1.向表中添加列

ALTER TABLE TableName ADD ColumnName Type;

2.删除列

ALTER TABLE TableName DROP COLUMN ColumnName;

3.modify,alter,update的区别：

modify：修改字段类型和长度（修改字段的属性）

Alter：修改表的数据结构

Update：修改数据内容的

4.tablespace和datafile之间的关系

一个tablespace可以有一个或多个datafile;

每一个datafile只能在一个tablespace内;

table中的数据，通过hash算法分布在tablespace中的各个datafile内;

5.DELETE和TRUNCATE TABLE的区别

1.两者都是删除表中的数据；

2.TRUNCATE TABLE A比DELETE快；

3.TRUNCATE TABLE是删除表中的所有行，而DELETE是删除表中的某一行，或者多行（除非DELETE不带WHERE语句）；

3.在删除时如果遇到任何一行违反约束，TRUNCATE TABLE仍然删除，只是表的结构及其列、约束、索引等保持不变，而DELETE直接返回错误；

4.对于被外键约束的表，不能使用TRUNCATE TABLE，而应该使用不带WHERE语句的DELETE;

5.如果想要保留标识计数值，要使用DELETE,因为TRUNCATE TABLE会对新行标志符列搜用的计数值重置为该列的种子。

6.MySql中查看SQL模式的命令

MySql中变量分为系统变量（以@@开头），和用户自定义变量，系统变量分为全局系统变量（global）和会话系统变量（session）；

@@global：仅用于访问全局系统变量的值；

@@sesson：仅用于访问会话系统变量的值；

@@：先访问会话系统变量的值，若不存在，则去访问全局系统变量的值；

sql_mode为系统变量，即是全局系统变量，又是会话系统变量。

7.MySql中的触发器

1.MySQL的触发器只支持行级出发，不支持语句级触发，触发程序与表相关，当对表执行INSERT、DELETE或UPDATE语句时，将激活触发程序。可以将触发程序设置为在执行语句之前或之后激活。例如，可以在从表中删除每一行之前，或在更新了要想创建触发程序或舍弃触发程序，可使用CREATE TRIGGER或DROP TRIGGER语句

2.触发程序不能调用将数据返回客户端的存储程序，也不能使用采用CALL语句的动态SQL（允许存储程序通过参数将数据返回触发程序）。

3.在MySQL中，使用new和old引用触发器中发生的记录内容，使用OLD和NEW关键字，能够访问受触发程序影响的行中的列（OLD和NEW不区分大小写）。在INSERT触发程序中，仅能使用NEW.col_name，没有旧行。在DELETE触发程序中，仅能使用OLD.col_name，没有新行。在UPDATE触发程序中，可以使用OLD.col_name来引用更新前的某一行的列，也能使用NEW.col_name来引用更新后的行中的列。

4.触发程序不能使用以显式或隐式方式开始或结束事务的语句，如START TRANSACTION、COMMIT或ROLLBACK。

8.事务隔离级别

未提交读(Read Uncommitted)：允许脏读，也就是可能读取到其他会话中未提交事务修改的数据

提交读(Read Committed)：只能读取到已经提交的数据

可重复读(Repeated Read)：在同一个事务内的查询都是事务开始时刻一致的

串行读(Serializable)：完全串行化的读，每次读都需要获得表级共享锁，读写相互都会阻塞

9.在数据库系统中，产生不一致的根本原因

- 数据冗余

如果数据库中存在冗余数据，比如两张表中都存储了用户的地址，在用户的地址发生改变时，如果只更新了一张表中的数据，那么这两张表中就有了不一致的数据。

- 并发控制不当

比如某个订票系统中，两个用户在同一时间订同一张票，如果并发控制不当，可能会导致一张票被两个用户预订的情况。当然这也与元数据的设计有关。

- 故障和错误

如果软硬件发生故障造成数据丢失等情况，也可能引起数据不一致的情况。因此我们需要提供数据库维护和数据恢复的一些措施。

10.数据库运行于Archivelog状态下可以防止数据的丢失

- 在archivelog mode只要其归档日志文件不丢失，就可以有效地防止数据丢失

11.聚集索引

聚集索引是一种索引，该索引中键值的逻辑顺序决定了表中相应行的物理顺序。

聚集索引确定表中数据的物理顺序。聚集索引类似于电话簿，按姓氏排列数据。由于聚集索引规定数据在表中的物理存储顺序，因此一个表只能包含一个聚集索引。但该索引可以包含多个列（组合索引），就像电话簿按姓氏和名字进行组织一样。

聚集索引对于那些经常要搜索范围值的列特别有效。使用聚集索引找到包含第一个值的行后，便可以确保包含后续索引值的行在物理相邻。例如，如果应用程序执行的一个查询经常检索某一日期范围内的记录，则使用聚集索引可以迅速找到包含开始日期的行，然后检索表中所有相邻的行，直到到达结束日期。这样有助于提高此类查询的性能。同样，如果对从表中检索的数据进行排序时经常要用到某一列，则可以将该表在该列上聚集（物理排序），避免每次查询该列时都进行排序，从而节省成本。

当索引值唯一时，使用聚集索引查找特定的行也很有效率。例如，使用唯一雇员 ID 列 emp_id 查找特定雇员的最快速的方法，是在 emp_id 列上创建聚集索引或 PRIMARY KEY 约束。

12.数据库语言分类

- DDL：数据库模式定义语句，关键字：create
- DML：数据库操纵语句，关键字：INSERT，DELETE，UPDATE
- DCL：数据库控制语句，关键字：grant、remove
- DQL：数据库查询语句，关键字：SELECT

13.数据库事务隔离的四个级别

1. 脏读 ：一个事务读到另一个事务未提交的更新数据。

脏读就是指当一个事务正在访问数据，并且对数据进行了修改，而这种修改还没有提交到数据库中，这时，另外一个事务也访问这个数据，然后使用了这个被更新后的数据。

2. 幻读 : 一个事务读到另一个事务已提交的新插入的数据。

   例如第一个事务对一个表中的数据进行了修改，这种修改涉及到表中的全部数据行。同时第二个事务向表中插入一行新数据。那么第一个事务发现表中还有没有修改的数据行，就好象发生了幻觉一样。

3. 不可重复读 ：一个事务读到另一个事务已提交的更新数据。

   指一个事务两次对同一行数据查询，由于第二个事务在此期间对此行数据进行了修改导致第一个事务两次读取到的同一行数据不同称之为不可重复读。

   数据库事务的隔离级别有4个，由低到高依次为Read uncommitted、Read committed、Repeatable read、Serializable，这四个级别可以逐个解决脏读、不可重复读、幻读这几类问题。

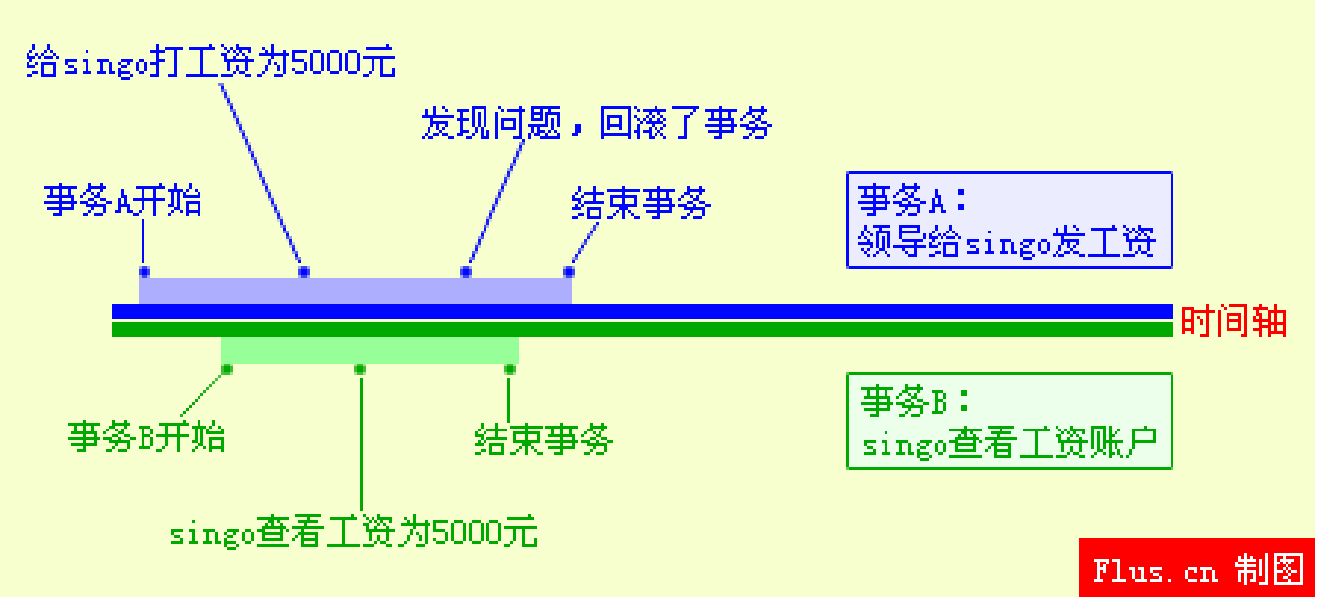
√: 可能出现　×: 不会出现

	脏读	不可重复读	幻读
Read uncommitted	√	√	√
Read committed	×	√	√
Repeatable read	×	×	√
Serializable	×	×	×

注意：我们讨论隔离级别的场景，主要是在多个事务并发的情况下，因此，接下来的讲解都围绕事务并发。

Read uncommitted 读未提交

公司发工资了，领导把5000元打到singo的账号上，但是该事务并未提交，而singo正好去查看账户，发现工资已经到账，是5000元整，非常高兴。可是不幸的是，领导发现发给singo的工资金额不对，是2000元，于是迅速回滚了事务，修改金额后，将事务提交，最后singo实际的工资只有2000元，singo空欢喜一场。



出现上述情况，即我们所说的脏读，两个并发的事务，“事务A：领导给singo发工资”、“事务B：singo查询工资账户”，事务B读取了事务A尚未提交的数据。

当隔离级别设置为Read uncommitted时，就可能出现脏读，如何避免脏读，请看下一个隔离级别。

Read committed 读提交

singo拿着工资卡去消费，系统读取到卡里确实有2000元，而此时她的老婆也正好在网上转账，把singo工资卡的2000元转到另一账户，并在singo之前提交了事务，当singo扣款时，系统检查到singo的工资卡已经没有钱，扣款失败，singo十分纳闷，明明卡里有钱，为何.....

出现上述情况，即我们所说的不可重复读，两个并发的事务，“事务A：singo消费”、“事务B：singo的老婆网上转账”，事务A事先读取了数据，事务B紧接了更新了数据，并提交了事务，而事务A再次读取该数据时，数据已经发生了改变。

当隔离级别设置为Read committed时，避免了脏读，但是可能会造成不可重复读。

大多数数据库的默认级别就是Read committed，比如Sql Server , Oracle。如何解决不可重复读这一问题，请看下一个隔离级别。

Repeatable read 重复读

当隔离级别设置为Repeatable read时，可以避免不可重复读。当singo拿着工资卡去消费时，一旦系统开始读取工资卡信息（即事务开始），singo的老婆就不可能对该记录进行修改，也就是singo的老婆不能在此时转账。

虽然Repeatable read避免了不可重复读，但还有可能出现幻读。

singo的老婆工作在银行部门，她时常通过银行内部系统查看singo的信用卡消费记录。有一天，她正在查询到singo当月信用卡的总消费金额（select sum(amount) from transaction where month = 本月）为80元，而singo此时正好在外面胡吃海塞后在收银台买单，消费1000元，即新增了一条1000元的消费记录（insert transaction ... ），并提交了事务，随后singo的老婆将singo当月信用卡消费的明细打印到A4纸上，却发现消费总额为1080元，singo的老婆很诧异，以为出现了幻觉，幻读就这样产生了。

注：MySQL的默认隔离级别就是Repeatable read。

Serializable 序列化

Serializable是最高的事务隔离级别，同时代价也花费最高，性能很低，一般很少使用，在该级别下，事务顺序执行，不仅可以避免脏读、不可重复读，还避免了幻像读。

14.用来诊断oracle IO、CPU、性能状况

V$SQLAREA(DISK_READS) 本视图持续跟踪所有shared pool中的共享cursor，在shared pool中的每一条SQL语句都对应一列。本视图在分析SQL语句资源使用方面非常重要。

STATESPACK 通过Statspack我们可以很容易的确定Oracle数据库的瓶颈所有，记录数据库性能状态，也可以使远程技术人员迅速了解的的数据库运行状况。

SQL_TRACE sql_trace是oracle提供的一个非常好的跟踪工具，主要用来检查数据库的异常情况，通过跟踪数据库的活动，找到有问题的语句。

V$SESSION_WAIT 它提供了任何情况下session在数据库中当前正在等待什么(如果session当前什么也没在做，则显示它最后的等待事件)。当系统存在性能问题时，本视图可以做为一个起点指明探寻问题的方向。

15.索引

一、什么是索引：

简单的来说，建立索引在进行数据库操作的时候不需要全盘一条条的扫描，删选出符合的记录，索引内部自己有一套优化算法，因此借助索引来对数据库进行操作可以提高查询的效率。

二、什么时候建立的索引将失效或效率不高（情况有很多，这里列举常见的几种， 假设在字段name上建立了索引）：

1、使用了运算符!=，以及关键字not in, not exist等，认为产生的结果集很大，往往导致引擎不走索引而是走全盘扫描

2、对索引字段使用了函数，如where substr(name, 1, 3)='mark'， 导致索引无效

3、使用**like**和通配符，第一个字符是**%**将导致索引失效，如**where name like "%ark"**

三、order by与索引

首先利用where进行数据查询，这一步是免不了的，至于这一步有没有利用索引暂时不考虑，关键是在获取所有符合的记录后还需要进行排序，看看order by是如何利用索引的。

如果order by中的字段有建立索引，同时：

1、该字段没有出现在**where**中，则在排序的时候需要正常排序，默认**order by**是升序排序**,** 故索引没有对排序产生有利帮助 （**B,C**错误）

2、该字段同时同时出现在**where**中，则在获取记录后不进行排序，而是直接利用索引， 效率变高。

补充： group by也和order by类似

四、哪些字段适合建立索引

1、表的主键、外键必须有索引；
2、数据量超过300的表应该有索引；
3、经常与其他表进行连接的表，在连接字段上应该建立索引；
4、经常出现在Where子句中的字段，特别是大表的字段，应该建立索引；
5、索引应该建在选择性高的字段上；
6、索引应该建在小字段上，对于大的文本字段甚至超长字段，不要建索引；
7、复合索引的建立需要进行仔细分析；尽量考虑用单字段索引代替：
A、正确选择复合索引中的主列字段，一般是选择性较好的字段；
B、复合索引的几个字段是否经常同时以AND方式出现在Where子句中？单字段查询是否极少甚至没有？如果是，则可以建立复合索引；否则考虑单字段索引；
C、如果复合索引中包含的字段经常单独出现在Where子句中，则分解为多个单字段索引；
D、如果复合索引所包含的字段超过3个，那么仔细考虑其必要性，考虑减少复合的字段；
E、如果既有单字段索引，又有这几个字段上的复合索引，一般可以删除复合索引；
8、频繁进行数据操作的表，不要建立太多的索引；
9、删除无用的索引，避免对执行计划造成负面影响；

复合索引: Mysql从左到右的使用索引中的字段，一个查询可以只使用索引中的一部份，但只能是最左侧部分。 例如索引是key index (a,b,c). 可以支持a | a,b| a,b,c 3种组合进行查找，但不支持 b,c进行查找 .当最左侧字段是常量引用时，索引就十分有效

一个索引可以对应多个列，索引也可以升序、降序

CREATE UNIQUE INDEX SC_INDEX ON SC(S_ID ASC, C_ID DESC);

SQL 练习

教学数据库的四个关系

教师关系 T(T_ID, TNAME, TITLE)

课程关系 C(C_ID, CNAME, T_ID)

学生关系 S(S_ID, SNAME, AGE, SEX)

选课关系 SC(S_ID, C_ID, SCORE)

1. 至少选修LIU老师所授课程中的一门课程的学生学号与姓名

```
SELECT S.S_ID, S.SNAME
FROM S, SC, C, T
WHERE S.S_ID = SC.S_ID AND SC.C_ID = C.C_ID AND C.T_ID = T.T_ID
    AND TNAME = 'LIU'
```

2. 至少选修课程号为C2和C4的学生学号

```
SELECT X.S_ID
FROM SC AS X, SC AS Y
WHERE X.S_ID = Y.S_ID AND X.C_ID = 'C2'
    AND Y.C_ID = 'C4'
```

3. 检索不学C2课程的学生姓名和年龄

```
SELECT SNAME, AGE
FROM S
WHERE S_ID NOT IN(
  SELECT S_ID
  FROM SC
  WHERE C_ID = 'C2'
)
```

```
SELECT SNAME, AGE
FROM S
WHERE NOT EXISTS(
  SELECT *
  FROM SC
  WHERE SC.S_ID = S.S_ID
    AND C_ID = 'C2'
)
```

4. 检索学习了全部课程的学生姓名

   解析：在S表中查找学生，要求这个学生学了全部课程，换言之，在表S中找学生，在C表中不存在一门课程，这个学生没有学

```
SELECT SNAME
FROM S
WHERE NOT EXISTS(
    SELECT *
    FROM C
    WHERE NOT EXISTS(
        SELECT *
        FROM SC
        WHERE SC.S_ID = S.S_ID AND SC.C_ID = C.C_ID
    )
)
```

5. 检索所学课程包含学生S3所学所有课程的学生学号

```
SELECT DISTINCT S_ID
FROM SC AS X
WHERE NOT EXISTS(
    SELECT *
    FROM SC AS Y
    WHERE Y.S_ID = 'S3'
        AND NOT EXISTS(
            SELECT *
            FROM SC AS Z
            WHERE Z.S_ID = X.S_ID AND Z.C_ID = Y.C_ID
        )
)
```

6. 检索至少有一门成绩超过学生S4一门成绩的学生学号

```
SELECT DISTINCT S_ID
FROM SC
WHERE SCORE > SOME(
    SELECT SCORE
    FROM SC
    WHERE S_ID = 'S4'
)
```

7. 检索不学C2课程的学生姓名与年龄

```
SELECT SNAME,AGE
FROM S
WHERE S_ID <> ALL(
    SELECT FROM S_ID
    FROM SC
    WHERE C_ID = 'C2'
)
```

```
SELECT SNAME,AGE
FROM S
WHERE S_ID NOT IN(
    SELECT FROM S_ID
    FROM SC
    WHERE C_ID = 'C2'
)
```

8. 检索平均成绩最高的学生学号

```
SELECT S_ID
FROM SC
GROUP BY S_ID
    HAVING AVG(SCORE) >= ALL(
        SELECT AVG(SCORE)
        FROM SC
        GROUP BY S_ID
    )
```

```
SELECT S_ID
FROM SC,(
    SELECT AVG(SOCRE)
    FROM SC
    GROUP BY S_ID
) AS RESULT(AVG_SCORE)
GROUP BY SC.S_ID
    HAVING AVG(SC.SOCRE) >= ALL(RESULT.AVG_SCORE)
```

```
WITH RESULT(AVG_SCORE) AS(
     SELECT AVG(SOCRE)
     FROM SC
     GROUP BY(S_ID)
)
SELECT S_ID
FROM SC,RESULT
GROUP BY S_ID
     HAVING AVG(SCORE) >= ALL(RESULT.AVG_SOCRE)
```

9. 在基本表T和C中检索只开设了一门课程的教师工号和姓名

```
SELECT T_ID, TNAME
FROM T
WHERE UNIQUE(
     SELECT T_ID
     FROM C
     WHERE C.T_ID = T.T_ID
)
```