

第一讲：关系型数据库管理系统绪论

知春路遇上八里桥

<2024-05-06 Mon>



1 数据库简介

2 关系型数据库

3 SQL - 结构化查询语言



1

数据库简介



RDMS 是什么

- ① RDMS (Relational Database Management System) 是关系数据库管理系统
 - ▶ MySQL (开源), 海豚具有很高的智商
 - ▶ Postgresql (开源), 大象有着惊人的记忆力
 - ▶ Oracle (老牌商业化产品, 闭源)
 - ▶ ...
- ② RDMS 广泛运用于互联网等行业基础组件, 它包含以下基本功能:
 - ▶ 管理用户的数据集
 - ▶ 是世界上最常用的软件



课程目标

- ① 本课程是介绍 RDMS 的设计和实现，不是介绍如何使用 RDMS
 - ▶ 数据库的概念、设计和实现
 - ▶ 以 MySQL 源代码实现为例进行讲解
- ② 适合以下人群
 - ▶ 有一定业务系统开发经验，每天 CURD 吐了的同学
 - ▶ 对关系型数据库内核设计感兴趣的同学
 - ▶ 在技术面试中经常被问到 MySQL 相关问题，想要深入了解源码的同学
 - ▶ 经常被问为什么 xxx 实现不了，希望从数据库底层原理怼回去的同学
- ③ 本课程需要掌握以下基础技能
 - ▶ 对 SQL 语言有基础了解
 - ▶ C/C++ 的编程经验
 - ▶ 大学课程中选修过《数据库概念》类似的课程



课程计划

- ① 授课时间 <2024-05-06 Mon> 至 <2024-09-01 Sun>
 - ▶ 先暂定四个月
- ② 课程准备时间 <2024-03-26 Tue>
 - ▶ 以最新的 MySQL 8.0.x => 即 8.0.37 源码作为依据进行分析
- ③ 涉及到以下相关话题：
 - ▶ 服务端 (mysql-server)
 - ① 启动流程
 - ② C/S 通信模型
 - ③ 解析器 Parser vs 优化器 Optimizer
 - ④ 底层存储：表空间和内存模型
 - ▶ 存储引擎 (InnoDB)
 - ① MVCC 和事务
 - ② 内存管理、磁盘管理
 - ③ 并发和锁
 - ▶ MySQL 周边
 - ① binlog, gdb, mtr 等



2

关系型数据库



关系型数据库起源

1970 年，IBM 研究员 E.F.Codd 博士在刊物 Communication of the ACM 上发表了一篇名为“A Relational Model of Data for Large Shared Data Banks”的论文，提出了关系模型的概念，奠定了关系模型的理论基础。

Home > Magazines > Communications of the ACM > Vol. 13, No. 6 > A relational model of data for large shared data banks

ARTICLE FREE ACCESS



A relational model of data for large shared data banks

Author: E. F. Codd [Authors Info & Claims](#)

Communications of the ACM, Volume 13, Issue 6 • pp 377–387 • <https://doi.org/10.1145/362384.362685>

Published: 01 June 1970 [Publication History](#)



5,687 48,496



Abstract

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in

Communications of
the ACM

Volume 13, Issue 6

[← Previous](#) [Next →](#)

Abstract

References



数据库中的表

关系数据库一般处理的是二维表，例如：

- 员工表 employees(emp_no, birth_date, first_name, ...)

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28

- 部门表 departments(dept_no, dept_name)

dept_no	dept_name
d009	Customer Service
d005	Development
d002	Finance



数据库存储形态

- ❶ 表有多种存储，早期直接使用裸文件存储：
 - ▶ CSV/TXT（纯文本文件）
 - ▶ Excel 表格（二进制格式）
 - ▶ 由数据库软件统一管理
- ❷ 在数据库系统中表可以通过 SQL 语言查询出来，查询返回是结构集

```
mysql> select * from employees limit 3;
```

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28

```
3 rows in set (0.00 sec)
```



读取文件中的数据

- 早期数据存储使用文本文件或格式化文件
- 直接通过代码操作数据的读取/更新/删除
- 下面是使用 Python 读取 csv 的例子

```
1 import csv
2
3 # 打开文件
4 with open('data.csv', 'r') as f:
5     # 创建一个 csv.reader 对象
6     reader = csv.reader(f)
7
8     # 遍历每一行
9     for row in reader:
10         # 读取每一行数据
11         print(row)
```



模型 vs 模式

- ① 模型 (model), 也称数据模型 (data model), 是现实世界特征的模拟和抽象
 - ▶ 数据结构: 常见结构有层次结构、网状结构、关系结构
 - ▶ 数据操作: 数据操作是对系统动态特性的描述
 - ① 指对数据库中各种对象的实例运行执行的操作的集合
 - ② 数据库主要有检索和更新两大类操作
 - ③ 数据模型必须定义这些操作的确切含义、操作符号、操作规则
 - ▶ 完整性约束: 数据的约束条件是一组完整性规则的集合
- ② 模式 (schema), 也称逻辑模式 (physical schema), 是模型对于的数据结构
 - ▶ 是数据库中全体数据的逻辑结构和特征的描述,
 - ▶ 是所有用户的公共数据视图



更多模型例子

- Relational (本课程关注重点)
- Key/Value
- Graph
- Document / Object
- Wide-Column / Column-family
- Array / Matrix / Vectors
- Hierarchical
- Network
- Multi-Value



关系模型

- ① 关系模型 (Relational Model) 包含三要素, 参考之前模型的说明
 - ▶ 数据结构 (Structure), 数据操作 (Manipulation), 完整性 (Integrity)
- ② 关系是一个无序的集合
 - ▶ n 元关系包含 n 个属性
 - ▶ 例如: departments(dept_no, dept_name) 是 2 元关系
- ③ 元组 (tuple) 是一系列属性值, 例如: dept_no, dept_name
 - ▶ 属性值是原子的
 - ▶ NULL 表示空值, 每个属性都应包含空值
- ④ 主键 (Primary Key) 标记每个元组:
 - ▶ 通过 SEQUENCE (PG/ORACLE) 或 AUTO_INCREMENT (MySQL) 实现
- ⑤ 外键 (Foreign Key) 通过一个属性映射到其他关系的元组
- ⑥ DML (数据操作语言) 表示对关系数据的存取
 - ▶ 可以使用关系代数来描述, 稍后介绍



关系代数

❶ 关系代数是一种 过程化 查询语言

- ▶ 它包含运算的集合，运算以一个或两个关系为输入，产生新的关系作为结果
- ▶ n 元关系的代数表示为： $R(a_1, a_2, \dots, a_n)$

❷ 关系代数基本运算¹包括，通过链式调用可以完成对数据的操作

运算符	语法	英文名	中文名
σ	$\sigma_{\text{predicate}}(R)$	Selection	选择
Π	$\Pi_{a_1, a_2, \dots, a_n}(R)$	Projection	投影
\cup	$R_1 \cup R_2$	Union	并
\cap	$R_1 \cap R_2$	Intersection	交
$-$	$R_1 - R_2$	Difference	集合差
\times	$R_1 \times R_2$	Cartesian Product	笛卡尔积
\bowtie	$R_1 \bowtie R_2$	Natural Join	自然链接

¹关系代数的表示法的含义后面通过 SQL 来说明

关系代数 vs SQL ²

假设存在关系： $R(a, b, c)$, $S(a, b, x, y)$ 和 $T(a, b, c)$, 则：

- ① 选择运算： $\sigma_{a>3}(R)$

```
select * from R where a > 3;
```

- ② 投影运算： $\Pi_{a,b}(R)$

```
select a, b from R;
```

- ③ 笛卡尔积 (R 和 S 所有可能的有序对)： $R \times S$, 结果包含属性 a, b, c, a', b', x, y

```
select * from R, S;  
select * from R cross join S;
```

- ④ 自然连接 (相同属性相等后连接)： $R \bowtie S$, 结果包含属性 a, b, c, x, y

```
select * from R natural join S;  
select * from R join S using (a, b);
```

²注意 SQL 关键字不区分大小写

关系代数 vs SQL (续)

假设存在关系： $R(a, b, c)$, $S(a, b, x, y)$ 和 $T(a, b, c)$, 则:

① 选择后投影： $\Pi_{a,b}(\sigma_{c=1}(R))$

```
select a, b from (select * from R where c = 1);
```

-- 简化版

```
select a, b from R where c = 1;
```

② 交集和投影： $\Pi_a(R \cap T)$

```
select a from  
(select * from R intersect select * from T);
```

③ 选择和并集： $(\Pi_{a,b}(\sigma_{c=1}(R))) \cup (\Pi_{a,b}(\sigma_{x>2}(S)))$

```
select a, b from R where c = 1  
union all  
select a, b from S where x > 2;
```



计算次序

假设存在关系： $R(a, b, c)$ 和 $S(a, b, x, y)$ ，则：

- ① 情形一： $\sigma_{a=3}(R \bowtie S)$

```
select * from
  (select * from R natural join S) t
  where t.a = 3;
```

- ② 情形二： $R \bowtie \sigma_{a=3}(S)$

```
select * from
  R natural join (select * from S where a = 3);
```

- ③ 上述两种情形结果相同，但是执行效率存在差异



3

SQL - 结构化查询语言



SQL 基础

- ① SQL 全称 Structured Query Language, 有时也称为 Query
- ② Data Manipulation Language (DML)
 - ▶ insert/select/delete/update
- ③ Data Definition Language (DDL)
 - ▶ create table/drop table
- ④ Data Control Language (DCL)
 - ▶ create user ... identified by ...
 - ▶ grant
- ⑤ 其他功能:
 - ▶ 查看定义 (Definition)
 - ▶ 数据完整性操作 (Integrity)
 - ▶ 事务 (Transactions)



① 建表语句：设置表中列名，类型等

```
1  create table `employees` (  
2      -- 定义字段  
3      `emp_no` int not null,           -- 整数类型  
4      `birth_date` date not null,      -- 日期类型  
5      `first_name` varchar(14) not null, -- Varchar 类型  
6      `last_name` varchar(16) not null, --  
7      `gender` enum('M','F') not null, -- 枚举类型  
8      `hire_date` date not null,      --  
9      primary key (`emp_no`)          -- 设置主键  
10 );
```

② 删表语句：直接删除表中的所有数据

```
1  drop table employees;                -- 删除表
```



DML - 查询

① 查询语句，包含过滤、分组功能

```
1 select vend_id, count(*) as num_prod
2   from products
3  where prod_price >= 10
4  group by vend_id
5 having count(*) >= 2;
```

```
-- 查询字段
-- 指定数据表
-- 过滤条件
-- 分组字段
-- 分组条件
```

② 子查询，设置别名

```
1 select cust_name, cust_state,
2       (select count(*)
3        from orders
4        where orders.cust_id = customers.cust_id)
5       as num_orders
6   from customers
7  order by cust_name;
```

```
-- 查询字段
-- 子查询
--
--
-- 设置别名
--
--
```



DML - 更新

① 插入数据

```
1  insert into
2      employees(emp_no, first_name, gender, hire_date) -- 插入的列
3      value ('12345', 'Jack', 'M', '1999-01-23');      -- 插入的值
```

② 更新数据

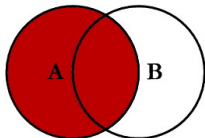
```
1  update employees -- 需要更新的表
2      set hire_date = '2003-10-10' -- 需要更新的字段
3      where emp_no = '12345';      -- 更新的过滤条件
```

③ 删除数据

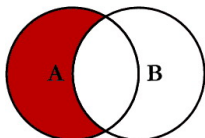
```
1  delete from employees where emp_no = '12345';
```



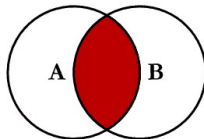
SQL JOINS



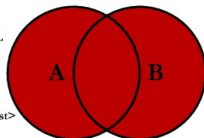
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



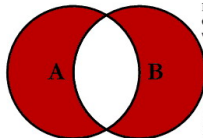
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



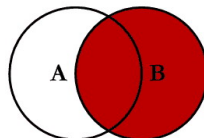
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



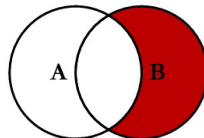
```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

© C.L. Moffatt, 2008



更多特性

- ① 聚合函数: `avg()/min()/max()/sum()/count()`
- ② 数据去重: `select count(distinct first_name) ...`
- ③ 字符串操作符: `where first_name like 'A%'`
- ④ 嵌套查询: `select ... where sid in (select id from student)`
- ⑤ 窗口函数: `select *, row_number() over (order by cid) from enrolled ...`
- ⑥ 公共表达式 (CTE): `with t1 as (select ...) select * from t1`
- ⑦ CTE 实现递归: 打印 1 到 10 的序列

```
1  with recursive t (counter) as (                -- 定义 CTE 及字段别名
2      (select 1)                                -- 递归初始条件
3      union all
4      (select counter + 1 from t where counter < 10) -- 递归逻辑
5  ) select * from t;
```



结束

