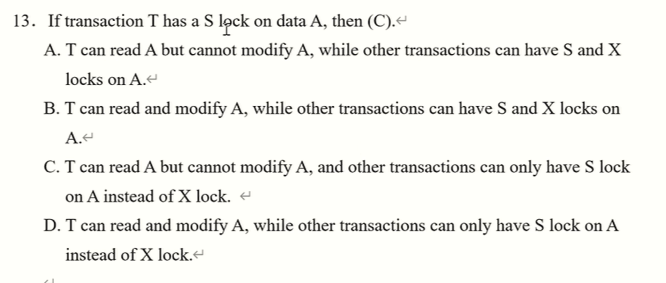


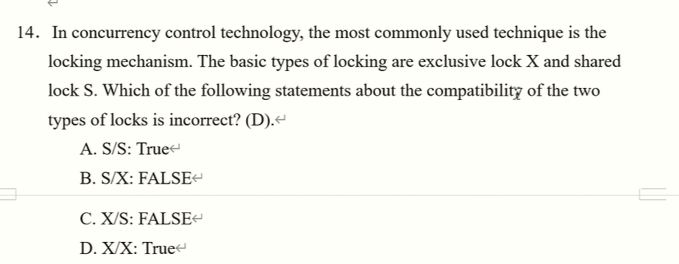


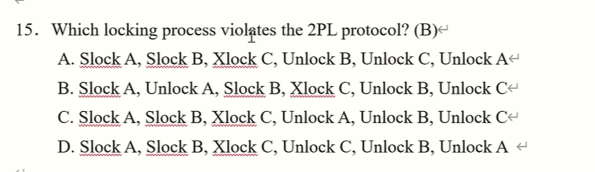


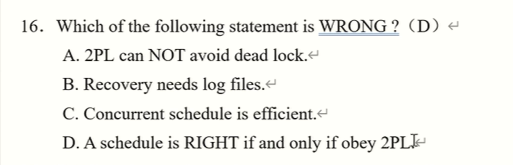


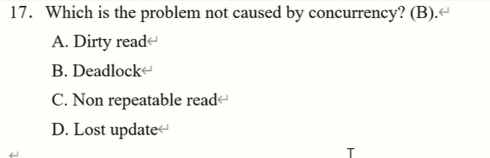




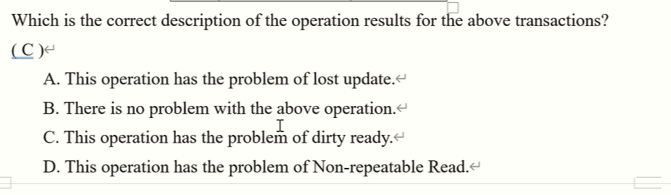


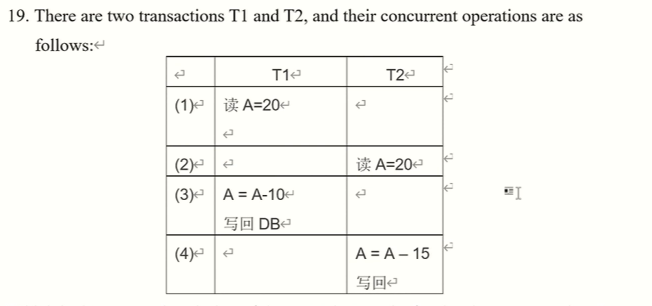


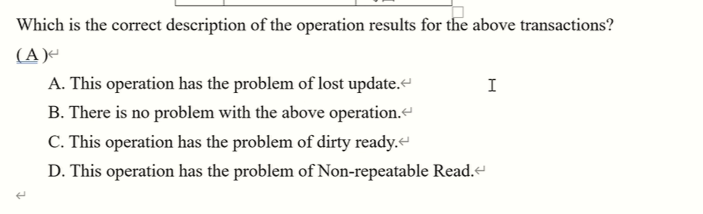


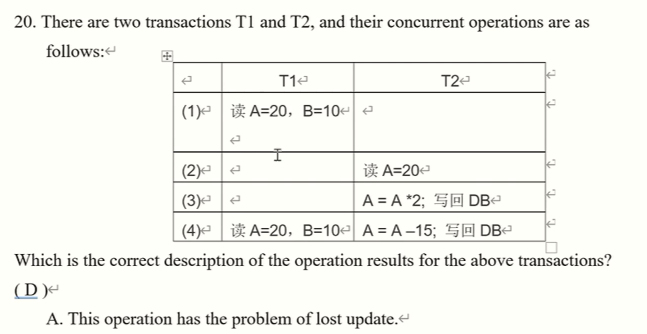


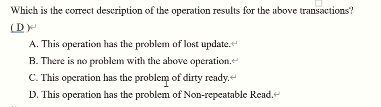


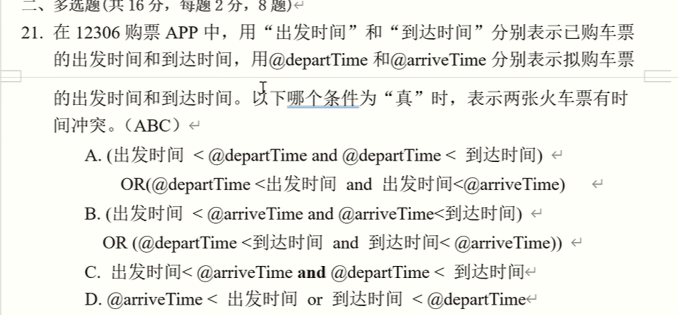


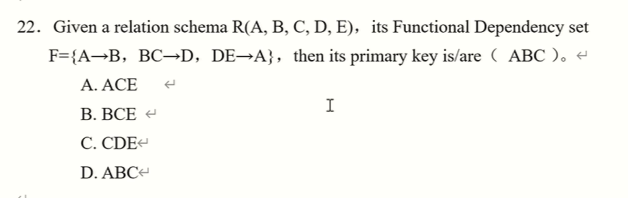


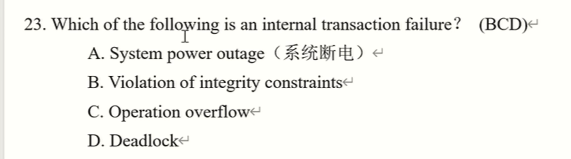




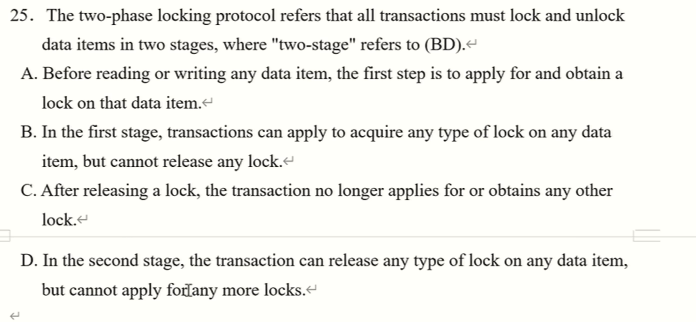


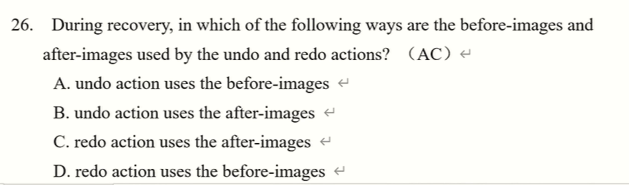


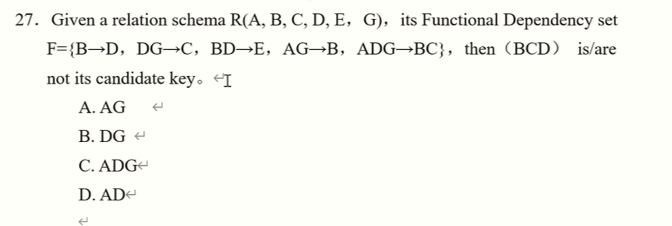


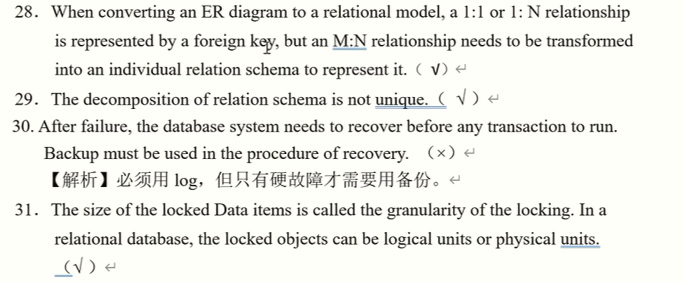


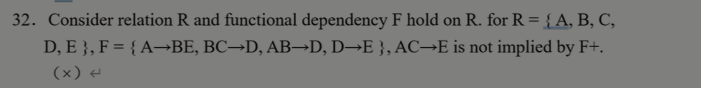


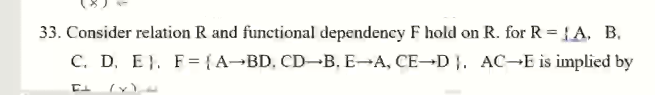












概念：

**字面量**是SQL语句中使用的常量，所有非数字字面值必须用单引号括起来(例如' London ')。

**集群索引**的顺序与数据文件中逻辑记录的物理存储顺序相同，而非集群索引的顺序与数据文件中逻辑记录的物理存储顺序无关

索引机制用于加快对所需数据的访问，就像在书中添加索引一样。

包含逻辑记录的文件称为数据文件，包含索引记录的文件称为索引文件。

一个索引文件由索引记录(称为索引条目)组成，包含两个字段:Search Key和pointer，形式如下:Search Key pointer

Search Key是用于查找文件中记录的一个属性或一组属性。pointer是文件中包含Search key值的逻辑记录的地址

索引文件中的索引记录是根据索引字段排序的，索引字段通常是单个属性。索引文件通常比数据文件小得多。

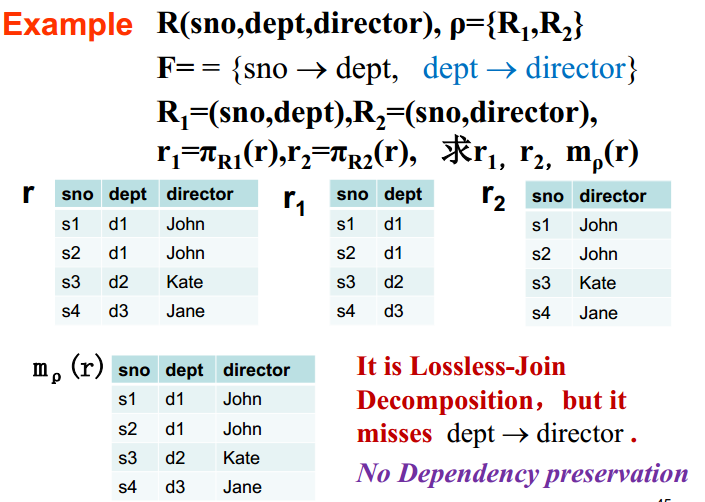
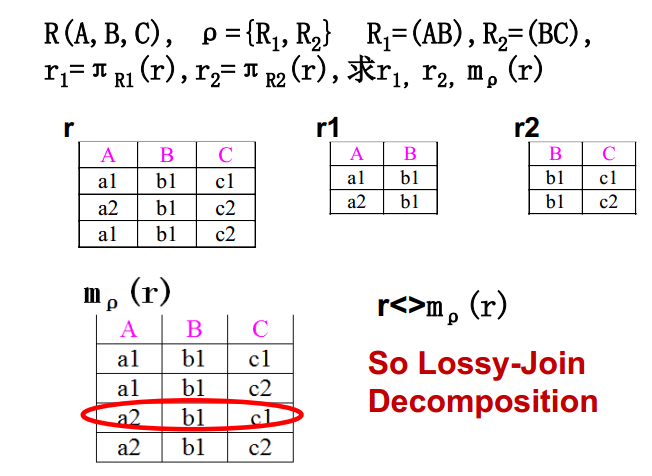
排序与索引的主要区别”:（1）排序要生成新表文件；索引只生成索引文件。（2）排序生成的新表记录重新调整改变了，

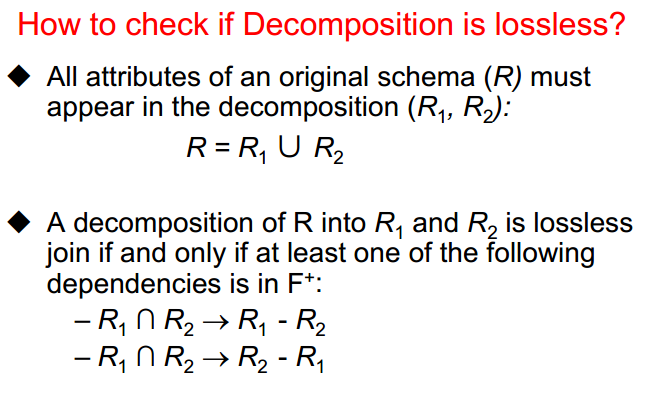
索引不改变原表中记录号。（3）排序生成的新表可以单独使用，索引文件不能单独使用（只有在表文件打开时才能使用）

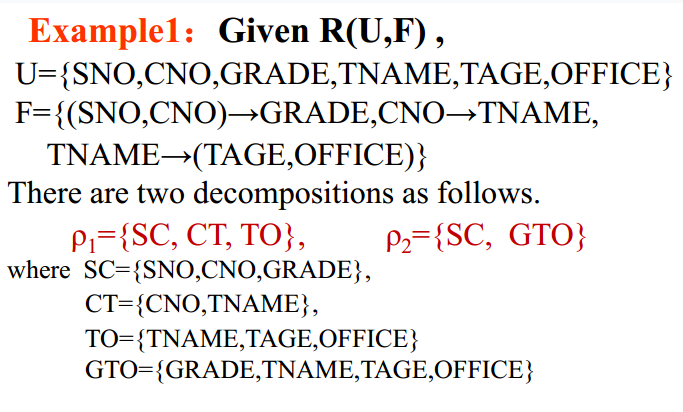
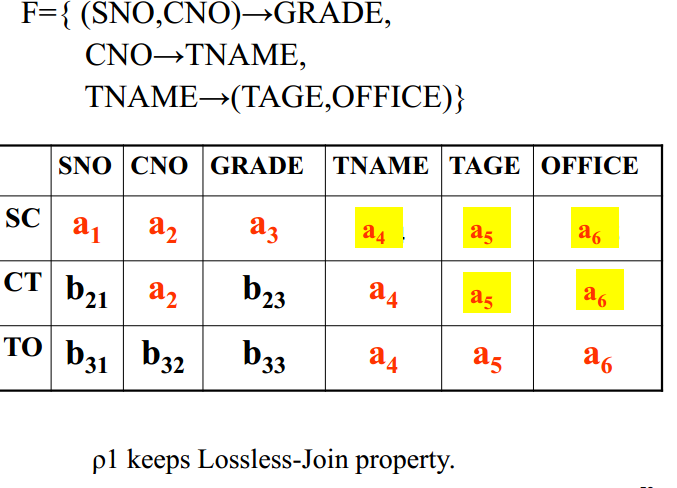
数据库表或自由表都可排序和索引，排序是从表记录中依某字段或字段表达式的值，按升序或降序重新组织排列记录先后顺序、并生成排序成功的新表文件的操作，排序操作后只有打开了排序成功后的新表文件，排序才有效；

索引是依表中某字段或字段表达式值，进行表中记录先后顺序重排的操作（升、降序），不过索引不生成新表文件，只产生索引文件，且不破坏原来表中的记录号次序，即保持记录号不变，只是调整了记录次序，索引文件因为不存储数据，只保存记录指针，所以索引文件比较小。注意：索引并未改变表中记录的物理位置，仅仅改变了表中记录的逻辑排序。但是，当用户将建立好的索引文件打开以后，记录的显示顺序或读取处理记录的顺序将会按照索引文件排列的记录顺序进行。这样大大提高了记录的检索速度。可以为一个表同时建立多个索引文件，每个索引文件表示处理记录的不同顺序。

无损连接分解：拆分后合并观察是否有问题 对于函数依赖，拆分后函数依赖仍要保持

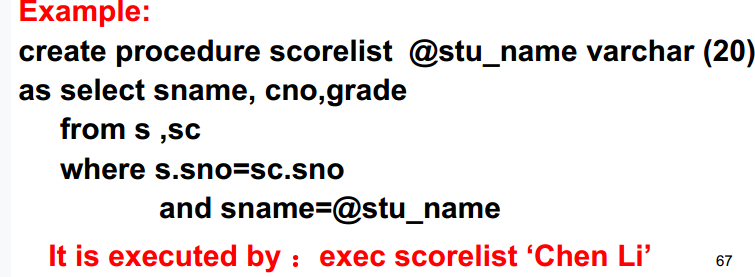
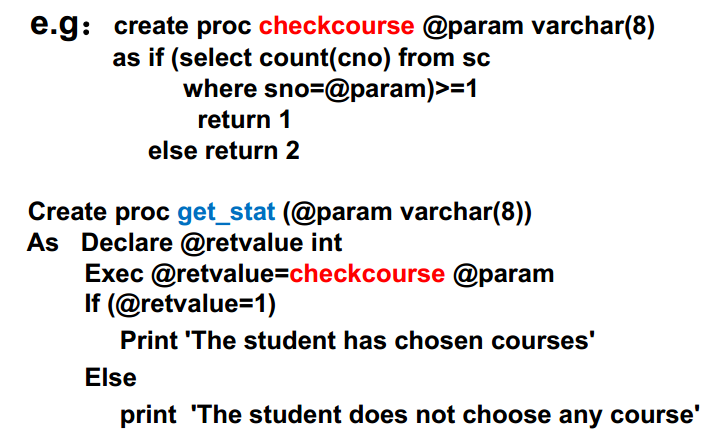


 两者之一必须在函数依赖中，否则不是无损连接



如果一个R的候选码是单一属性，它至少是二级范式。关系R中都是主属性，最少是三级范式。

复合候选码有公共属性，可能不是BCNF，无重叠的复合候选码，一定是BCNF。



触发器是一种特殊类型的存储过程，其中不允许使用参数。

触发器不能由用户直接调用，只能在给定数据项被修改时由系统自动触发并执行。

触发器用于保证数据的完整性，即PK和FK的匹配，也用于实现复杂的业务规则。

SQL SERVER中的触发器不支持在临时表上创建触发器，但临时表可以用于触发器。

在SQL SERVER中有两个特殊的临时表，即插入表和删除表，它们与触发表或视图具有相同的结构。

用户可以参考上述两个临时表，但不能修改其中的数据。

AFTER(或For)只能在表上创建。(1)触发器仅在某些事件发生时被唤醒，例如插入、更新或删除。

(2)触发器测试一个条件，即对约束进行检查。

如果这个条件不成立，那么在响应这个事件时，没有其他与触发器相关的事情发生。

如果满足条件，则由DBMS执行AFTER触发器的动作

INSTEAD OF类型 (既可建于基本表上，也可建于视图上)

 若视图在创建时使用了“with check option”选项, 则不能在其上创建触发器。

 当在一个视图上定义了一个触发器，可以用Instead of代替AFTER触发器。

 若如此，当一个事件唤醒触发器时，则执行该触发器的动作，而不执行该事件本身

针对一个基本表或视图上的 insert, update 或 delete 的每个事件，只能定义一个INSTEAD OF 类型的触发器.

针对一个基本表上的 insert, update 或 delete 的每个事件，可以定义多个AFTER 类型的触发器.

AFTER 触发器的执行过程在执行触发器动作之前, 先检查完整性约束；因此，若 insert, update 或 delete 操作违反了数据完整性，则 AFTER 触发器的动作不会被执行。

