04 SQL优化

The Nature of SQL

南京大学软件学院

关系代数

- E.F.Codd 关系理论之父，关系代数究竟 有什么用?

什么是代数？-表达式的**等价变换**

(2+4)×3 =6×3 **=18**

=(2+2+2) ×3

=2×3+4×3

=2×3+(2+2) ×3

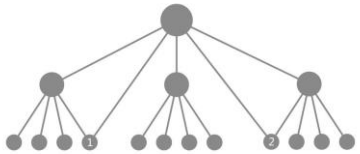
=……

关系代数也是一样

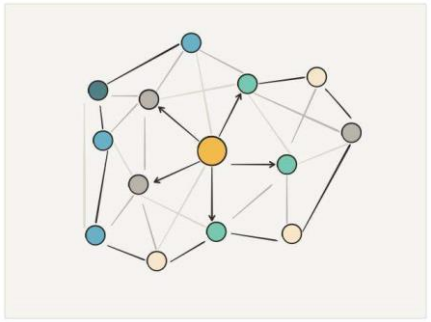
2、3、4 这些数字对应的就是 **关系（表）**

+ - × / 这些运算符对应的就是 **关系操作**

关系代数使数据库变成了科学而不是艺术



层次型数据库



网状数据库

RDBMS
关系型数据库

SQL操作：不需要去考虑操作实现细节

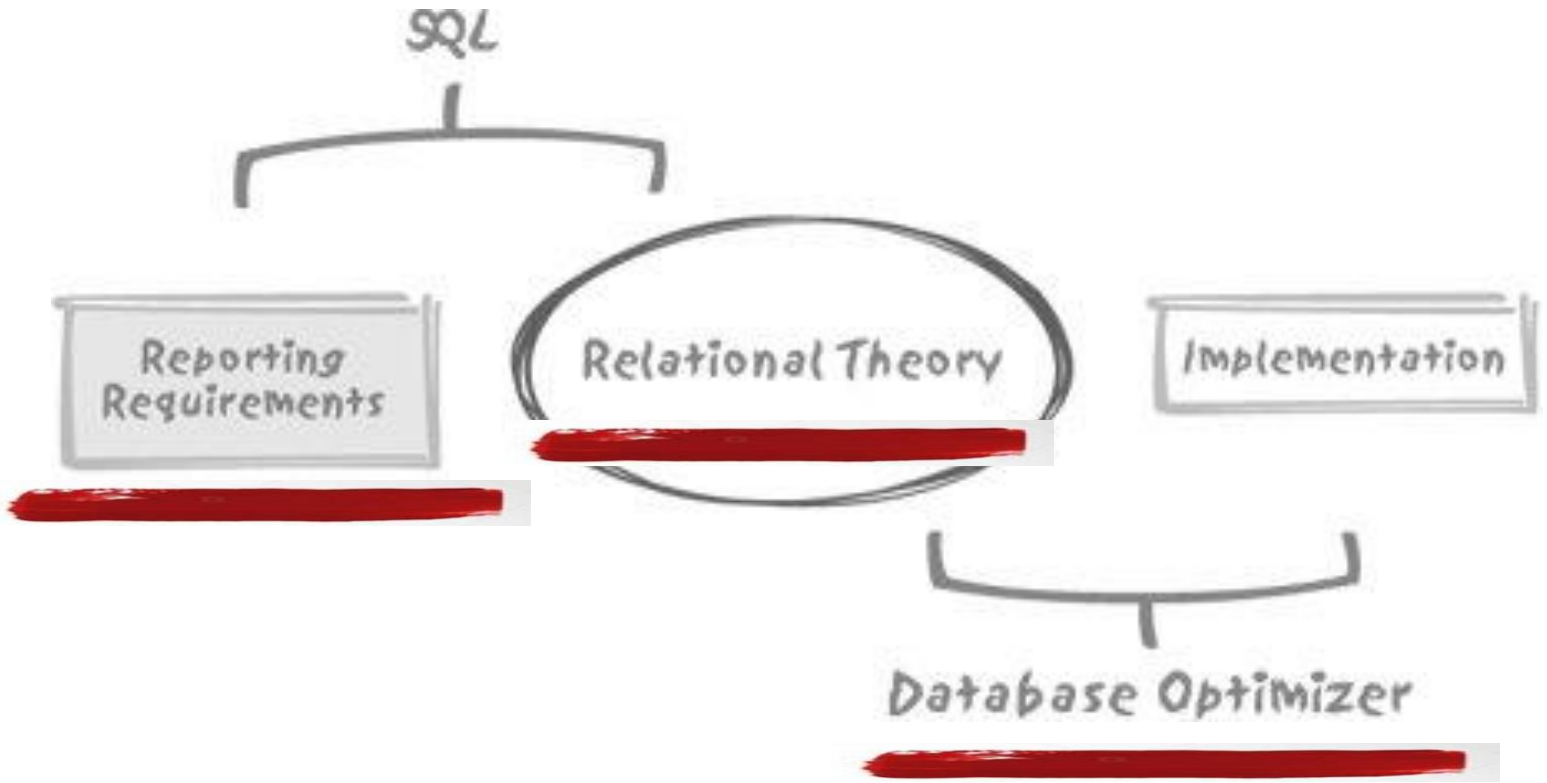
关系设计：拥有理论上的普遍规则

关系数据库变成了科学，也变成了黑盒

NoSQL

Redis Cassandra CouchDB mongoDB membase riak

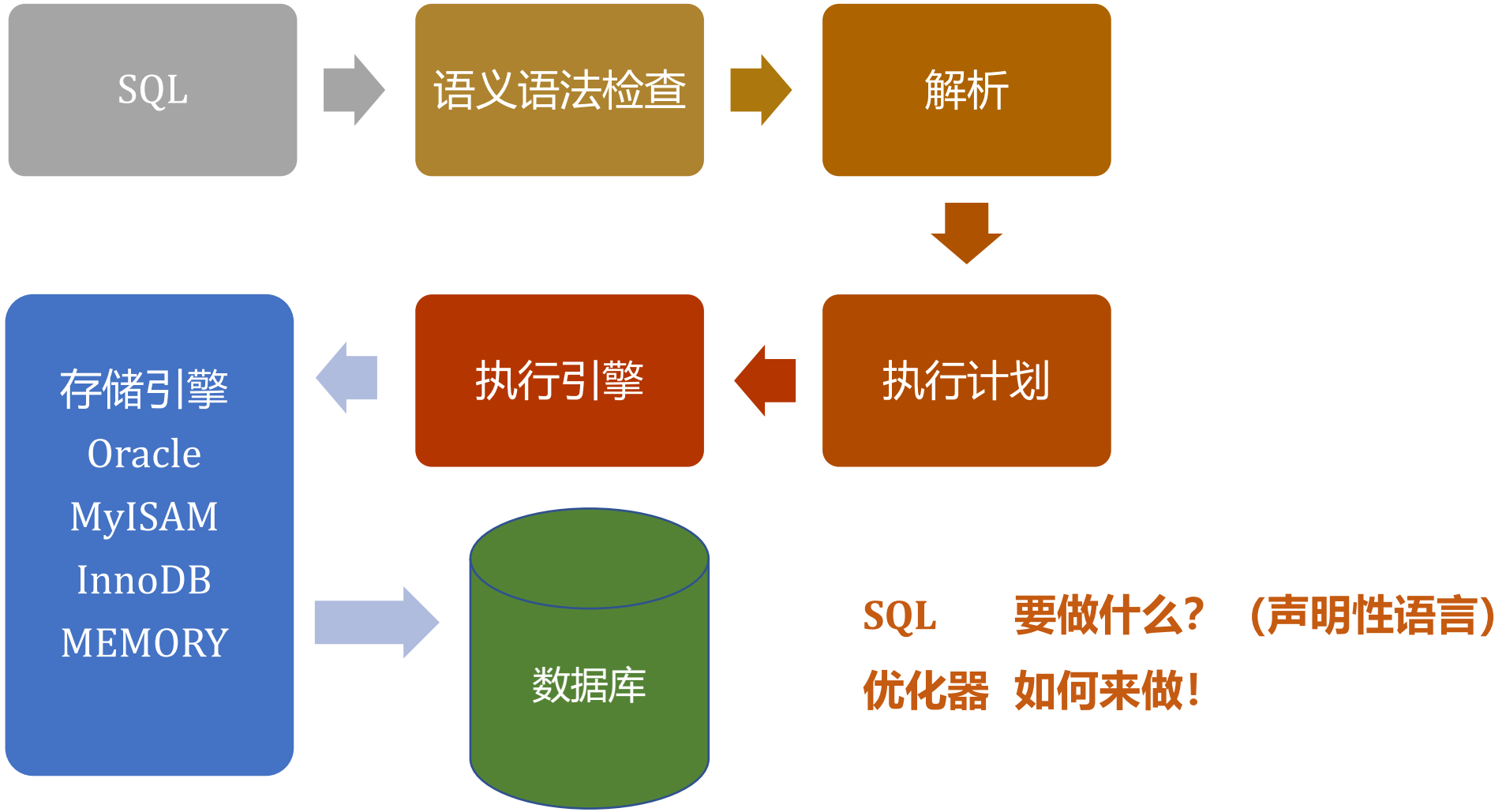
SQL与查询优化器


SQL
Reporting Requirements
Relational Theory
Implementation
Database Optimizer

SQL和查询优化器

- 优化器借助关系理论提供的语义无误的原始查询进行**有效的等价变换**
- 优化器根据数据库的实际实现情况对理论上等价的不同优化方案做出**权衡**，
- 产生可能的最优查询执行方案

SQL的执行顺序



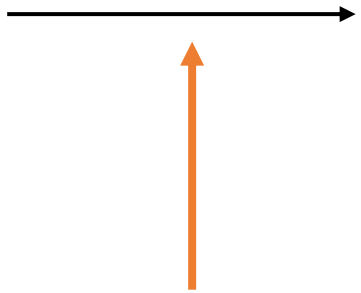
SQL　　要做什么？（声明性语言）

优化器　如何来做！

软解析：绑定变量

安全问题：

```
String sql = "SELECT id,nick FROM user
WHERE username='"+username+"' AND
password='"+password+"'";
```



实际传到DBMS的SQL为：

```
SELECT id,nick FROM user WHERE
username='test' AND password='1' or '1'='1'
```

攻击者输入的password为 **1’ or ‘1’=’1**

优化器只能对关系领域进行优化

- 忽略这点很可能出现错误

例子：查询不是经理的员工当中，哪五个人收入最高？

```
select empname, salary
   from employees
   where status != 'EXECUTIVE'
    and rownum <= 5
   order by salary desc
```

```
select *
   from (select empname, salary
         from employees
         where status != 'EXECUTIVE'
         order by salary desc)
   where rownum <= 5
```

优化器的有效范围

- 优化器需要借助数据库中找到的信息
- 能够进行数学意义上的等价变换
- 优化器考虑整体响应时间
- 优化器改善的是独立的查询

Practice in class 4-1

- Oracle的rownum是一个非常讨厌的SQL方言，但它是Oracle数据库中唯一的限定返回行数的函数，其它数据库也有类似的方言
  - DB2使用FETCH FIRST子句
  - MySQL和PostgreSQL使用LIMIT子句
  - SQL Server使用TOP关键字
  - 请你用你手上常用的数据库试一下本课程那个限定返回行数查询的例子，看看有没有Oracle出现的问题
  - 如果你是用Oracle，你试一下，你能通过rownum=5，来返回第5行记录嘛？

使用SQL需要考虑的因素

- 获得结果集所需访问的数据量
- 定义结果集所需的查询条件
- 结果集的大小
- 获得结果集所涉及的表的数量
- 同时修改这些数据用户的多少

数据总量

- SQL考虑最重要因素：必须访问的数据总量
- 没有确定目标容量之前，很难断定查询执行的效率

定义结果集的查询条件

- Where子句，特别在子查询或视图中可能有多个where子句
- 过滤条件的效率有高有低，受到其他因素的影响很大

- 影响因素：过滤条件、主要SQL语句、庞大的数据量对查询的影响

结果集的大小

- 查询所返回的数据量，重要而被忽略
- 取决于表的大小和过滤条件的细节
- 例外是若干个独立使用效率不高的条件结合起来效率非常高
- 从技术角度来看，查询结果集的大小并不重要，重要的是用户的感觉

- 熟练的开发者应该努力使响应时间与返回的记录数成比例

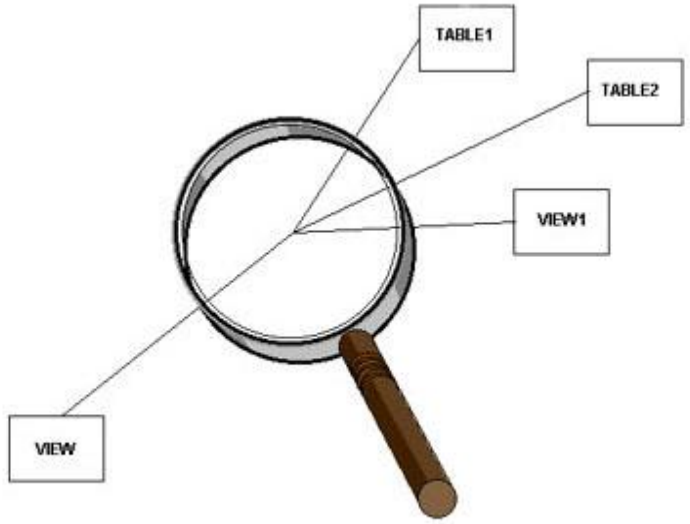
表的数量

- 表的数量会对性能有影响
- 表的join连接
  - （太）多表连接该质疑设计的正确性了
  - 对于优化器，随着表数量的增加，复杂度将呈指数增长。
  - 编写（太）多表的复杂查询时，多种方式连接的选择失误的几率很高

表的数量

- 还有一个容易忽视的问题，复杂查询和复杂视图

基本的原则是，当是视图返回的数据远多于上级查询所需要的时候，就放弃使用该视图



并发用户数

- 设计的时候需要注意
  - 数据块访问争用（block-access contention）
  - 阻塞(locking)
  - 闩定(latching)
  - 保证读取一致性(read consistency)

- 一般而言，整体吞吐量>个体响应时间

Practice in class 4-2

- 你还有什么方法（自己遇到的，或者查询技术资料、论坛等等资源）能够在数据库应用方面，照顾好用户的情绪？欢迎你的分享。

End

下一讲，我们将开始进入SQL优化的部分