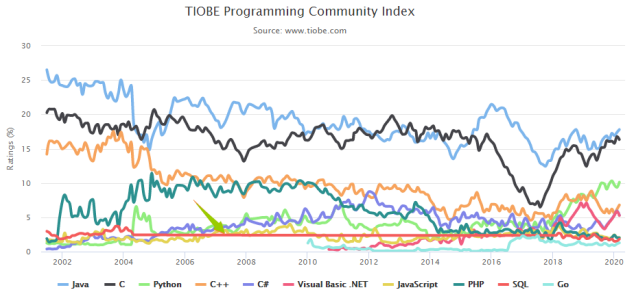
> 此数据库系列，是学习工作中的总结，具体章节系列如下图所示。如果您需要扫盲数据库，突击数据库的面试，那就盘他，盘他！所谓初恋，初次见面，下凡数据库基础。请多多关照！

童鞋别被脱裤系列

1 了解sql----长生不老，异常稳定

查看近几年的TIOBE发现了，一直在前十,可见是个老且管用的东西。



来自网络

了解几个术语：

- DDL(Data Definition Language)，定义数据库对象。
- DML:(DATA Manipulation Language)，用来操作和数据库相关记录。
- DCL:(Data Cnotrol Language)定义访问权限和安全级别。
- DQL(Data Query Language)。查询记录。

sql大小写规范

- 表名、表别名、字段名、字段别名等可以小写
- SQL保留字，函数名，绑定变量大写

> SELECT name,age FROM student WHERE id="1";

2 为啥要存储数据

> 我们大部分的系统都会考虑到数据的存储，那么如何更有效地保存好数据，做

> 好数据备份。当我们拥有了数据，可以进行数据挖掘，大数据分析，舆情预测，自然语言处理等等一系列的操作，可见存储数据是多么的重要。这一篇文章对数据库基础知识的扫盲，下一节将是数据存储，数据备份等。另外我需要知道硬盘的处理速度比cpu,内存，网卡都会慢。

3 DBMS是什么

- DB、DBS和DBMS区别

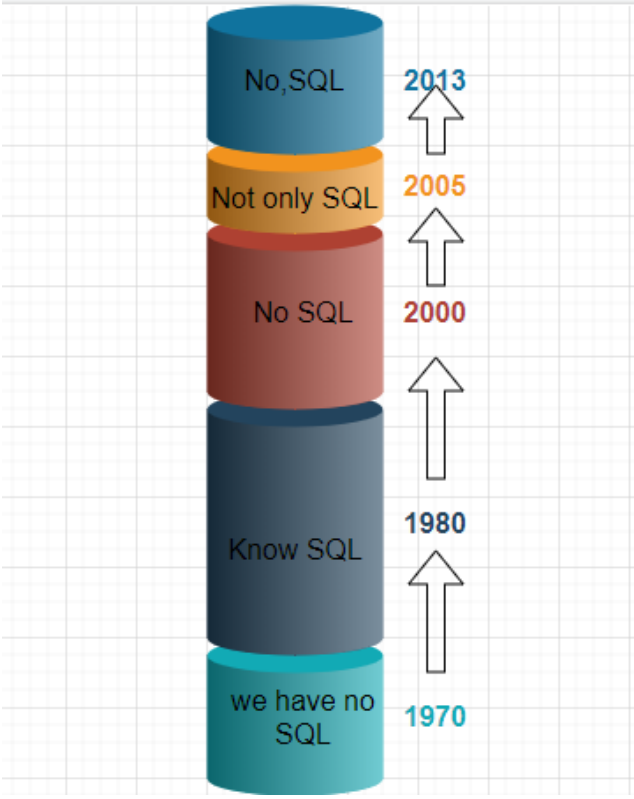
	全称	功能
DBMS	DataBase Management System	可对多个数据库进行管理
DB	DataBase	存储树的集合，理解为多个数据表
DBS	DataBase system	老大哥。包含了数据库管理系统和数据库管理人员DBA

4 当前主流的DBMS有哪些

>

5 sql与Nosql

Nosql的timestamp：



1970：Nosql=we have no sql

1980：Nosql=know sql

2000:Nosql=No SQL

2005:Nosql=not only sql

2013:Nosql=No,SQL

键值数据库

> 通过key-value方式存储，key为唯一表示，优点，查询快，缺点是无法像关系型数据库一样使用条件过滤，这样可能导致遍历所有的键，消耗大量的计算。所以经常用来作为缓存，比如redis。

文档数据库

> 管理文档，一个文档相当于一条记录，MongoDB。

搜索引擎

> 虽然关系型数据库常常通过索引的方式提高检索效率(不一定)，但是对于全文检索却比较低。搜索引擎的优势比如Elasticsearch、Splunk和Solr采用全文搜索，核心原理为倒排索引

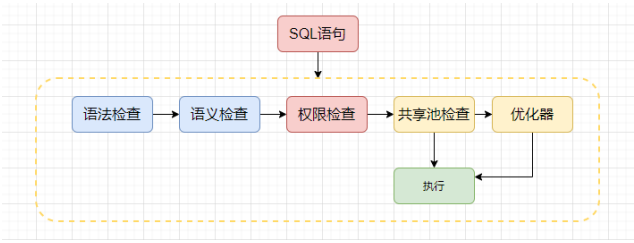
列式数据库

> 相对于行式数据库，将数据按照列存储，这样可以大量降低系统的IO(因为相邻的数据类型一样，方便压缩，自然就会降低IO)，适合分布式文件系统，比如

图数据库

> 典型的就是网络中的人与人的关系，节点和边关系。

6 Oracle中的sql如何执行的

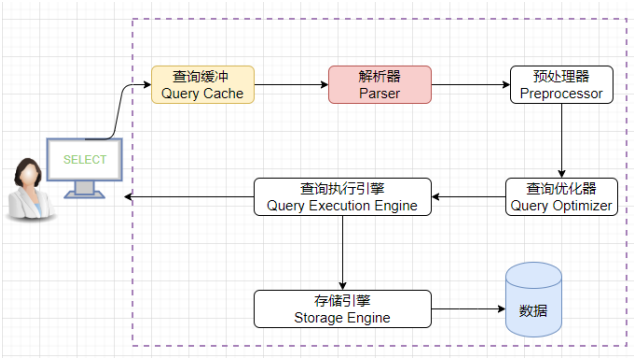


查询执行流程

- 语法检查：检查SQL拼写是否正确。
- 语义检查：检查SQL访问对象是否存在。
- 权限检查：检查用户是否有访问权限。
- 共享池检查：包含了库缓存、数据字典缓冲区等。主要用来缓冲执行计划或者表、视图等对象。

- 优化器：进行硬解析，决定创建解析树和生成执行计划应该怎么做
- 执行器：有了优化器，那么在执行器思考如何被执行

7 mysql中sql如何执行的



查询执行流程

- 查询缓存

> 首先注意，mysql8.0之后已经放弃了这个功能(因为如果数据更新，缓存会情况，如果为动态数据经常更新，这样反而增加SQL查询时间)。缓存通常的理解是一个中间层，如果在中间层存在查询语句就直接返回，如果没有则给解析器处理。

- 解析层

> 主要进行语法分析和语义分析。

- 优化器

> 确定SQL语句的执行路径。是根据全表检索还是根据索引。

- 执行器

> 进行权限检查。

那么mysql和oracle两者执行情况有啥不一样呢

> MySql具有多种存储引擎且可以自定义存储引擎，那么有哪些存储引擎，优点缺点是啥？

	描述
InnoDB	Mysql5.5以后默认存储引擎，支持事务，行级锁，外键约束
MyISAM	Mysql5.5以前为默认存储引擎，不支持事务和外键，最大特点速度快，占资源少
Memory存储引擎	使用系统内存为存储介质，更快的响应速度。
NDB存储引擎	用于Mysql Cluster分布式集群环境
Archive存储引擎	压缩机制的特点便于文件的归档，常用来做仓库

8 如何查看一条sql的资源使用情况

- prifiling是否开启

> mysql> select @@profiling;如果为0代表关闭，设置为1表示打开。

- 执行sql查询语句

> mysql> select *from student;

- 查看当前所有profiles

> show profiles;查看当前会话的profiles

- 查看执行时间show profile

9 DDL

- 创建数据库

> #创建公众号原创作者数据库
>
> CREATE DATABASE WeChat_Official_Account_Author;

- 创建表结构(注意语句最后；结束，最后一个字段定义结束没有逗号)

> #创建表 作者名(author_id,author_name且id为递增)
>
> CREATE TABLE authors_name(author_id int(15) NOT NULL AUTO_INCREMENT)，
>
> author_name varchar(255) NOT NULL);

- 添加字段

> ALTER TABLE authors_name ADD(age int(12));

- 修改字段名

> ALTER TABLE authors_name RENAME_COLUME age to author_age;

- 删除字段

> ALTER TABLE author_names DROP COLUME author_age;

10 数据库的常见约束

> 不成规矩，不成方圆

- 主键约束

> 唯一标识一条记录，不重复且不能为空(UNIQUE+NOT NULL)。主键可以使是一个字段或者多个字段的组合，一个数据表主键只能有一个

- 外键约束

> 外键确保表与表之间引用的完整性。外键可以重复也可以为空。

- 唯一性约束

> 字段在表中可以使唯一的。

- NOT NULL约束

> 表明字段不应为空，必须有取值。

- CHECK约束

> 检查特定字段取值范围的有效性

11 常见查询语句

- 查询姓名列

> #查询单列
>
> SELECT author_name FROM authors_name;
>
> #查询所有列
>
> SELECT * FROM authors_name;

- 去除重复行

> SELECT DISTICT age FROM authors_name;

- 查询排序

> #ASC 递增排序，DESC递减排序
>
> SELECT author_name FROM authors_names ORDER BY age DESC;

- 约束返回结果的个数

> #返回5条数据 LIMIT需要放在最后
>
> SELECT author_name FROM
> authors_names ORDER BY age DESC
> LIMIT 5;

- SELECT语句中关键字顺序

> 关键字的顺序：
> SELECT.....FROM...WHERE...GROUP BY

- SELECT语句的执行顺序

> FROM>WHERE>GROUP BY
> HAVING>SELECT

- 那么一句sql的执行原理是什么？请看下图。

12 sql运算符

- 比较运算符(不同的DBMS支持的运算符可能不同)

运算符	含义
>	大于
>=	大于等于
<	小于
<=	小于等于
=	等于
！=或<>	不等于
IS NULL	是否为空
IS NOT NULL	是否不为空
IN	判断一个值是 IN 列表中的任意一个值
NOT IN	判断一个值不是 IN 列表中的任意一个值
LIKE	通配符匹配
BETWEEN AND	判断一个值是否在两值之间

比较运算符

- 逻辑运算符

操作符	含义
AND	逻辑并
OR	逻辑或
NOT	逻辑否

逻辑运算符

> 当WHERE字句中同时出现OR和AND的时候，AND执行优先级会更高。一般来说()优先级最高，其次是AND,然后是OR。

- 通配符过滤

> 通配符是对文本类型进行模糊哈讯，但是通常是全表扫描，所以效率很低。只有当LIKE后面没有通配符，并对字段进行索引的时候不会进行全表扫描。匹配一部分特殊字符。"LIKE"操作符。

- 通配符匹配之任意字符串出现的任意次数(%)
- 通配符匹配之耽搁字符(_)

13 常见sql函数

> 提供函数，类似接口，更方便快速的得出想要的结果。

	描述	例子
ABS()	取绝对值	SELECT ABS(-5)---5

	描述	例子
MOD()	取余	SELECT MOD(101,3)---2
ROUND()	四舍五入为指定的小数位数，如果两个参数，分别为字段名称和小数位数	SELECT ROUND(38.29,1)--38.3
LENGTH()	计算字段长度。一个汉字三个字符。	SELECT LENGTH('小蓝')--6
UPPER()	字符转大写	SELECT LOWER('qwe')--QWE
LOWER()	字符转小写	SELECT LOWER('QWE')--qwe
REPLACE()	替换函数	SELECT REPLACE('QWE123D','QWE',789)--789123D
SUBSTRING()	截取字符串	SELECT SUBSTRING('QWE123',1,3)--
CHAR_LENGTH()	计算机字段的长度，汉字，数字都算一个字符	SELECT CHAR_LENGTH('小蓝')--2
CONCAT()	连接字符串	SELECT CONCAT('XIAOLAN',789)---XIAOLAN789

	描述	例子
DATA()	返回时间的日期	SELECT DATA('2020-03-13 11:30:20')--2020-03-13
YEAR()/MONTH()/DAY()	返回时间的年份/月份/天数	SELECT YEAR(NOW())--2020
HOUR()	返回时间的小时	SELECT hour('12:13:14')--12
MINUTE()	返回时间的分钟	SELECT MINUTE('12:13:14')--13
SECOND()	返回时间的秒部	SELECT SECOND('12:13:14')--14
CURRENT_DATE()	系统当前日期	SELECT CURRENT_DATE('2020-03-13 11:30:20')--2020-03-13
CURRENT_TIME()	系统当前时间，没有具体日期	SELECT CURRENT_TIME('2020-03-13 11:30:20')--11:30:20

	描述	例子
CURRENT_TIMESTAMP()	日期+时间	SELECT CURRENT_TIMESTAMP--2020-03-13 11:30:20

14 聚集函数

	描述	例子
COUNT()	总行数,不管某个字段是否为NULL	SELECT COUNT(*) FROM authors_name WHERE age>25
MAX()	最大值	SELECT MAX(Age) FROM authors_name
MIN()	最小值	SELECT MIN(Age) FROM authors_name
SUM()	求和	SELECT SUM(Age) FROM authors_name
AVG()	平均值	SELECT AVG(Age) FROM authors_name

- 数据分组

> 使用GROUP BY字句进行数据分组。

- HAVING过滤分组和WHERE的区别

> WHERE 是用于数据行，而 HAVING 则作用于分组。如果分组完以后需要排序，就在其后增加ORDER BY完成

17 视图

- 什么是视图

> 视图可以理解为一个中间表(结果集)，咋们叫虚拟表，它主要把我们经常查询的结果存放于中，从而提升使用的效率。本身不具有数据。如下图所示。

- 为什么使用视图

:baby_bottle:重用SQL语句

:baby_bottle:使用表的一部分而不是整个表

:baby_bottle:更改数据格式和表示。

:baby_bottle:通过授予表的特定访问权限来保护数据，。。

- 如何使用视图

:v:

- 使用视图过滤不想想要的数据
- 更新视图

18 事务处理

> 要么完全执行，要么不执行。

- A(Atomicity)原子性。不可分割，进行数据处理的基本单位。
- C(Consistency)。在进行事务操作以后，会从一致的状态变为另一种一致的状态。即使事务回滚也不能被破坏。
- I(Isolation)隔离性。事务的独立性。一个事务在提交之前，对其他的事务不可见。
- D(Durability)。通过事务日志保证。即使系统崩溃，通过数据库日志的更新让系统恢复到最后一次成功的更新状态。

19 事务隔离

> 我们知道当在高并发的情况下，这个时候需要较高的吞吐量，那么采取方式之一就是将原来的串行操作变化为并行。这个时候可以通过降低数据库的隔离标准，来换取事务的并发能力。

讲述相关内容之前，我们先定义一个表如下。

ID	Age	Name
1	18	小蓝
2	19	小林
3	20	小旋

- 脏读

小蓝今天想去看看数据库内容，并想把朋友小地增加到数据库中，于是操作如下：

```
SQL> BEGINT:
SQL> INSERT INTO authors value(4,20,
```

此时小蓝还没有提交这个事务，小林去访问了这个表(小林去年买了个表，哈哈哈嗝)，于是

```
SQL>SELECT * FROM authors;
```

然后得到这个结果:

ID	Age	Name
1	18	小蓝
2	19	小林
3	22	小旋
4	20	小地

结论：小蓝还没有提交事务，小林访问却看到了增加的小地，这就是脏读。

- 不可重复读

小蓝听说小地也在表里，然后想看看是为何人如此牛掰，几岁了？

```
SQL> SELECT Age FROM authors where N
```

结果如下：

Age
20

我的天，这么年轻？小蓝试图用个事务去修改其年龄

```
SQL>BEGIN;
SQL>UPDATE authors SET Age = 25 wher
```

此时小蓝去查询下修改是否成功

```
SQL>SELECT Age FROM authors where Na
```

结果如下：

Age
25

牛掰，修改成功？那么问题来了，小蓝虽然修改了，但是并没有提交呀，这就是不可重复读，两次查询出现了不同的结果。

- 幻读

今天小旋过来想看看，表里都有哪些小伙伴。

```
SQL> SELECT *FROM authors;
```

结果如下：

ID	Age	Name
1	18	小蓝
2	19	小林
3	22	小旋
4	20	小地

这个时候小林遇到个小妹妹，发现其文采还不错，开启个事务将其放入表中。

```
SQL> BEGIN;
SQL> INSERT INTO authors values(5,'2
```

小林记性太好了，于是还想看看到底有哪些人，

```
SQL> SELECT * FROM authors;
```

结果如下

ID	Age	Name
1	18	小蓝
2	19	小林
3	22	小旋
4	20	小地
5	21	小魏

啊！小林惊呆了，怎么多了个妹妹！！！这就是幻读!

- 隔离级别

	脏读	不可重复读	幻读
读未提交	允许	允许	允许
读已提交	禁止	允许	允许
可重复读	禁止	禁止	允许
可串行化	禁止	禁止	禁止

总结下：

读未提交:

> 允许脏读，也就是可能读取到其他会话中未提交事务修改的数据

读已提交

> 只能读取到已经提交的数据。Oracle等多数数据库默认都是该级别 (不重复读)

可重复读

> 可重复读。在同一个事务内的查询都是事务开始时刻一致的，InnoDB默认级别。在SQL标准中，该隔离级别消除了不可重复读，但是还存在幻读

串行读：

> 全串行化的读，每次读都需要获得表级共享锁，读写相互都会阻塞.

20 python如何操作oracle

```
import cx_Oracle

#建立和数据库系统的连接
conn = cx_Oracle.connect('用户名/密码@oracleserver的ip地址/数据库名字')
#获取操作游标
cursor = conn.cursor()
#执行SQL,创建一个表
cursor.execute("""create table authors(id number, name varchar2(50),password varchar(50),primary
key(id))""")
#关闭连接，释放资源
cursor.close()
#执行完成，打印提示信息
print 'Completed!'
```

简单操作

- 插入数据

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#在Oracle插入数据

import cx_Oracle

conn = cx_Oracle.connect('用户名/密码@oracleserver的ip地址/数据库名字')
cursor = conn.cursor()

#插入一条记录
cursor.execute("""insert into authors values(1,'admin','password')""");

#再插入一条数据
param={'id':2,'n':'admin','p':'password'}
cursor.execute('insert into authors values(:id,:n,:p)',param);

#一次插入多条数据,参数为字典列表形式
param=[{'id':3,'n':'admin','p':'password'},{'id':4,'n':'admin','p':'password'},
{'id':5,'n':'admin','p':'password'}];
cursor.executemany('insert into authors values(:id,:n,:p)',param);

#再一次插入多条数据
param=[];
#生成5条插入数据，参数为元组列表形式
for i in range(6,11):   # [6,7,8,9,10]
    param.append((i,'user'+str(i),'password'+str(i)))
#插入数据
cursor.executemany('insert into authors values(:1,:2,:3)',param);

cursor.close();
#提交更改
conn.commit();
conn.close();
```

插入数据

- 查询数据

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-

#在Oracle中查询数据
import cx_Oracle

conn = cx_Oracle.connect('用户名/密码@oracleserver的ip地址/数据库名字')
cursor = conn.cursor()

#执行查询 语句
cursor.execute("""select * from tb_user""")

#获取一条记录
one = cursor.fetchone()
print '1: id:%s,name:%s,password:%s'%one;

#获取两条记录!!!注意游标已经到了第二条
two = cursor.fetchmany(2)
print '2 and 3:',two[0],two[1]

#获取其余记录!!!注意游标已经到了第四条
three = cursor.fetchall();
for row in three:
    print row   #打印所有结果

print '条件查询'
cursor.prepare("""用户名/密码@oracleserver的ip地址/数据库名字""")
cursor.execute(None,{'id':5})
for row in cursor:  #相当于fetchall()
    print row

cursor.close();
conn.close();
```

查询数据

21 初探调优

为什么调优，无非就是希望响应更快，吞吐量更大，用户体验更好。

> 那么怎么获得反馈

- 用户

> 他们是直接体验者，来的直接。

- 日志汇聚分析,服务器监控，数据库内部监控

> 通过性能工具进行查看，想起一张图送给大家。

这里是性能指标图

> 一般从哪几个方面着手数据库调优，总之没有最好，只有更合适。

- 前期DBMS的调研，选择合适业务的DBMS

> 比如需要有事务处理能力，可以选择mysql的InnoDB。如果采用如果考虑大幅度的降低系统IO，那么可以考虑Nosql中的列式数据库，之前说过列式存储方便使用压缩，但是不适合频繁的增删改。

- 选择合适的缓存比如redis

> 将经常使用的数据放入缓存中(内存),提升查询效率。

- 库级别的优化

> 主从架构优化读写策略，具体方法请关注系列篇第二节。

> 好了，上面的基础部分学习应该差不多了，那么数据库相关的优化，主从架构，读写分离，数据库的分片等都是怎么样的呢？尽请期待后续学习分享，一起成长！

参考资料

《mysql必知必会》

《mysql45讲》

https://blog.csdn.net/gengkui9897/article/details/89294936

《高性能mysql》