1. **索引对关系型数据库系统查询优化的意义？应该在什么时候使用索引？是不是在任何时候使用索引都能获得益处？举例说明。**

关系型数据库查询优化的途径之一是依赖于存取路径的优化，而在关系型数据库中索引是用得最多的一种存取路径，建立合适的索引是实现查询优化的首要前提。索引提供了对数据的快速访问，根据操作建立合适的索引能够很大程度上优化存取路径，从而提高查询效率。

一般在数据量较大的时候适合使用索引，在需要频繁进行查询、排序或分组操作的属性性建立合适的索引。并非在任何情况下使用索引都能优化查询效率，应该根据实际操作需求考虑。例如对于小文件上的顺序查找使用堆文件的形式的开销不大，使用索引反而可能会增加开销。因为索引本身需要占用一定的存储空间，而且维护索引也需要一定的开销。对于增删改操作频繁的属性上建立索引可能会起到反作用。

. 索引并不是越多越好，索引固然可以提高相应的 select 的效率，但同时也降低了 insert 及 update 的效率，因为 insert 或 update 时有可能会重建索引，所以怎样建索引需要慎重考虑，视具体情况而定。一个表的索引数最好不要超过 6 个，若太多则应考虑一些不常使用到的列上建的索引是否有必要。

并不是所有索引对查询都有效，SQL 是根据表中数据来进行查询优化的，当索引列有大量数据重复时，SQL 查询可能不会去利用索引，如一表中有字段 sex，male、female 几乎各一半，那么即使在 sex 上建了索引也对查询效率起不了作用。

2. **判断并发事务运行正确性的标准是什么？封锁法的基本思想？它是怎么保证并发事务正确执行的？采用封锁法以后必须解决的问题是什么？**

判断并发事务运行正确性的标准是对并发事务运行的调度是否可串行化。

封锁法的基本思想并发事务对同一数据对象操作前，向系统发出请求对操作对象进行加锁。

事务对数据对象的加锁请求获准后，它便对该对象有了一定的控制，在这个事务释放它的锁之前，其他事务对该数据对象的锁请求不能获准，无法对其进行操作，从而避免了访问冲突，保证并发事务正确执行。

采用封锁法后可能会出现活锁、死锁等问题，其中必须解决的问题是由于事物之间的循环等待导致的死锁问题。

3. **嵌入式 SQL 打印报表，内容是水手级别和该级别水手人数。**

```
    …
// 声明 SQL 通信区
EXEC SQL INCLUDE SQLCA;
// 定义宿主变量
EXEC SQL BEGIN DECLARE SECTION;
    float RATING;
    int NUM;
EXEC SQL END DECLARE SECTION;
// 连接数据库
EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;
// 使用游标读取数据
EXEC SQL DECLARE C1 CURSOR FOR
    SELECT rating, COUNT(sid)
    FROM Sailors
    GROUP BY rating
EXEC SQL OPEN C1;
while (TRUE)
{
    EXEC SQL FETCH C1 INTO :RATING, :NUM;
    if (SQLCA.SQLCODE == 10)
        break;
    if (SQLCA.SQLCODE < 0)
        break;
    （循环打印数据）
    ……
}
EXEC SQL CLOSE C1;
```

4. 现代数据库系统如何实现数据的独立性？数据模式遵循的范式越高越好吗？

（1） 数据的独立性是指数据库中数据和程序的独立性，它包括物理数据独立性和逻辑数据独立性。物理数据独立性指用户的应用程序与存储在磁盘上的数据库中的数据是相互独立的；逻辑数据独立性指用户的应用程序与数据库的逻辑结构是相互独立的。数据独立性的重要性：如果数据的独立性好，那么当数据的存储结构或数据的逻辑发生变化时，不会影响到应用程序。

数据的独立性是数据库系统的最基本的特征之一 。数据独立性是指应用程序和数据结构之间相互独立，互不影响。在三层模式体系结构中数据独立性是指数据库系统在某一层次模式上的改变不会使它的上一层模式也发生改变的能力。正是三级模式间的两层映像保证了数据库系统中的数据具有较高的数据独立性。数据独立性包括数据逻辑独立性和数据物理独立性。

（2） 数据库的范式主要目的是防止数据冗余、更新异常、插入异常和删除异常，因此，如果达到了该目的也就可以了，但范式越高可能带来处理速度缓慢和处理逻辑复杂的问题，因此需要权衡考虑。并不是应用的范式越高越好，要看实际情况而定。应用的范式等级越高，则表越多。表多会带来很多问题：1 查询时要连接多个表，增加了查询的复杂度 2 查询时需要连接多个表，降低了数据库查询性能

5. 从查询优化角度分析，为什么 SQL 查询 where 子句应尽量避免使用 OR？

对查询进行优化，应尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。 应尽量避免在 where 子句中使用 or 来连接条件，否则将导致引擎放弃使用索引而进行全表扫描

6. 稠密索引是否一定能提高针对索引属性查询的效率？

稠密索引不一定能提高针对索引属性的查询效率。

② 如果是查询小文件中的全部或相当多的记录时，使用索引并不能提高查询效率，反而会因为索引增加开销；

② 如果稠密索引为次索引，但不是簇集索引，也就是说一个键值对应的多条记录分散在不同的物理块中；当一个键值对应的记录较多时，取这些记录时访问物理课的 I/O 开反而会降低查询的效率。

7. 相较层次和网状数据库系统，查询优化对关系数据库系统更为重要？这句话对吗？

我认为这句话是对的。层次和网状数据库的数据模型使用指针表示属性之间的关系，这样的结构也固定了这两种数据库的查询路径，进行优化的空间有限。

关系数据理论和关系数据查询语言提供了查询优化的空间。关系型数据库的抽象程度深，其查询语言一般是非过程语言，仅表达查询要求，不说明查询的过程。对于同一个查询语句，对应的关系代数等价的不同表达式的查询效率有着很大的差异；集合操作不同的执行规则和策略也对查询效率有着很大的影响；同时关系数据在物理存储形式和存取方式和路径上都没有限制。因此对于关系数据库系统来说，查询优化就更为重要，它对系统的性能有着很大的影响。

8. 对于课程/学生和选课表的数据库，（1）关系模式的设计考虑了哪些问题？（2）表达每门课的先修情况，如何调整数据库的设计？设计方案和理由。

(1)考虑了完整性约束和规范化的问题。

① 数据库中各个关系表都有主键，主键的值是唯一的且不为空，满足了实体完整性约束。同时 enroll 表中定义了对 course 和 student 表的外键，满足了引用完整性约束。

② 关系中的属性都是原子的，因此满足第一范式的要求；同时各关系表中不存在部分函数依赖和传递函数依赖，因此该数据库的设计也满足第二和第三范式的要求。

(2)先修课程包括课程号、课程名、开课院系等属性。如果将先修课程情况直接作为属性附加到 course 关系表中，会导致关系中出现部分函数依赖，不符合二范式的要求。而且先修课程可能不止一门，上述的做法还会产生大量冗余。因此需要新增一个关系表表达先修课程情况。关系表设计如下：

cno	dname	cno1	dname1

其中 cno 和 dname 表示当前课程的课程号和开课院系，cno1 和 dname1 表示先修课程的课程号和开课院系。cno 和 dname、cno1 和 dname1 都是对 course 表的外键。