索引的目的是加速查询的速度：原则上一个列最多只能建两个索引
索引建完自动生效，不改变语法书写格式

索引

创建索引

```
create (unique/BITMAP) index 索引名称 on 表名(列名) tablespace 表空间名;  --unique
用于指定是否强制要求索引列为唯一性数据，表空间可选择是否指定，不指定则用默认表空间。出于性能考虑索引表空间和表的表空间要分开。
```

索引分类

一、按存储形式分类

- b_tree索引
- 位图索引
- 基于函数索引
- 方向键索引

B-TREE索引

适用场景：列基数比较大的时候使用（行业、身高）
列基数：该列不重复数据的个数 count(distinct col)

```
create index ind_name on tb_name(col_name)
```

位图索引

说明：位图索引在创建时，会扫描整张表，为索引列的每个取值建立一个不重复的位图（BITMAP）来描述该取值
适用场景：列基数比较小的时候使用（性别、婚姻状况）

```
create bitmap index ind_name on tb_name(col_name);
create bitmap index ind_student54_sex on student_54_new(sex)
```

反向键索引

说明：可以视作一种特殊的B-TREE索引，存储索引列的反向值
背景：为防止B-TREE索引在某叶上数据量占比过高而使用的一种索引
适用场景：原始数据分支不明显但反向数据分支明显的列（身高：集中在一米七一米八）

```
create index ind_name on tb_name(col_name) reverse
```

基于函数的索引

说明：可以视作一种特殊的B-TREE索引，存储函数处理后的数据
背景：在某个字段上以原值建立了索引，但是在筛选时在该字段上经常加函数，导致索引无法生效
适用场景：对某列进行筛选时经常需要配合函数使用（例如查找姓名中的首字母）

```
create index ind_name on tb_name(function(col_name));
```

二、按唯一性分类

唯一索引

索引列中不可能出现重复值

```
create unique index ind_name on tb_name(col_name)
```

- B-TREE索引可以建立唯一索引，位图索引不能建立唯一索引
- 如果在某列上建立了唯一约束或主键约束，ORACLE会自动在该列上建立一个同名的唯一索引

非唯一索引

```
create index ind_name on tb_name(col_name);
```

三、按列的个数分类（索引覆盖的列的个数）

单列索引

基于一个列建立的索引

```
create index ind_name on tb_name(col_name)
```

复合索引（联合索引）

基于两个或两个以上列建立的索引

```
--写在前面的字段（ename）且叫为主键列，触发索引需要用到主键列，与主键列的位置没有关系
create index ind_emp_enamejob on emp(ename,job)
```

修改索引名称

```
alter index 索引名 rename to 新索引名;
```

删除索引

通过drop index 索引名

索引建立或使用的规则与建议

1.如果对某大表进行筛选时，某列或某几列频繁出现在WHERE子句中，并且检索出的数 据低于总行数的15%（50%），应考虑在这些列上建立索引。
2.如果对某大表进行排序时，某列或某几列频繁出现在ORDER BY子句中，应考虑在这些 列上建立索引。
3.小表不要建立索引。
4.对于含有空值的列，如果经常在查询时查询非空值，建议在该列上建立索引；如果经 常在查询时查询空值，建议在该列上建立基于函数的索引。
5.为了提高表连接的性能，应在连接列上建立索引（建立一般普通的索引即可）
6.索引是数据库的一种实体对象，级别类似于表，会占用内存空间，ORACLE会自动进行 索引维护，表和索引可以建立在不同的表空间。
7.通过索引可以提升数据的查询速度，但是会相对地降低DML语句的操作速度，尤其是 插和改的速度，ORACLE会花费时间在索引维护上，所以说要把握好索引的数量
8.对于列基数比较大的列，适合B-TREE索引，列基数比较小的列，适合位图索引。
9.对于复合索引，至少要引用到索引列中的第一个列才会使用该索引。
10.某列可以出现在多个索引中，但相同的某列或某几列无法多次建立索引。 --就是可以 有多个组合，但不能有重复组合，不同顺序的相同几列视为不同组合
11.索引建立后并不一定会被引用，ORACLE会分析整个SQL后做出最优的执行方式。
12.ORACLE会自动在主键约束和唯一约束列上建立唯一索引。
13.对于一般的B-TREE索引，通配符出现在搜索词的首位时不会引用索引
14.在索引列上使用<> !=号时，或对空值进行判断时，索引不会生效
15.rebiuld重建索引，减少碎片，提高效率。

索引的优缺点

优点

1.大大加快数据的检索速度;
2.创建唯一性索引，保证数据库表中每一行数据的唯一性;
3.加速表和表之间的连接;
4.在使用分组和排序子句进行数据检索时，可以显著减少查询中分组和排序的时间。

缺点

1.索引需要占物理空间。
2.当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，降低了数据的 维护速度。

SQL性能优化问题

SQL执行顺序from—where—group by—having—select—order by From后面的表执行顺序从右往左，从后往前
1、 如果结果集没有影响的关联，将小的表放在后面
2、 Where条件顺序，将过滤条数大的放在后面，过滤条数小的放在前面
3、 尽量减少对表的重复查询
4、 使用exists代替in：in后面用子查询，用exists代替in（看exists子查询中where条件， 结果返回true或者fasle），如果in后面是具体的值，还是用in，用in的SQL语句总是多了 一种转换过程
5、 Distict,查询效率低，要先排序，再去重
6、 索引正确使用，不能使用聚合函数，不能使用not
7、 大于等于效率要高于大于，用>=替代>，前提是整数相比较
8、 Like效率低，使用instr代替instr(name,'n')>=1可以代替like'%c%'
9、 Where 是过滤行，having对分组的过滤
10、 要查看执行计划(F5, EXPLAIN)
11、 对 WHERE + ORDER BY 组合的优化, 在where 进行限制后 在进行 order BY
12、 尽量少排序 ORDER BY
13、 任何地方都不要使用select * from表，用具体的字段列表代替"*"，不要返回用不到 的任何字段
14、 尽量用 JOIN 替换子查询
15、 尽量少使用 OR ,索引失效
16、 尽量避免 UNION 使用 UNION ALL 然后再 GROUP BY 去重
17、 尽早过滤数据, WHERE 过滤 使用 join时 先过滤再 JOIN
18、 尽量避免一条 UPDATE 更新多条记录, 用 MERGE INTO , 效率比 UPDATE 高

作业练习

```
--1.给emp表的ename列创建一个索引，建立索引前后时间对比
select * from emp where ename like 'S%'
create index idx_emp_ename on emp(ename);
```

```sql
select * from emp where ename like 'S%'

--2.创建job和sal的复合索引，查询工资大于2000的MANAGER员工信息
create index idx_emp_jobsal on emp(job,sal)
select * from emp where sal>2000

--3.查询所有人选修c002课程及格的情况,自己判断如何创建索引
--在sc表的cno和score字段上创建复合索引
create index idx_sc_sccno on sc(cno,score)
select sc.*,case when score>=60 then '及格' else '不及格' end 及格情况
from sc where cno='c002'

--4.使用小写函数来创建索引，查询带有m的员工信息
create index idx_emp_ename on emp(lower(ename))
select * from emp where ename like 'm%'

--5.修改删除一个索引
alter index idx_emp_ename rename to ind_emp_ename1；--修改
drop index ind_emp_ename1 --删除
```