一、索引的优点

索引是与表相关的一个可选结构

一个表中可以存在索引，也可以不存在索引，不做硬性要求。

用以提高 SQL 语句执行的性能

快速定位我们需要查找的表的内容（物理位置），提高 sql 语句的执行性能。

减少磁盘 I/O

取数据从磁盘上取到数据缓冲区中，再交给用户。磁盘 IO 非常不利于表的查找速度（效率的提高）。

使用 CREATE INDEX 语句创建索引

在逻辑上和物理上都独立于表的数据

索引与表完全独立，表里的内容是我们真正感兴趣的内容，而索引则是做了一些编制 ，索引和数据可以存放在不同的表空间下面，可以存放在不同的磁盘下面。

Oracle 自动维护索引

当对一个建立索引的表的数据进行增删改的操作时，oracle 会自动维护索引，使得其仍然能够更好的工作。

####二、索引的类型与结构

####1、索引类型

从总的概念上来说，索引分为 B 树索引（也叫平衡树索引，即就是什么都不写，最常用）和位图索引（多用于数据仓库）。这两种索引在逻辑结构（存储）上完全不同。

####B 树索引

其中 B 树索引 又可以具体分为：

（1）唯一索引：

唯一索引确保在定义索引的列中没有重复值

Oracle 自动在表的主键列上创建唯一索引

使用 CREATE UNIQUE INDEX 语句创建唯一索引

语法：create unique index index_name on table_name (column_name);

具体列值： 索引相关列上的值必须唯一，但可以不限制 NULL 值。

（2）组合索引：

组合索引是在表的多个列上创建的索引

索引中列的顺序是任意的

如果 SQL 语句的 WHERE 子句中引用了组合索引的所有列或大多数列，则可以提高检索速度

语法：create index index_name on table_name (column_name1，column_name2);

具体列值：该表中的元组由两列共同确定一行，例如班级号 学号 唯一确定一个学生。

（3）反向键索引：

反向键索引反转索引列键值的每个字节，为了实现索引的均匀分配，避免 b 树不平衡

通常建立在值是连续增长的列上，使数据均匀地分布在整个索引上

创建索引时使用 REVERSE 关键字

语法：create index index_name on table_name (column_name) reverse;
具体列值： 适用于某列值前面相同，后几位不同的情况，例如
sno： 1001 1002 1003 1004 1005 1006 1007
索引转化： 1001 2001 3001 4001 5001 6001 7001
（4）位图索引：
位图索引适合创建在低基数列上
位图索引不直接存储 ROWID，而是存储字节位到 ROWID 的映射
节省空间占用
如果索引列被经常更新的话，不适合建立位图索引
总体来说，位图索引适合于数据仓库中，不适合 OLTP 中

语法：create bitmap index index_name on table_name (column_name);
具体列值： 不适用于经常更新的列，适用于条目多但取值类别少的列，例如性别列。
（5）基于函数的索引：
基于一个或多个列上的函数或表达式创建的索引
表达式中不能出现聚合函数
不能在 LOB 类型的列上创建
创建时必须具有 QUERY REWRITE 权限

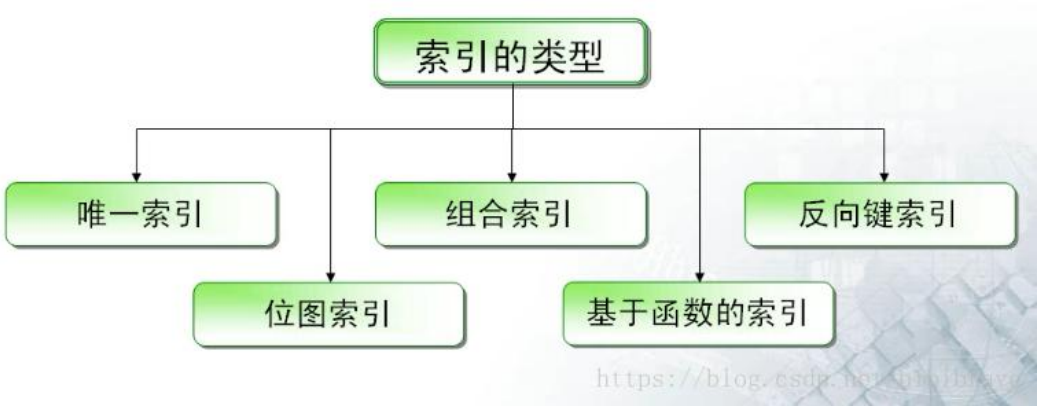
语法：create index index_name on table_name (函数（column_name））；
具体列值： 不能在 LOB 类型的列上创建，用户在该列上对该函数有经常性的要求。
例如：用户不知道存储时候姓名是大写还是小写，使用
select * from student where upper(sname)=‘TOM’；
####2、索引结构
索引的结构是一个倒立的树状结构，其中每个节点的左子树比他的右子树小，索引最终指向表里面的数据与表里面的数据对应。

如上图，前三行是索引的内部构造，第三行与最后一行，这是索引指向表里数据的一个指向。索引是建立在列上的。最后一行是索引建立在表中某列上的值。

根节点块 ：如果索引列的值>0 时，指向 B1 这个分支节点块，如果索引列的值>500 时，指向 B2 这个分支节点块，如果索引列的值>1000 时，指向 B3 这个分支节点块。
分支节点块：对于 B1 来说，再进行细分 如果索引列的值>0 且<200 时，指向 L1 这个分支节点块，如果索引列的值>200 且<400 时，指向 L2 这个分支节点块，如果索引列的值>400 且<500 时，指向 L3 这个分支节点块。
叶子节点块： 对于 L1 来说，如果数据行的值为 0，那就放在 R1 这个数据行中，如果数据行的值为 29，那就放在 R2 这个数据行中，如果数据行的值为 190，那就放在 R3 这个数据行中，等。
####三、索引的创建。
1— 建立一张表：

```
create table student(sno number ,
                     sname varchar2(10),sage int,
                     male char(3));   --一个汉字占三个字符
```

insert into student values(1,'TOM',21,'男');
insert into student values(2,'kite',22,'男');
 insert into student values(3,'john',23,'女');



2 —

创建该表上的索引：

create index ind1 on student(sno);
1
3-- 查询索引相关信息：
-- 索引的全部信息 select * from user_indexes;
--查询索引涉及到的列 select * from user_ind_columns u where u.index_name='IND1’;

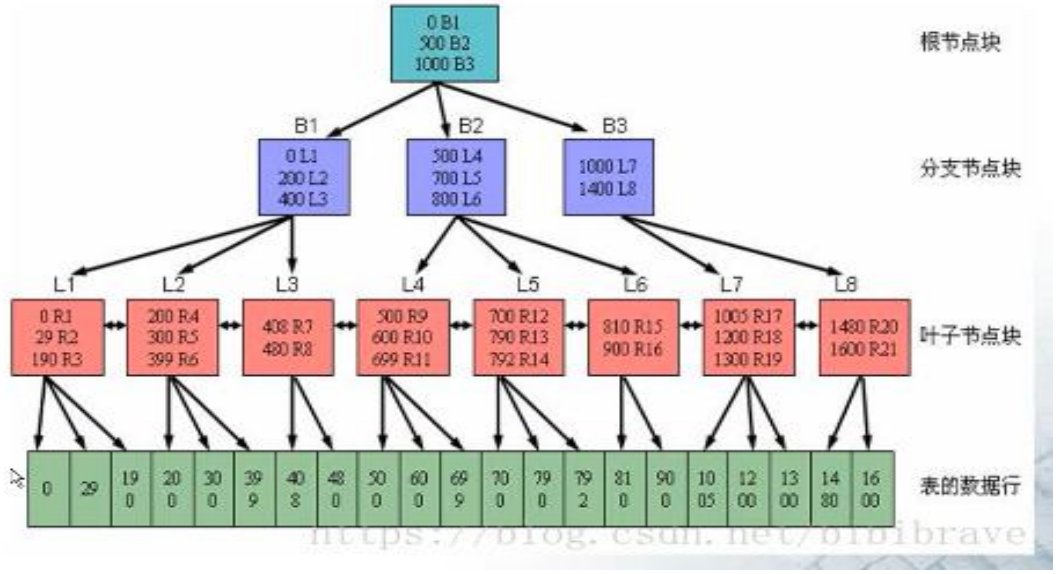
####四、索引的变动，查询索引碎片

由于我们在对表的使用过程中，必然引发增删查改等操作，当表中的数据不存在，但其索引仍然存在，极大的影响了查询速度，降低了索引的利用率。

我们可以通过 查看 index_stats 表中的 pct_used 列的值，如果 pct_used 的值过低，说明在索引中存在碎片，可以重建索引，来提高 pct_used 的值，减少索引中的碎片。

对索引碎片的查询，在该语句前后都要使用分析索引语句，已使得索引利用率发生改变。

分析：
alter index index_name validate structure;
查询碎片：
select name,pct_used from index_stats where name='index_name';



当表中数据发生变化时，我们可以通过两种方式来对索引进行更新，一是通过删除该索引，再建立新的索引来提高索引的利用率。二是通过重建索引来提高索引利用率。

①删除索引 ： drop index index_name （同时也证明表和索引之间相互独立）
create index index_name on 表名（列名） tablespace tname；

②重建索引： alter index index_name rebuild REBUILD [ONLINE] [NOLOGGING] [COMPUTE STATISTICS];
其中：

ONLINE 使得在重建索引过程中，用户可用对原来
的索引进行修改，也就是其他的用户同时可以对表进行增删改操作；
NOLOGGING 表示在重建过程中产生最少的重做条目 redo Entry，加快重建的速度；
COMPUTE STATISTICS 表示在重建过程中就生成了 oracle 优化器所需的统计信息，避免了索引重建之后再进行 analyze 或 dbms_stats 来收集统计信息。
####五、索引分区：
可以将索引存储在不同的分区中
与分区有关的索引有三种类型：

局部分区索引 — 在分区表上创建的索引，在每个表分区上创建独立的索引，索引的分区范围与表一致 （按照表分区对索引进行分区）
create index ind1 on stu（sno） local ；
全局分区索引 — 在分区表或非分区表上创建的索引，索引单独指定分区的范围，与表的分区范围或是否分区无关
*create index ind1 on stu（sno） global ； partition by range（列名） （partition 索引分区名 1 values less than(条件 1)， partition 索引分区名 2 values less than(条件 2)， partition 索引分区名 3 values less than(条件 3)， ）；
全局非分区索引 — 在分区表上创建的全局普通索引，索引没有被分区
create index ind1 on stu（sno） global ；

---- 索引的作用;
---- 索引的类型;
---- 索引的语法;
---- 创建索引的原则;

1. 不允许在 大数据(lob)列类型上创建索引
2. 不允许在 更新频繁的列上创建索引;