1. 请介绍一下索引及其原理？

数据库有三种常用的文件类型，分别是堆文件、直接文件和索引文件。其中，堆文件是最简单、最原始的文件结构，一般用顺序搜索或者顺序扫描的方式获取数据，I/O和CPU代价很大，适用于较小的文件。第二种直接文件，主要是将某一属性用散列函数直接映射成记录的地址，这种方式查询速度很快，但是因为其地址空间固定，会导致溢出，不便处理变长记录，导致应用不是很广泛，目前有各种改进优化的措施。最后一种索引文件，是非常重要的一种文件存储方式。在主键上建立的主索引可以用来做唯一性检查，在非主键上建立次索引都是稠密索引，分为簇集索引和非簇集索引。索引能够优化I/O，但是在更新时，增加了维护的开销，另外也带来了一定的空间开销。

动态索引主要利用B树和B+树来实现索引。B树中每个节点包含了键值和键值对应的地址。查找过程：从根节点出发，在根节点所包含的的关键字中查找，若找到则查找成功；否则，一定可以确定要查的关键字在某两个关键字之间，根据对应的指针到下一层节点中寻找。所以，成功搜索一个对象不用到达树的叶节点。B+树中所有关键字都存放在叶子节点中，一般都是基于双向链表的顺序集。非叶结点包含了子树中的最大值或者最小值。B+树可提供三种存取路径，通过索引级集进行树形搜索；通过顺序集进行顺序搜索；先通过索引找到入口，再沿顺序集顺序搜索（范围查找）。B树节省存储空间，但是插入、删除负责度高；B+树查询效率相对较低，是B树的最坏情况，但是更新复杂度低了一些。

动态散列主要用来建立散列文件。线性散列的基本思想是让桶的数目随着溢出的增加而增加，随着桶中负荷的减少而降低。虽然查询效率更高，但仍然存在一系列的问题戴解决。

联合索引：等值查询中，查询条件a返回的条目比较多，查询条件b返回的条目比较多，而同时查询a、b返回的条目比较少，那么适合建立联合索引；对于有等值查询的列和范围查询的列，等值查询的列建在前、范围查询的列建在后比较实用；如果联合索引列的前置列与索引单列一致，那么单列查询可以用到索引，这样就避免了再建单列索引，因此联合索引的前置列应尽量与单列一致。

超过3个列的联合索引不合适，否则虽然减少了回表动作，但索引块过多，查询时就要遍历更多的索引块了；对于某段时间内，海量数据表有频繁的更新，这时可以先删除索引，插入数据，再重新建立索引来达到高效的目的。

对于单列索引来说，比较适合建在重读度低的列上。

1. 介绍一下查询处理及优化的主要内容？

查询优化有不同的途径，一种是对查询语句进行变换，如改变基本操作的次序，使查询语句执行起来更加有效。另外一种途径是根据系统提供的存取路径，选取合理的存取策略，如选用顺序搜索或索引进行查询。具体的，优化措施包括代数优化、基于存储路径的规则优化以及代价估算。

代数优化是对查询进行等效变换，以减少执行的开销。最常用的变换原则是，尽量缩减查询的中间结果。尽量先做选择、投影等一元操作，再做连接等二元操作。在连接时，先做小关系之间的连接，再做大关系的连接，通常都会通过查询树来解决问题。

查询的启发式规则：对于小关系，不必考虑其他存取路径，直接用顺序扫描；如果无索引或者散列，或者估计选中的元组数在关系中占有较大的比例，直接用顺序扫描；对于主键的等值查询，优先采用主键上的索引或者散列；对于非主键的等值条件查询，如果选中的数据量较小，则可用无序索引，否则只能用簇集索引或者顺序扫描；对于范围查询，一般先通过索引找到范围的边界，然后通过索引的顺序集沿相应的方向扫描；对于and连接的合取选择条件，若有相应的多属性索引，则优先考虑，否则采用二次索引，先将地址取出，再做合取操作，再进行IO操作。

连接操作的实现方法R\*S：嵌套循环法—取出外层表的一个元组，与内层表进行比较，凡满足连接条件的元组就进行连接，并作为结果输出，应将物理块少的关系作为外关系。利用索引或者散列，内关系选择合适的存取路径；排序归并发，按连接属性排序，则都只要进行一次扫描；散列连接，用相同的散列函数将两表散列到同一文件，符合连接条件的元组必然在同一桶中，只要把桶中符合条件的元组取出，即可得到结果。

代价估算主要用于编译实现的数据库系统中，一次编译可供多次执行，一般先用规则优化，选择几个可取的执行策略，然后对他们进行代价比较，从中择优

1. 数据库视图的理解。

视图（View）是从一个或多个表（或视图）导出的表。视图与表（有时为与视图区别，也称表为基本表——Base Table）不同，视图是一个虚表，即视图所对应的数据不进行实际存储，数据库中只存储视图的定义，在对视图的数据进行操作时，系统根据视图的定义去操作与视图相关联的基本表。

视图能简化用户操作：使用户可以将注意力集中在所关心地数据上。

视图能够对机密数据提供安全保护：对不同的用户定义不同的视图，使机密数据不出现在不应该看到这些数据的用户视图上。

1. 事务管理

原子性（要么全做要么不做）、一致性（完整性约束）、隔离性（并发事务）、持久性（持久影响）。

恢复：后备副本、后备副本和运行记录、多副本。

运行记录：活动事务表（ATL）、提交事务表（CTL）、前像文件（undo）、后像文件（redo）。

事务的执行过程：一般会先把TID（事务的id号）加入到活动事务表中，然后把前像记录到log中，然后把后像提交到log和数据库中，完成后把TID加入到提交事务表中，从活动事务表中删除。这个是一般方法，但效率较低。可以先先把TID（事务的id号）加入到活动事务表中，然后把后像记录到log中，然后把TID加入到提交事务表中，再把后像提交到数据库，最后删除ATL中的TID。

事务失效、系统失效、介质失效。

并发问题：丢失更新、读脏数据、读值不可复现。

X锁、（S，X）锁。

1. Join操作

内连接inner join：交集

外连接：left outer join（左表的完全集，没有匹配的用null值替代）、right outer join（右表的全集）和full outer join（并集）。

SELECT \* FROM A LEFT JOIN B ON A.ID = B.ID AND A<>0 WHERE A.name = 'x'

FROM --> JOIN --> ON -->AND-->　LEFT--> WHERE -->SELECT

先将两张表根据ON 条件 作连接（逻辑上，相等于将两张表笛卡尔集后根据ID相等条件筛选数据，实际情况后面分析）

根据ON 后面,WHERE 之前 的 AND 条件筛选数据

跟据LEFT 无论如何，要保证A表的数据完整性。所以在上一步骤产生的结果集中补齐A表因无法比与B表匹配而被AND 条件筛选的掉的数据；

再根据WHERE筛选结果集。

在使用Left (right) join的时候，一定要在先给出尽可能多的匹配满足条件，减少Where的执行。

尽量避免子查询，而用join.

1. 手写SQL语句。

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

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.S#=b.S# and a.Score>b.Score

1. 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) as AvgScore

from SC

group by S#

having AVG(Score)>60

1. 查询所有同学的学号、姓名、选课数、总成绩；

select s.S#, s.Sname, COUNT(sc.C#) as CourseCount, SUM(sc.Score) as ScoreSum

from Student s left outer join SC sc

on s.S# = sc.S#

group by s.S#,s.Sname

order by s.S#

1. 查询姓“李”的老师的个数；

select COUNT(distinct Tname) as count

from Teacher

where Tname like '李%'

1. 查询没学过“叶平”老师课的同学的学号、姓名；

select s.S#,s.Sname

from Student s

where s.S# not in

(

select distinct(sc.S#) from SC sc, Course c, Teacher t

where sc.C#=c.C# and c.T#=t.T# and t.Tname='叶平'

)

1. 查询学过“001”并且也学过编号“002”课程的同学的学号、姓名；

select s.S#,s.Sname

from Student s,SC sc

where s.S#=sc.S# and sc.C#='001'

intersect

select s.S#,s.Sname

from Student s,SC sc

where s.S#=sc.S# and sc.C#='002'

select s.S#,s.Sname

from Student s,SC sc

where s.S#=sc.S# and sc.C#='001' and exists

(

select \* from SC sc2 where sc.S#=sc2.S# and sc2.C#='002'

)

in 是把外表和内表作hash 连接，而exists是对外表作loop循环，每次loop循环再对内表进行查询。

如果查询的两个表大小相当，那么用in和exists差别不大。

如果两个表中一个较小，一个是大表，则子查询表大的用exists，子查询表小的用in。

1. 查询学过“叶平”老师所教的所有课的同学的学号、姓名；

select s.S#,s.Sname

from Student s

where s.S# in

(

select sc.S#

from SC sc,Course c,Teacher t

where c.C#=sc.C# and c.T#=t.T# and t.Tname='叶平'

group by sc.S#

having COUNT(sc.C#)=

(

select COUNT(c1.C#)

from Course c1,Teacher t1

where c1.T#=t1.T# and t1.Tname='叶平'

)

)

1. 查询课程编号“002”的成绩比课程编号“001”课程低的所有同学的学号、姓名；

select s.S#,s.Sname

from Student s,

(select sc1.S#,sc1.Score from SC sc1 where sc1.C#='002') a,

(select sc2.S#,sc2.Score from SC sc2 where sc2.C#='001') b

where s.S#=a.S# and s.S#=b.S# and a.S#=b.S# and a.Score<b.Score

1. 查询有课程成绩小于60分的同学的学号、姓名；

select s.S#,s.Sname

from Student s

where s.S# in

(

select distinct(sc.S#) from SC sc

where s.S#=sc.S# and sc.Score<60

)