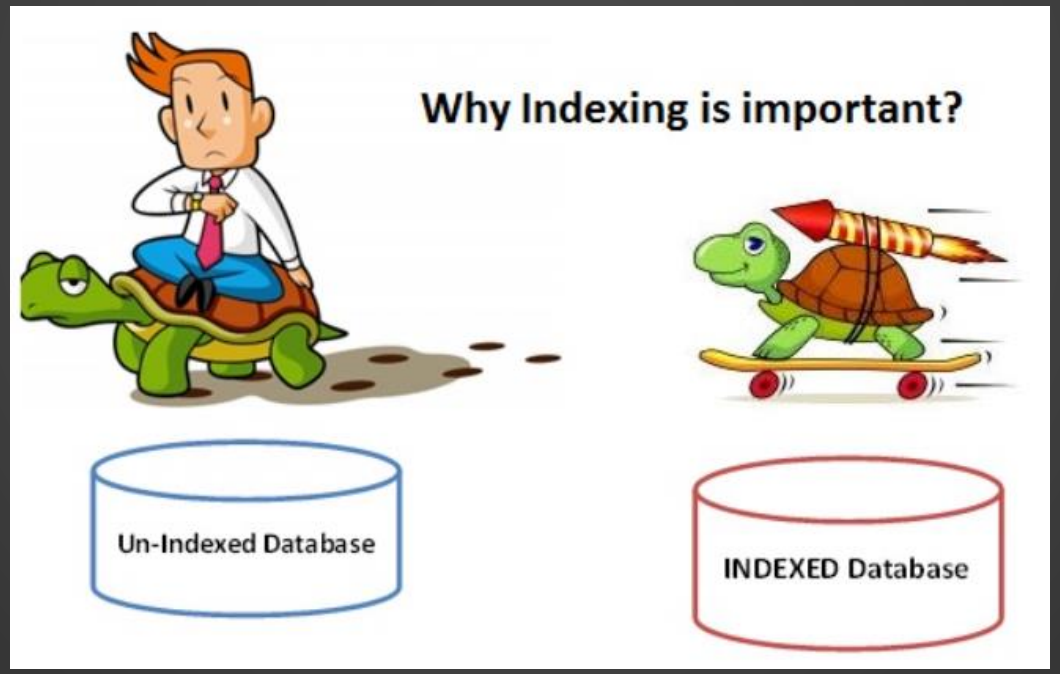
索引

索引

- 索引（index），类似于汉语字典的目录（索引），通过字典的目录，可以按拼音、笔画、偏旁部首等排序的目录（索引）快速查找到需要的字。



索引

假设订单表orders，表的数据有10W条数据，其中有一条数据是oid='US-2017-1357144'，如果查找这条记录，SQL语句就是：

```
select * from orders where oid='US-2017-1357144';
```

- 如果在没有建立索引的情况下，MySQL需要扫描这10W条数据找这条数据；
- 如果在oid上建立索引，那么MySQL只需要扫描一行数据便可找到这条oid='US-2017-1357144'的记录。

这就是索引的作用！

索引

- 索引的作用是提高数据检索的效率，而检索效率的实现的本质是数据结构，通过不同数据结构的选择，实现各种数据快速检索。
- MySQL的底层数据引擎最常见的是InnoDB引擎，InnoDB使用的是B+树。
- MySQL的索引用的就是B+树，B+树在查找效率、范围查找中都有着非常不错的性能。

索引

查看索引：show index from orders;

- Table:表名
- Non_uqiue:0表示索引不能包括重复值，即唯一索引，1表示可以有重复值
- Key_name:索引名称，primary，主键索引，主键是一个特殊的索引
- Seq_in_index:该列在索引中的位置，目前这个索引是单列的，所以该列的值时1，如果是组合索引，则表示每列在索引中的顺序。
- Column_name:表示定义索引的列字段。
- Collation:表示列以何种顺序存储在索引中
- Cardinality:索引中唯一值数目的估计值，只是估计值。
- Index_type:索引类型，主要由FULLTEXT，HASH，BTREE，RTREE，这里使用的事BTREE，B树。
- Visible:是否可见，是。

索引

- 创建索引

```
1. create index oid_index on orders(oid);
2. alter table orders add index cid_index(customerid);
```

- 删除索引: `drop index cid_index on orders;`

代码演示。

explain语句

explain：分析查询语句使用索引的情况。

- explain select * from orders where oid='US-2017-1357144';
- explain select * from orders;
- explain select count(*) from orders;

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	orders	NULL	ref	oid_index	oid_index	82	const	2	100.00	NULL

explain语句执行结果说明

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	orders	NULL	ref	oid_index	oid_index	82	const	2	100.00	NULL

explain语句执行结果说明：

- id：选择标识符，select的查询序号，
- select_type：select的类型，例如，SIMPLE：简单SELECT，不使用UNION或子查询等。
- table：查询所访问的数据表
- type：表示查询的类型，常用的类型有：ALL、index、range、ref、eq_ref、const、system、NULL，从左到右，性能从差到好
  1. ALL：Full Table Scan， MySQL将遍历全表以找到匹配的行
  2. ref：非唯一性索引扫描，返回匹配某个值的所有行
  3. NULL：MySQL在优化过程中分解查询语句，执行时不用访问表或索引

explain语句执行结果说明

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	orders	NULL	ref	oid_index	oid_index	82	const	2	100.00	NULL

explain语句执行结果说明：

- possible_keys：表示查询时，可能使用的索引
- key：表示实际使用的索引，必然包含在possible_keys中
- key_len：表示索引字段的长度，也就是索引中使用的字节数
- ref：显示索引的哪一列被使用

explain语句说明

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	orders	NULL	ref	oid_index	oid_index	82	const	2	100.00	NULL

explain语句执行结果说明:

- rows: 扫描出的行数(估算的行数)
- filtered: 查询的表行占表的百分比
- Extra: 执行情况的描述和说明

1. Using index:select的数据列只用从索引中就能够取得，不必读取数据行
2. Using where:说明使用了where条件

索引的优缺点

- 使用索引的优点：

1. 可以通过建立唯一索引或者主键索引，保证数据库表中每一行数据的唯一性。
2. 建立索引可以大大提高检索数据的效率，以及减少表的检索行数
3. 可以加速表与表之间的相连

- 使用索引的缺点：

1. 在创建索引和维护索引会耗费时间，随着数据量的增加而增加。
2. 索引文件会占用物理空间，除了数据表需要占用物理空间之外，每一个索引还会占用一定的物理空间
3. 当对表的数据进行INSERT，UPDATE，DELETE的时候，索引也要动态的维护，这样就会降低数据的维护速度

函数

MySQL中的函数

- 函数：帮助我们实现特定功能的SQL语句集合，分为：内置函数和自定义函数
- 内置函数：count()、sum()、avg()、current_time等
- 自定义函数：用户自己编写的函数，用于实现特定的功能

- 例如，从订单表orders中查询某个顾客购买的产品数量：

```
select customername,sum(salesnum) from orders
      where customername='马丽' group by customername;
select customername,sum(salesnum) from orders
      where customername='邹涛' group by customername;
```

自定义函数

- 自定义函数常用操作：

1. 创建自定义函数
2. 调用自定义函数
3. 删除自定义函数

自定义函数

- 创建自定义函数：

```
delimiter //
create
      function customer_sales(ct_name varchar(10))
      returns int
begin
      函数体语句;
      return sales_num;
end //
delimiter ;
```

自定义函数

- 调用自定义函数:

```
select customer_sales('邹涛');
```

- 删除自定义函数:

```
drop function customer_sales;
```

存储过程

存储过程

- 前面学习了函数，函数能够返回一个特定的值，数值或者字符串。
- 如果我们希望返回一个结果集，就无法使用函数了，需要存储过程。
- 例如，从订单表orders中查询某个顾客的订单记录：

```
select * from orders where customername='邹涛';
select * from orders where customername='马丽';
```

- 存储过程： procedure，存储在数据库服务器中的一组sql语句，通过在查询中调用一个指定的名称来执行这些sql语句命令。

存储过程

- 存储过程常用操作：

1. 创建存储过程
2. 调用存储过程
3. 删除存储过程

存储过程

- 创建存储过程

```
delimiter //
create procedure ct_info(ct_name varchar(10))
begin
select * from orders where customername=ct_name;
...... #SQL语句
end //
delimiter ;
```

存储过程

- 调用存储过程：

```
call ct_info('马丽');
```

- 删除存储过程：

```
drop procedure ct_info;
```

存储过程 vs 函数

- 相同点：

1. 都是一组SQL语句的集合，用于完成特定的功能
2. 都需要先定义，再调用

- 不同点：

1. 函数定义的关键字：function，存储过程定义的关键字：procedure
2. 函数的定义较为严格，需要定义返回值类型、返回值，而存储过程的定义较为宽松，不需要这些
3. 存储过程能够返回一个结果集，而函数只能返回一个结果
4. 调用函数使用select，调用存储过程使用call

游标

游标

- 思考问题：例如，通过存储过程得到了一个结果集，现在需要对这个结果集进行遍历，如何进行?
- 答案：游标，利用循环+游标对结果集进行遍历

- 游标（cursor）：游动的标识，对于一个存储在MySQL服务器上的数据库查询，即被select语句检索出来的结果集，在存储了游标之后，可以根据需要滚动或浏览其中的数据
- 游标多用于存储过程

游标

- 游标使用步骤：

1. 创建游标：`declare days_cursor cursor for select`查询语句；
2. 打开游标：`open days_cursor;`
3. 使用游标：`fetch days_cursor into para_id,...;`
4. 关闭游标：`close days_cursor;`

MySQL中的循环

- MySQL中的循环

```
declare done int default 0;
declare continue handler for not FOUND set done = 1;
repeat
      ..... #循环体语句
until done end repeat;
```

游标

- 举个例子，从order表中根据姓名查询出某个顾客的订单记录（结果集），然后将该结果集中id、oid，及根据senddate和orderdate计算出来的发货天数，插入新建的表中。
- 实现步骤：

1. 建表
2. 定义存储过程，在存储过程中实现以下两个操作：
   a) 将指定顾客的订单记录查询出来
   b) 利用循环+游标，将上述查询出来的记录中的id、oid，及根据senddate和orderdate计算出来的发货天数，插入新建的表中。

事务

事务

事务（transaction）：一组sql语句，用来维护数据库的完整性，保证成批的sql操作要么完全执行，要么完全不执行

例如，银行账户转账业务，客户A转账100元给客户B

- 客户A减去100元
- 客户B增加100元
- 必须同时成功或者同时失败。最小单元不可再分

事务

- 开启事务：begin;

start transaction;

- 事务回滚：ROLLBACK

撤销指定sql语句的过程

- 事务确认：COMMIT

将为存储的sql语句结果写入数据表

- 保留点：SAVEPOINT

事务处理中设置的临时占位符，可以对它发布回退