数据库设计的第一范式

- 数据库表中的所有字段都只具有单一属性
- 单一属性的列是由基本的数据类型所构成的
- 设计出来的表都是简单的二维表

举例说明：

学号	姓名	性别	家庭信息	学校信息
20150001	李白	男	3口人，北京	硕士，研二
20150002	杜甫	男	2口人，南京	本科，大四
20150003	王维	男	4口人，上海	本科，大三
20150004	白居易	男	3口人，河南	硕士，研一
20150005	刘禹锡	男	4口人，上海	本科，大一
20150006	李清照	女	5口人，西安	硕士，研三
20150007	苏轼	男	2口人，南京	大专，大二
20150008	屈原	男	4口人，吉林	博士，博五
20150009	陶渊明	男	1口人，长沙	博士，博三

调整如下：

学号	姓名	性别	家庭人口	户籍	学历	所在年级
20150001	李白	男	3口人	北京	硕士	研二
20150002	杜甫	男	2口人	南京	本科	大四
20150003	王维	男	4口人	上海	本科	大三
20150004	白居易	男	3口人	河南	硕士	研一
20150005	刘禹锡	男	4口人	上海	本科	大一
20150006	李清照	女	5口人	西安	硕士	研三
20150007	苏轼	男	2口人	南京	大专	大二
20150008	屈原	男	4口人	吉林	博士	博五
20150009	陶渊明	男	1口人	长沙	博士	博三

数据库设计的第二范式

要求一个表中只具有一个业务主键，也就是说符合第二范式的表中不能存在非主键列的值对部分主键的依赖关系

举例说明：

订单号	产品号	产品数量	产品折扣	产品价格	订单金额	订单时间
2008003	205	100	0.9	8.9	2870	20080103
2008003	206	200	0.8	9.9	2870	20080103
2008005	207	200	0.75	10	2000	20080203
2008006	207	400	0.85	12	4800	20080206
2008007	207	1000	0.88	14	14000	20080209
2008008	210	240	0.95	8	12255	20100423
2008008	211	300	0.75	8	12255	20100423
2008008	212	350	0.8	15.9	12255	20100423

调整如下，

订单号	产品号	产品数量	产品折扣	产品价格
2008003	205	100	0.9	8.9
2008003	206	200	0.8	9.9
2008005	207	200	0.75	10
2008006	207	400	0.85	12
2008007	207	1000	0.88	14
2008008	210	240	0.95	8
2008008	211	300	0.75	8
2008008	212	350	0.8	15.9

订单号	订单金额	订单时间
2008003	2870	20080103
2008003	2870	20080103
2008005	2000	20080203
2008006	4800	20080206
2008007	14000	20080209
2008008	12255	20100423
2008008	12255	20100423
2008008	12255	20100423

数据库设计的第三范式

指每一个非主属性既不部分依赖于也不传递依赖于业务主键，也就是在第二范式的基础上消除了非主属性对主键的传递依赖

举例说明：

学号	姓名	性别	家庭人口	班主任姓名	班主任性别	班主任年龄
20150001	李白	男	3口人	陈洁	女	35
20150002	杜甫	男	2口人	陈洁	女	35
20150003	王维	男	4口人	陈洁	女	35
20150004	白居易	男	3口人	李丽	女	32
20150005	刘禹锡	男	4口人	李丽	女	32
20150006	李清照	女	5口人	王安	男	29
20150007	苏轼	男	2口人	南林	男	34
20150008	屈原	男	4口人	南林	男	34
20150009	陶渊明	男	1口人	王安	男	29

如下调整：

学号	姓名	性别	家庭人口	班主任姓名
20150001	李白	男	3口人	陈洁
20150002	杜甫	男	2口人	陈洁
20150003	王维	男	4口人	陈洁
20150004	白居易	男	3口人	李丽
20150005	刘禹锡	男	4口人	李丽
20150006	李清照	女	5口人	王安
20150007	苏轼	男	2口人	南林
20150008	屈原	男	4口人	南林
20150009	陶渊明	男	1口人	王安

班主任姓名	班主任性别	班主任年龄
陈洁	女	35
陈洁	女	35
陈洁	女	35
李丽	女	32
李丽	女	32
王安	男	29
南林	男	34
南林	男	34
王安	男	29

案例需求说明

电子商务网站的数据库结构

- 用户信息：{**用户名**，密码，手机号，姓名，注册日期，出生日期}
- 商品信息：{商品名称，出版社名称，图书价格，图书描述，作者}
- 分类信息：{分类名称，分类描述}
- 商品分类(对应关系表)：{商品名称，分类名称}
- 订单表：{订单编号，下单用户名，下单日期，支付金额，物流单号}
- 订单商品关联表：{订单编号，订单商品分类，订单商品名，商品数量}

场景一：编写SQL查询出每一个用户的订单总金额

```
select 下单用户名,sum(d.商品价格*b.商品数量)
from 订单表 a join 订单商品关联表 b on a.订单编号=b.订单编号
join 商品分类关联表 C on C.商品名称=b.商品名称
and C.分类名称=b.订单商品分类
join 商品信息表 d on d.商品名称 = c.商品名称
group by 下单用户名
```

场景二：假设下单用户就是商品的收货人，我们在发货前一定要查询出每个订单的下单人的信息，而这些信息全部记录在用户信息表中。

编写SQL查询出下单用户和订单详情

```
select a.订单编号,e.用户名,e.手机号,d.商品名称,C.商品数量,d.商品价格
from 订单表 a join 订单商品关联表 b on a.订单编号=b.订单编号
join 商品分类关联表 C on C.商品名称=b.商品名称
and c.分类名称=b.订单商品分类
join 商品信息表 d on d.商品名称= c.商品名称
join 用户信息表 e on e.用户名=a.下单用户名
```

什么叫做反范式化设计

反范式化是针对范式化而言的，在前面介绍了数据库设计的范式，所谓的反范式化就是为了性能和读取效率的考虑 而适当的对数据库设计范式的要求进行违反，而允许存在少量的数据冗余，换句话来说反范式化就是使用空间来换取时间

反范式化改造

商品信息：{商品名称，分类名称，出版社名称，图书价格，图书描述，作者}

分类信息：{分类名称，分类描述}

订单表：{订单编号，下单用户名，手机号，下单日期，支付金额，物流单号，订单金额}

订单商品关联表：{订单编号，订单商品分类，订单商品名，商品数量，商品单价}

编写SQL查询出每一个用户的订单总金额

```
select 下单用户名,sum(订单金额)
from 订单表
group by 下单用户名;
```

反范式化改造后的查询 编写SQL查询出下单用户和订单详情

```
select a.订单编号,a.用户名,a.手机号,b.商品名称,b.商品单价,b.商品数量
from 订单表 a
join 订单商品关联表 b on a.订单编号=b.订单编号;
```

数据库、表及字段的设计规范

整数类型

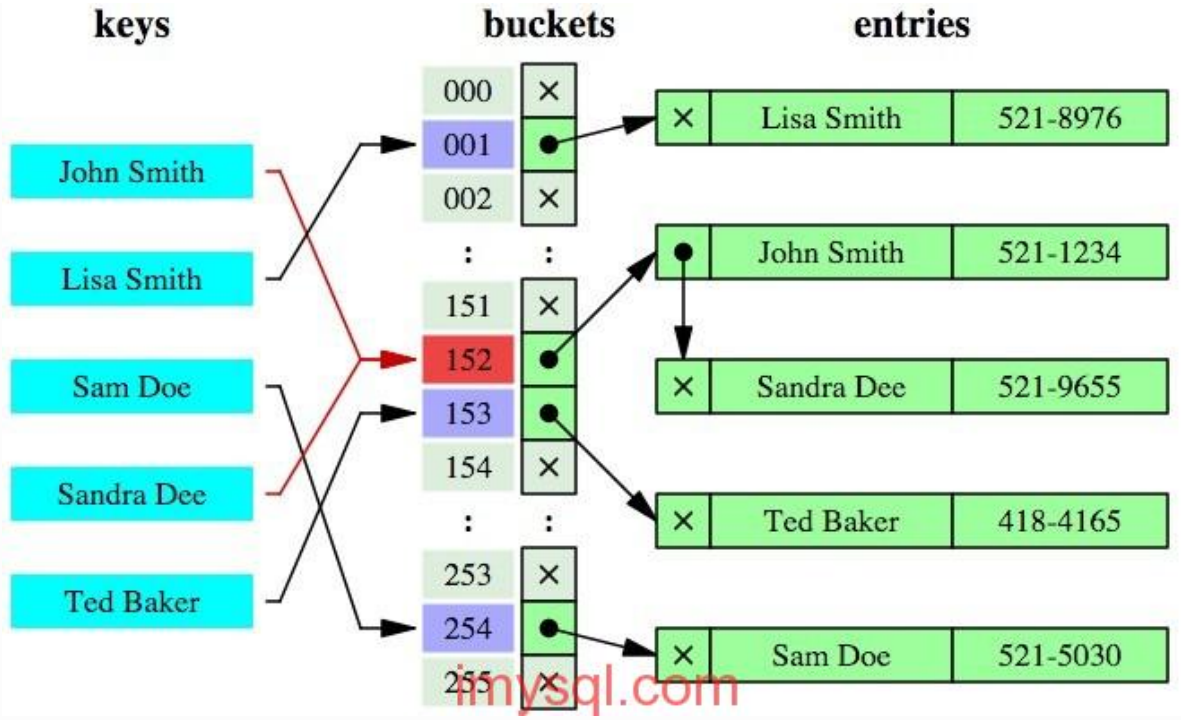
实数类型

字符串类型

日期类型

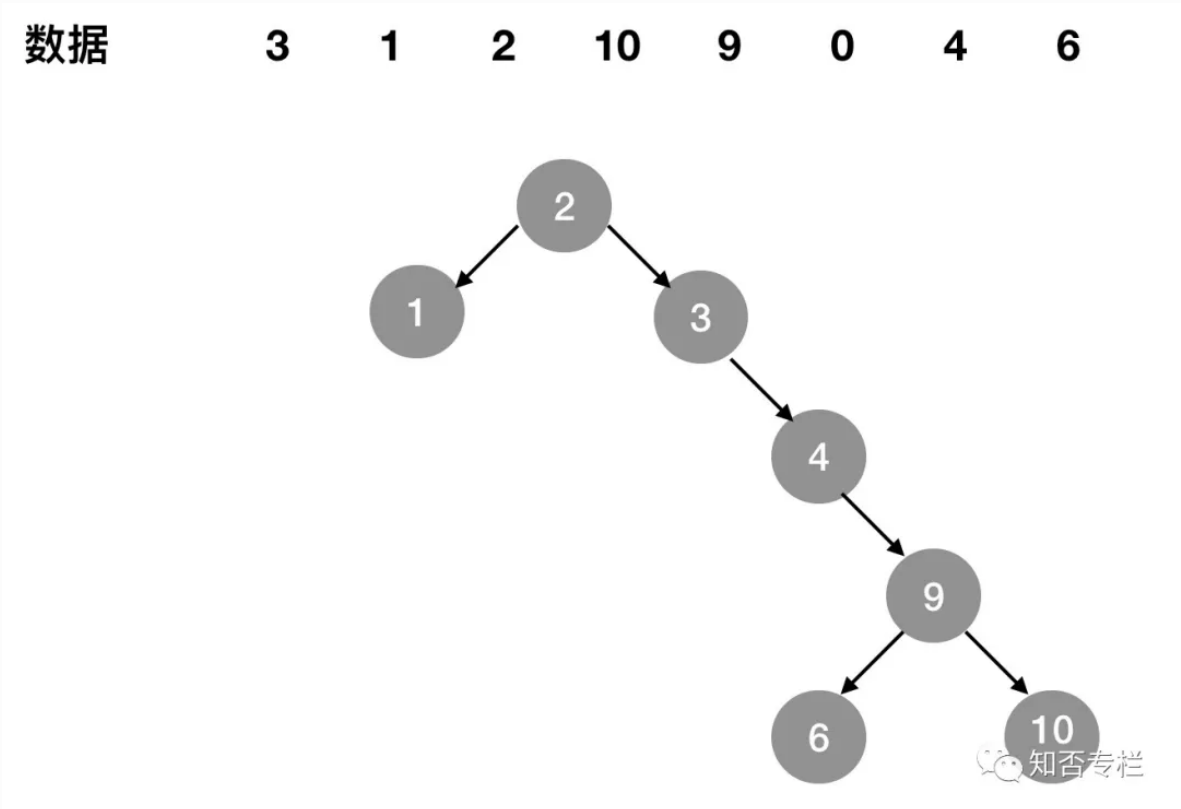
索引

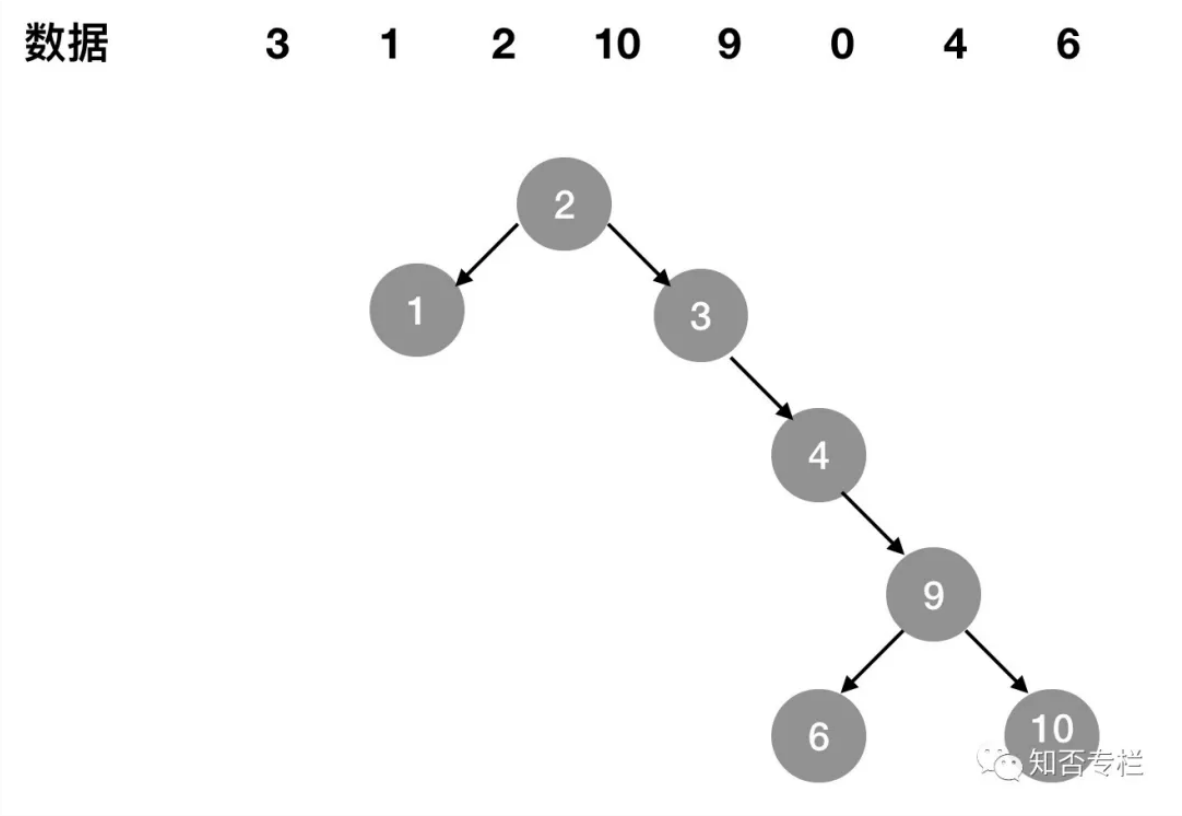
hash


keys
buckets
entries
John Smith
Lisa Smith
Sam Doe
Sandra Dee
Ted Baker
000
001
002
151
152
153
154
253
254
255
Lisa Smith
521-8976
John Smith
521-1234
Sandra Dee
521-9655
Ted Baker
418-4165
Sam Doe
521-5030
imysql.com

数组、链表、红黑树

二叉树


数据
3
1
2
10
9
0
4
6
2
1
3
4
9
6
10
知否专栏



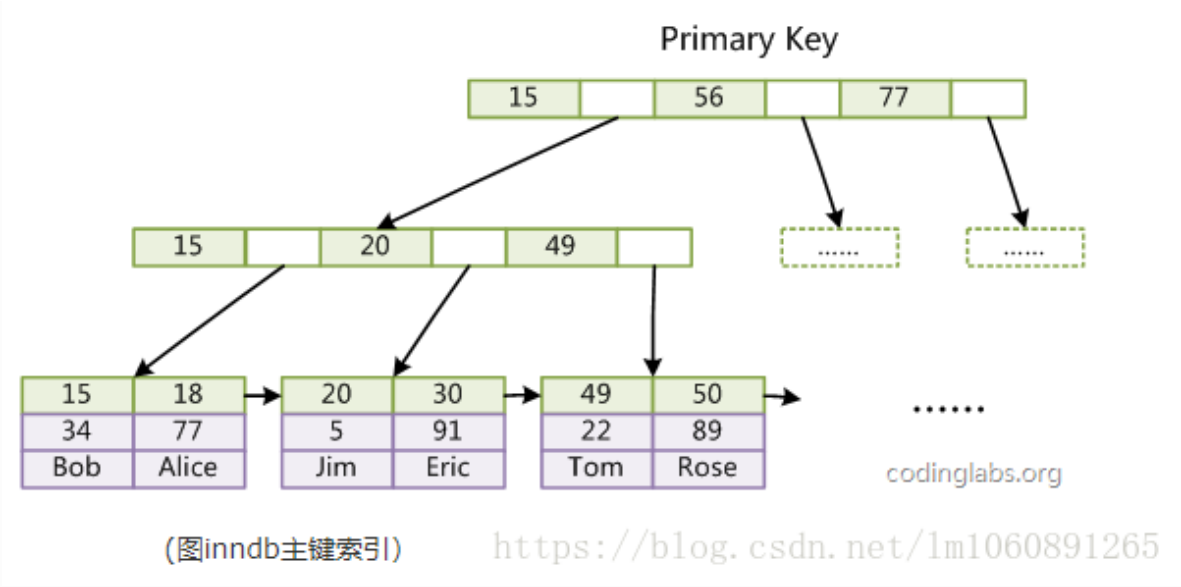
索引实现

对比

	MyISAM	InnoDB
事务	不支持	支持
读写效率	高	低
索引	支持全文索引	不支持全文索引
外键	不支持	支持
锁	表锁	行锁
文件存储形式	*.MYD *.MYI *.FRM	*.FRM(默认为共享表空间，可修改)
适用场景	大量select语句	大量update语句
删除表后数据文件是否存在	自动清除	不自动清除

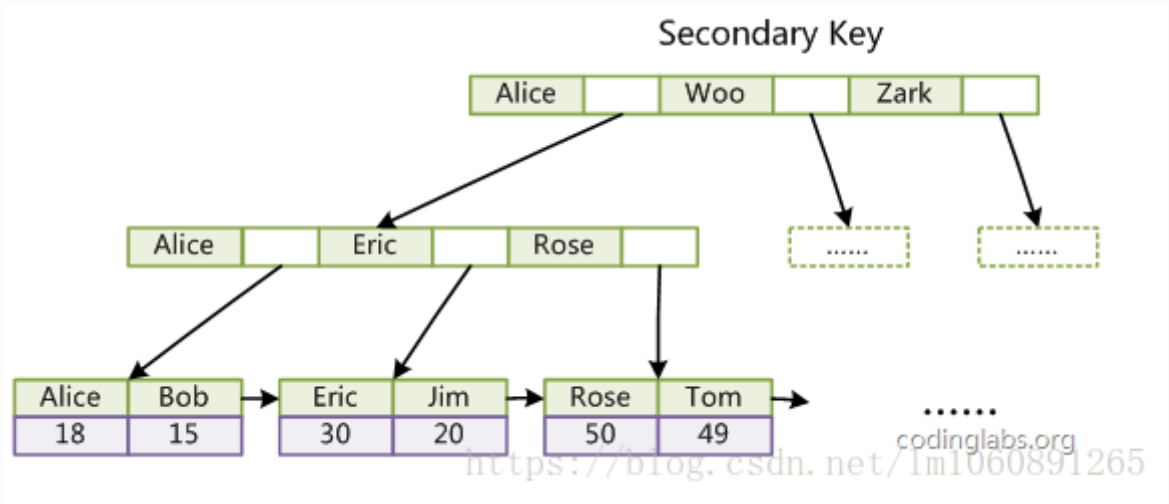
InnoDB索引实现

主键索引



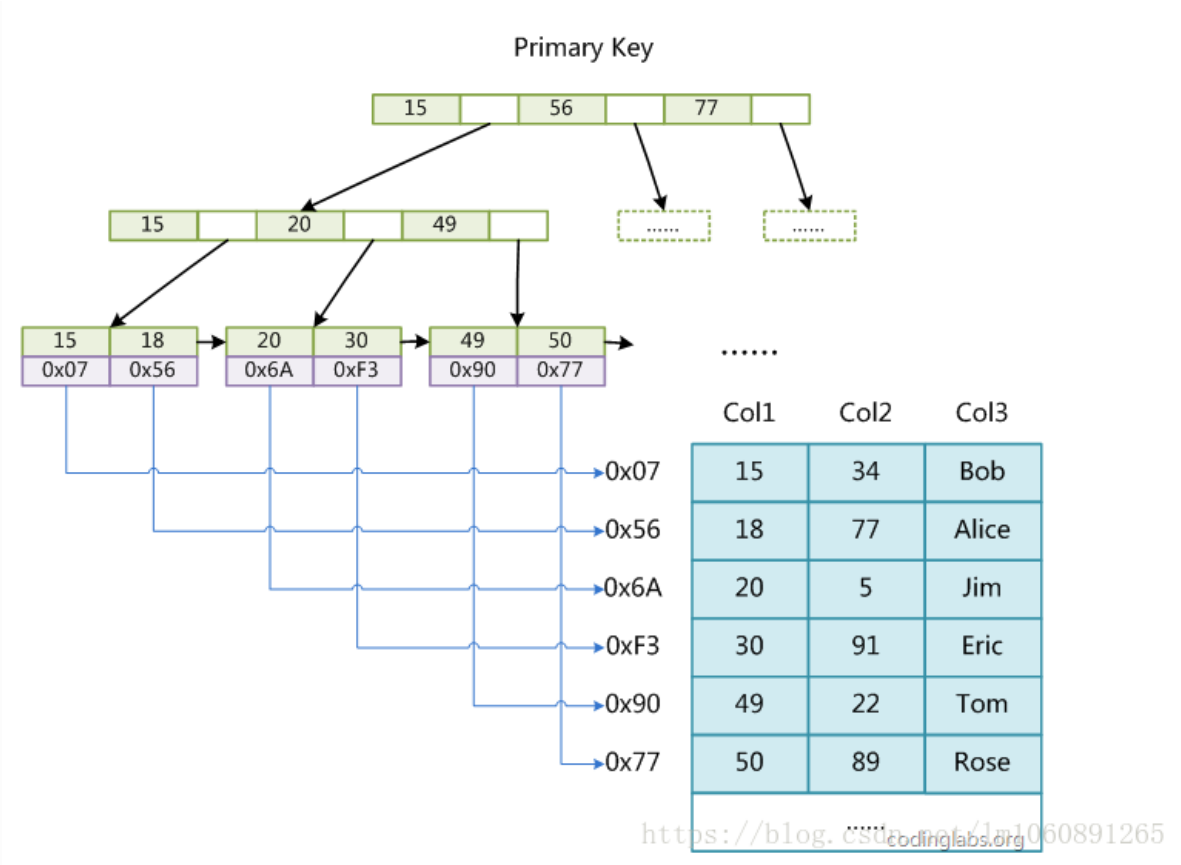
(图inndb主键索引)

二级索引（辅助索引）

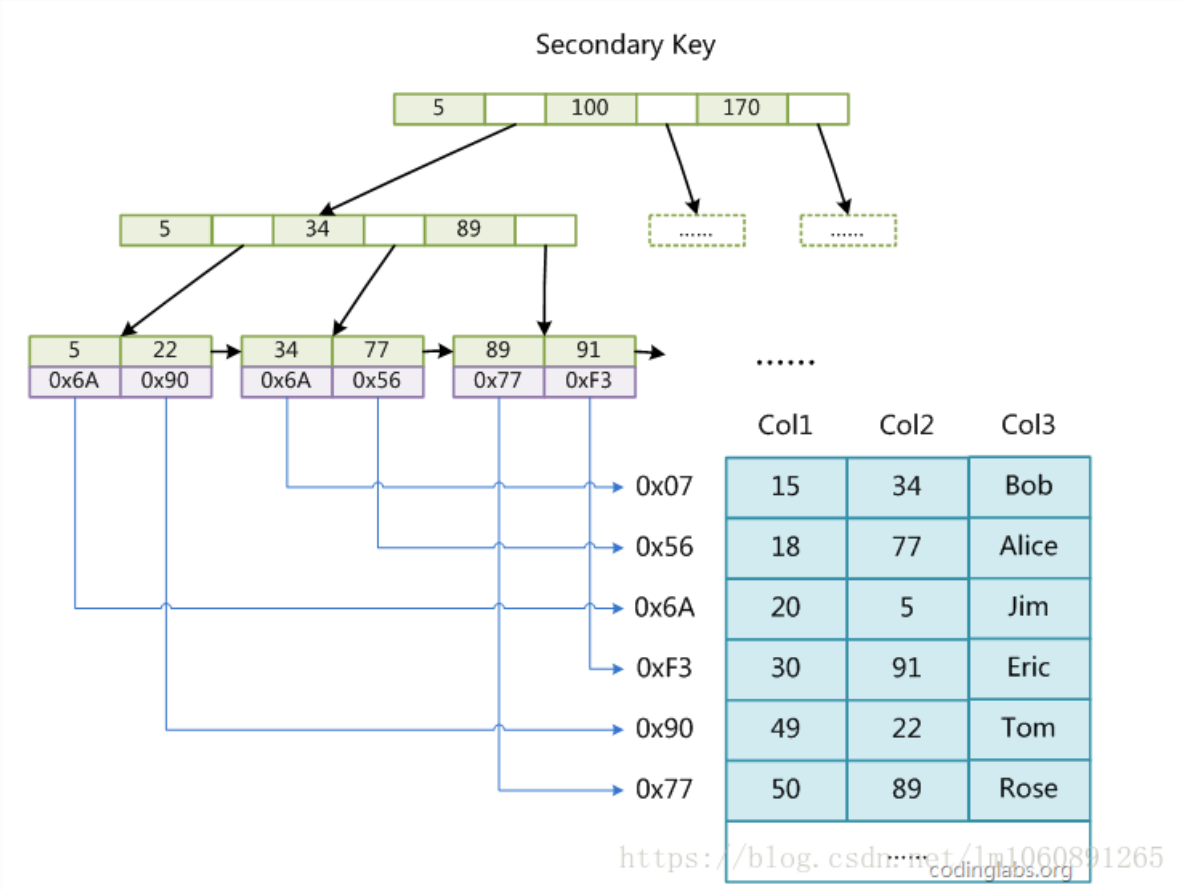


Myisam索引实现

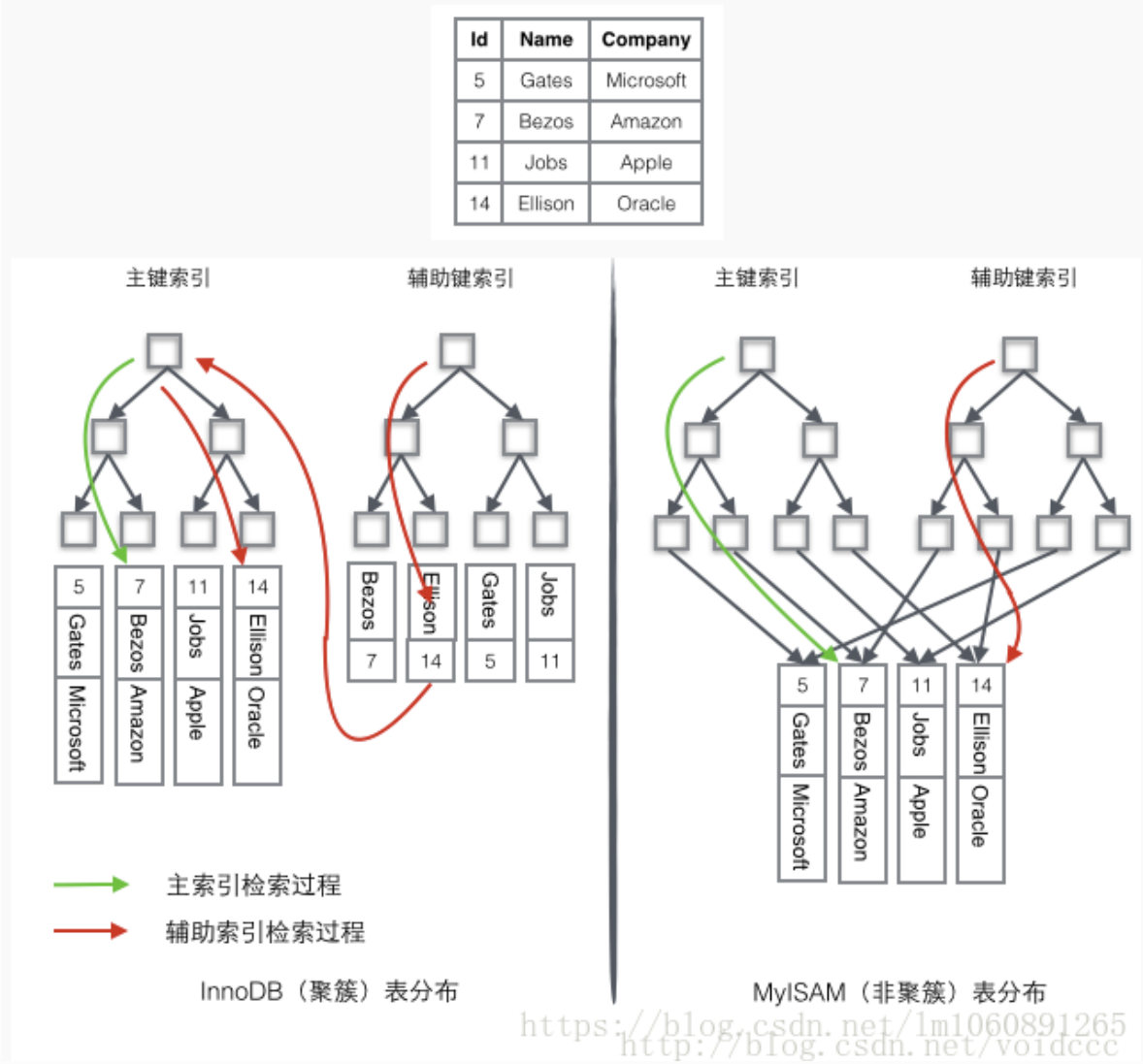
主键索引



二级索引（辅助索引）



聚簇索引和非聚簇索引对比



覆盖索引（covering index）指一个查询语句的执行只用从索引中就能够取得，不必从数据表中读取，聚簇索引的叶子节点存放的是主键值和数据行，支持覆盖索引。

索引实战

```
CREATE TABLE `emp` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(24) NOT NULL DEFAULT '' COMMENT '姓名',
  `age` int(11) NOT NULL DEFAULT '0' COMMENT '年龄',
  `job` varchar(20) NOT NULL DEFAULT '' COMMENT '职位',
  `add_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '入职时间',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8 COMMENT='员工记录表';

ALTER TABLE emp ADD INDEX idx_emp_nameAgeJob (NAME, age, job)
```

测试

```
explain select * from emp where name = 'z3'
explain select * from emp where name = 'z3' and job= 'manage'
explain select * from emp where name = 'z3' and age ='22' and job= 'manage'
-- 尽量全值匹配
-- 左前缀匹配

explain SELECT * FROM EMP WHERE left(name,4) = "july"
```

```sql
select * from emp where id = 3
explain select * from emp where id + 1 = 2

-- 覆盖索引
select name from emp where  name = 'z3'

select * from emp where name != 'z3'

select * from emp where name is not null

-- like
select * from emp where name like '%july%'

-- 数字和字符串转换
-- 表关联的字符集不同
explain select * from emp where id = "1"

select "10" > 4
```

执行计划

1、id

```sql
EXPLAIN SELECT
    t.*
FROM
    course c,
    teacher t,
    teacher_card tc
WHERE
    c.tid = t.tid
AND t.tcid = tc.tcid
AND (c.cid = 2 OR t.tcid = 3)


explain SELECT
  tc.tcdesc
FROM
  teacher_card tc
WHERE
  tc.tcid = (
SELECT
  t.tcid
FROM
  teacher t
WHERE
  t.tid = ( SELECT c.tid FROM course c WHERE c.cname = "sql" )
  );
```

2、select_type

```
explain SELECT
  cr.cname
FROM
  ( SELECT * FROM course WHERE tid = 1 UNION SELECT * FROM course WHERE tid = 2 )
cr
```

3、type

const、system、 eq_ref、ref、range、index、all

4、possible_keys、key、key_len

5、ref、rows、extra

慢查询

1、已经执行完成的SQL

查看慢查询是否开启

`show VARIABLES like 'slow_query_log';`

慢查询的日志存放位置

`show VARIABLES like 'slow_query_log_file';`

慢查询的时间 `show VARIABLES like 'long_query_time';`

2、正在执行的SQL

`show processlist`

mysqldumpslow

`mysqldumpslow -s r -t 10 /var/lib/mysql/localhost-slow.log`

pt-query-digest

`pt-query-digest --explain u=root,p=123456 /var/lib/mysql/localhost-slow.log`

MySQL锁

共享锁

读锁，S锁

```
begin

select * from lock1 where id = 3 lock in share mode

commit

-----------------------
```

```
begin

select * from lock1 where id = 3 lock in share mode

update lock1 set num = 45 where id = 4

COMMIT
```

排它锁

写锁，X锁

```
begin

select * from lock1 where num = 90 for update

select * from lock1 where id = 3 lock in share mode

update lock1 set num = 45 where id = 4

COMMIT


begin

select * from lock1 where id = 7 for update

select * from lock1 where num = 50 for update

commit
```

意向锁

意向排它锁(IX)

意向共享锁(IS)

表锁、IX和IS相互之间都兼容

临键锁

Next-key，左开右闭，范围查询并且有数据命中

```
begin

select * from lock1 where id = 8 for update

update lock1 set num = 44 where id = 5

delete from lock1 where id = 5

insert into lock1(id,num) VALUE(4,47)

select * from lock1 where num = 50 for update

commit
```

间隙锁（GAP）

查询没有记录命中，退化成间隙锁