# 数据库要点

一．考点

## 事物ACID属性

* 原子性：
* 一致性：
* 隔离性：
* 持久性：

## 事务隔离级别

* 可串行化：高
* 可重复读
* 已提交读：提交的意思是事务提交
* 未提交读：低

## 范式

* 1NF:
* 2NF:
* 3NF:
* BCNF:
* 4NF:

## 连接

* 交叉连接(笛卡尔积) cross join / inner join 不写连接条件
* 内连接 inner join + 连接条件/ join + 连接条件
* 左外连接 left outer join
* 右外连接 right outer join
* 自然连接 natural join 两个表之间有相同名称的列，内连接的简化

## 数据库锁

* 读锁、写锁
* 表锁、页锁(某表中同一页的数据2KB-16KB)、行锁

## 数据库存储引擎

MySql允许为每个表单独选择或改变存储引擎

* InnoDB 行级锁定的事务引擎，可以使用外键约束
* MyISAM 表级锁定的非事务引擎
* MEMORY 内存表使用的非事务引擎
* BDB 页级锁定的事务引擎
* Merge 使多个相同的MyISAM看起来像一个单表(表分割/分库分表)的专用引擎
* Maria 6.0.6版本中MyISAM的代替品，添加充分的恢复功能
* Falcon 行级锁定的高性能事务引擎
* Archive 存储大量未索引数据的专用引擎，主要用来存档

## 索引

索引是寻找资源中特定项目的一种机制，便捷化检索表中行和列的子集，而不需要检查表中的每行。数据库服务器使用索引定位表中的行。索引是一种以特定顺序保存的专用表，它并不包含实体中的所有数据，而是那些用于定位表中行的列，以及描述这些行的物理位置的信息。

有了索引后，如果索引有利于改善查询，查询优化器就可以选择索引。同时，如果索引不止一个，优化器会选择最有利的那个索引。

* MySql自动为主键和外键建立索引
* 唯一索引：要求索引列不允许出现重复值
* 多列索引

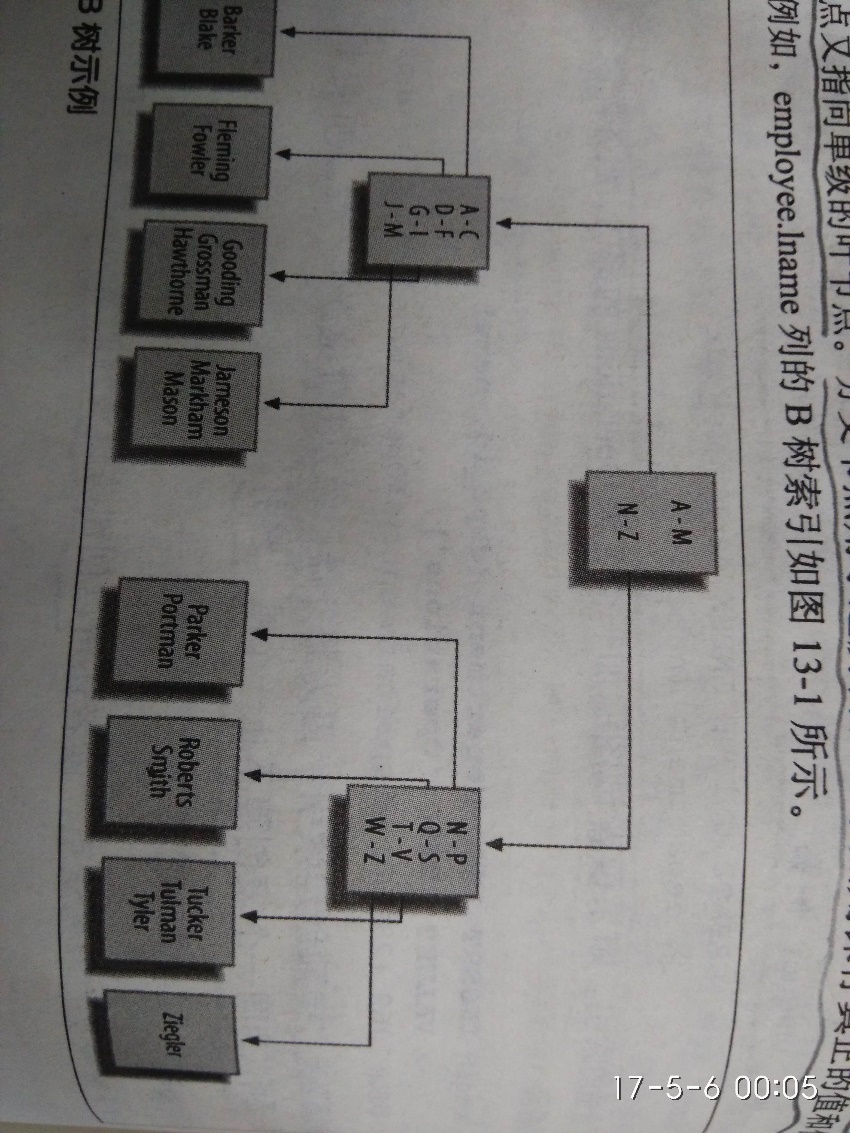
|  |
| --- |
| ALTER TABLE employee  ADD INDEX emp\_names\_idx (lname, fname) |

### 索引类型

* B树索引：MySql、Oracle和SQL Server的默认索引

B树索引是一种平衡树索引，它以树结构组织，有一个或多个分支节点，分支节点又指向单级的叶节点。分支节点用于遍历树，叶节点则保存真正的值和位置信息。

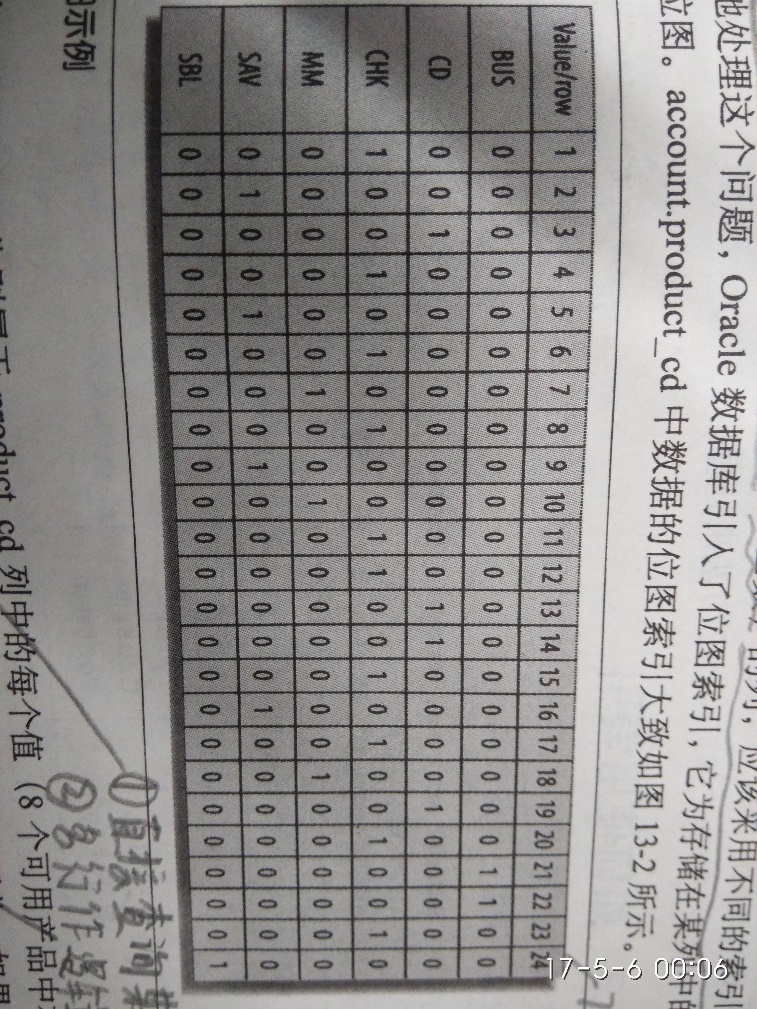
服务器通过增加或删除分支节点重新将值分配得更加均匀。通过保持树的均匀，不需要遍历多层分支节点，服务器就能够快速地到达叶节点查找到需要的值。



举例，通过上面的B树来查找以G开头的英文单词

* 位图索引：Oracle数据库引入

位图索引用于那些包含少量值却占据了大量行(所谓低基础)的列，它为存储在某列中的每个值生成一个位图。



对于高基数，位图索引会失败。

* 全文索引/文档索引：SQL Server、Oracle、MySql仅MyISAM引擎支持

全文索引是针对文档的专业的索引和搜索机制，方便用户在文档中查找单词或者短语。

### 使用索引

* 服务器首先利用索引快速定位特定表中的行，之后再访问相关表提取用户请求的补充信息。
* 如果索引包含满足查询的所有内容，那么服务器就不必访问相关表了。
* 使用explain语句显示查询的执行计划，可以查看索引使用。
* 为了维护索引，所以每次修改表数据时，都要更新索引，所以，索引不一定是好事，我们只在以下几种情况推荐使用索引：

1. 确保所有主键列被索引
2. 为所有被外键约束引用的列创建索引
3. 索引那些被频繁检索的列

## 常用关键字

* distinct
* order by (desc/asc)
* having
* (not) in
* (not) exists
* union/union all
* explain
* limit
* like

## 常用SQL语句

* create TABLE
* insert
* update
* delete
* drop
* select

## SQL聚集函数

* group by
* Max(), Min(), Avg(), Count(), Sum()
* having过滤条件中可以有聚集函数，包括未在select中出现的聚集函数
* having, where的过滤条件作用时间：having分组之前，where分组之后
* null值的处理：count(\*)会对行的数目计数，count(val)会对val列所包含的值的数目进行计数并且忽略所有遇到的null值
* 对多列的分组：group by product\_cd, open\_branch\_id;

|  |
| --- |
| SELECT product\_cd, SUM(avail\_balance) prod\_balance  FROM account  WHERE status = ‘ACTIVE’  GROUP BY product\_cd  HAVING MIN(avail\_balance) >= 1000 AND MAX(avail\_balance) <= 10000 |

二．注意事项

* 写SQL语句时，关键字要大写；
* 多表连接时，给表加上别名，用表别名.列来访问某表的某列

三．扩展点

## 内部函数

## 子查询 (包含查询，子查询)

* 子查询返回的结果集类型

1. 单列单行, 标量子查询，可用运算符 =, <>, <, >, <= ,>=。以上运算符不可以用于2) 3)
2. 单列多行, 可用运算符 (not) in, > all, >any
3. 多列多行

|  |
| --- |
| WHERE (open\_branch\_id, open\_emp\_id) IN  (SELECT branch.branch\_id, employee.emp\_id  ···); |

* 非关联子查询：子查询完全独立，可以单独执行
* 关联子查询

|  |
| --- |
| SELECT c.cust\_id, c.cust\_type\_cd, c.city  FROM customer c  WHERE 2 = (SELECT COUNT(\*)  FROM account a  WHERE a.cust\_id = c.cust\_id); |

执行过程：先从customer表中检索出XXX行客户记录，接着为每个客户执行一次子查询，每次执行包含查询都要向子查询传递客户ID。若子查询返回值2，则过滤条件满足，该行将被添加到结果集。

* 子查询同样可以用于update, delete等
* (not) exists

|  |
| --- |
| ···WHERE EXISTS (SELECT 1 FROM ···) |

这里，exists条件只是简单地检查子查询能否返回至少一行，与内容无关。

## 条件逻辑

条件逻辑是程序执行时从多个路径中选取其一的能力。

* 简单case表达式

|  |
| --- |
| CASE V0 //V0代表一个值  WHEN V1 THEN E1 //V1代表一个值,E1表达式，下同理  WHEN V2 THEN E2  ···  WHEN VN THEN EN  [ELSE ED]  END |

* 查找型case表达式

|  |
| --- |
| CASE  WHEN C1 THEN E1 //C1代表表达式条件  WHEN C2 THEN E2  ···  WHEN CN THEN EN  [ELSE ED]  END |

注意两者的区别，查找型case表达式更强大