数据库及 R 的数据库操作

杨琬妮

2017 年 4 月 4 日

0 为什么使用数据库

在日常学习和工作生活中，我们或多或少需要进行数据管理，数据管理包括对数据进行收集、组织、编码、储存、检索和维护。

在长期的使用过程中，人们发现传统的文件系统有以下几个问题：

- **数据冗余和不一致**：多种文件格式，不同文件中的信息重复
- **难以访问数据**：需要编写一个新程序来执行每个新任务
- **数据隔离**：多种文件和格式
- **完整性问题**：完整性约束（如，账户余额 >0）成为程序代码的一部分
- **难以添加新的约束条件或对已有的限制条件进行修改**

以上提到的这些问题都可以用数据库来解决！

数据库系统有以下优势：

数据和程序是独立的

我们可以独立使用所需的数据。因为程序和数据是独立的，所以当不相关数据的类型被添加到数据库或从数据库中删除时，或者当物理存储更改时，程序不必被修改

数据是集成的

因为数据被集成到单个数据库中，复杂的请求可以比数据位于单独的非集成文件中快得多。

数据的重复减少了

随着数据的集成，数据重复的机会大大减少，且数据均以最新形式呈现

数据的有效性

使用数据库管理系统（DBMS）管理数据将使数据更加有效。DBMS 提供不同的检查以确保数据的有效性

提高数据的安全性

多途径访问

数据库软件使我们能够以多种方式访问数据

基于数据库诸多优点，企业更多地倾向于使用数据库进行数据管理，一些应用系统也是基于数据库开发的，数据库在日常工作中十分实用。

下面，我将对常用的数据库模型 ——关系型数据库做一个简单地介绍。接着简单介绍 SQL(Structured Query Language) 的相关内容，最后和大家一起用 R 连接数据库，并对数据库进行相关操作。

1 关系型数据库简介

关系模型是应用较为为广泛的数据库模型，是数据存储的传统标准。我们常见的一些数据库系统，Access、SQLServer、MySQL、Oracle 都是关系型数据库系统。

数据模型包括：

- **数据结构**
- **数据操作**
- **数据上的约束**

1.1 关系模型（Relational Model）

几个基本概念：

关系（Relation）：一个关系就是一张二维表

属性（Attribute）：关系的列命名为属性

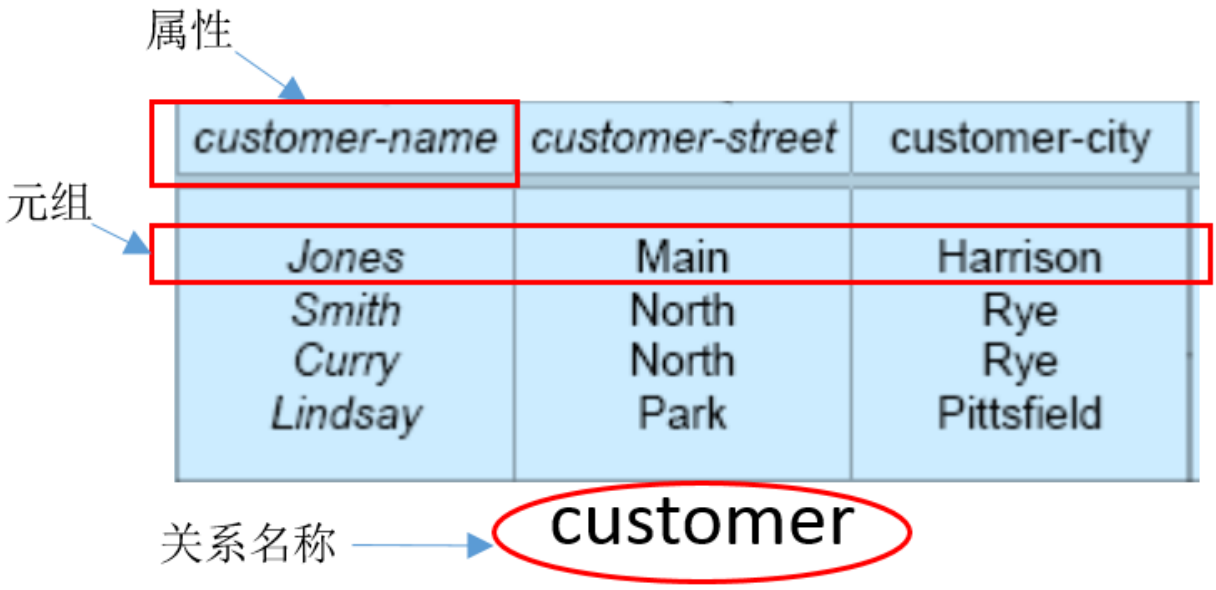
元组（Tuple）：关系中除含有属性名所在行以外的其他行成为元组。每个元组均有一个**分量**（component）对应于关系的每个属性

域（Domain）

模式（Schema）：关系名及其属性结合的组合成为这个关系的模式

键（Key）: 键由关系的一组属性集组成，通过定义键可以保证关系实例上任何两个元组的值在定义键的属性集上取值不同。任意两个元组不能在键上的所有属性上具有完全相同的值，除非其中有一个是空值（NULL）

主键（Primary Key）: 不能有完全相同的值，且不能有 NULL



1.2 数据上约束

什么是约束？就是对可能存入数据库的数据进行限制。

约束的类型

- 域约束（Domain Constraints）
- 键约束（Key Constraints）
- 完整性约束（Integrity constraints）

-实体完整性（Entity Integrity Constraint）

-参照完整性（Referential Integrity Constraint）

-语义完整性（Semantic Integrity Constraint）

域约束

对输入数据的类型、长度、取值范围等的约束

键约束

主键（Primary key）

外键（Foreign key）

简单理解：外键就是另一个（也可以是同一个）关系的主键

主键和外键的存在，使我们可以建立多表之间的联系，使所设计的数据库大大减少信息的重复（冗余），并有效地保持信息的一致性。

完整性约束

实体完整性：主键的值不能为空

参照完整性：

假设有两个关系之间存在以下联系：关系 R1 引用了关系 R2，即 R1 表中的外键引用的是 R2 表的主键

R1 表中外键的值只能是：（1）R2 中对应主键现有的主键值，或者（2）空值

1.3 数据操作

- 关系代数（Relational algebra）
- 关系演算（Relational calculus）
- 结构化查询语言（Structured Query Language，SQL）

这次我们主要介绍 SQL~

2 SQL

2.1 概述

SQL（Structured Query Language）是一种特殊目的的编程语言，是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统的语言。

SQL 的组成

- 数据定义语言（DDL）：用来定义数据库的逻辑结构，主要包括表、视图、索引
- 数据操纵语言（DML）：包括数据查询和数据更新
- 数据控制语言（DCL）：包括授权、完整性约束、事务开始和结束等控制语句
- 嵌入式 SQL：规定了 SQL 在宿主语言中的使用规则

本次主要介绍数据操纵语言 ~

2.2 SQL 的数据查询

1. 选择查看某些列

Select < 列名 > from < 表名 >;

分号 (;) 代表该 SQL 语句结束。查看多列时，列名用逗号 (,) 隔开，选择查看该表所有列，列名处用 * 号，即：

Select * from < 表名 >;

使用 distinct 可删除重复行，但是代价比较大：

Select distinct < 列名 > from < 表名 >;

要使更改显示结果的列名，可用 as:

Select < 原列名 1> as < 自定义列名 1>,< 原列名 2> as < 自定义列名 2>,... from < 表名 >;

2. 选择查看某些行

Select < 列名 > from < 表名 > where < 表达式 >;

注：< 表达式 > 填写所筛选的相关行所符合的条件, 多个条件限制表达式之间可用 and 联接, 详见示例。

选择表达式可能用到的一些特殊操作符：

（1）通配符：可用于代表任意字符

- _ : 代表任意一个字符
- % ：代表任意多个字符
-
- [^charlist] 或者 [!charlist]: 不在字符列中的任何单一字符

3. 多表之间的并、交、差

如果两表之间的属性兼容（即属性数目相同，且对应属性的域相同），即可对两表做并、交、差操作。

并：< 表名 1> union [all] < 表名 2> ;

交：< 表名 1> intersect [all] < 表名 2> ;

差：< 表名 1> except [all] < 表名 2> ;

注：并、交、差运算会消除重复元组，使用可选 [all] 指令表示不消除重复元组

4. 表的联接（多表查询）

表的联接涉及到数据库对表的几个基本操作：笛卡尔乘积、选择（选行）、投影（选列）

（1）笛卡尔乘积

Select * from < 表 1>,< 表 2> ;

（2）选行

Select * from < 表 1>,< 表 2> where < 表达式 >;

（3）选列

Select < 列名 > from < 表 1>,< 表 2> where < 表达式 >

注：

- 两表联接基本上是通过将两表中的对应字段 (对应的字段通常是主键和外键关系) 分别含有相同值（匹配）的信息联接起来，形成一个信息较为完整的大表，因此，两表联接会删除重复的列，这样的结果较为简洁。??
- 两表可能会有相同的列名，这时引用列名须标明该列属于哪张表，具体格式为：< 表名. 列名 >
- 表的联接操作默认删除对应字段中没有相同值的行??（举例）

- 外链接：保留表中对应字段没有相同值的行，分为左外链接、右外连接、全外链接

5. 外联接

因为自然联接会将许多信息删除，为了避免一些数据损失，我们可以使用外联接，保留那些对应字段没有匹配值的元组。

ANSI 的标准 SQL 不支持外联接，但是许多 DBMS 产品支持:

（1）左外联接

Select * from student , enrollment where student .SID * = enrollment .studentnumber;

（2）右外联接

Select * from student , enrollment where student .SID =* enrollment .studentnumber;

6. 子查询（嵌套查询）

在某查询结果的基础上进行查询（详见示例）

嵌套 SQL 与联接 SQL 的比较

- 嵌套 SQL 的求解层次更分明，执行效率也高于联接 SQL。
- 嵌套 SQL 最后得到的属性只能来源于一张表，联接 SQL 最后得到的属性可以来源于多张表。
- 某些查询只能采用嵌套 SQL，而无法采用联接 SQL。
- 嵌套 SQL 中，子查询语句不能使用 Order By 子句，Order By 子句只能对最终查询结果排序。

7. 统计查询

对数据库数据进行某些运算

- 5 个聚合函数：count（）、sum（）、avg（）、max（）、min（）
- 1 个分组子句：group by 子句

聚合函数常用语法 Count（[distinct | all] * ）统计元组个数 Count（[distinct | all] ）对某一列中的值计数 Sum（[distinct | all] ）求和（列必须是数值型）Avg（[distinct | all] ）计算平均值（列必须是数值型）Max（[distinct | all] ）求最大值 Min（[distinct | all] ）求最小值

注意：在适当的时候使用 distinct

分组子句的使用

【例子】Select major , count（*）from student group by major

Select major , count（*）from student group by major having count（*）>1

- group by: 分组子句，一般跟在 where 子句后，没有 where 子句时，跟在 from 子句后
- having：剔除不符合条件的分组，总是跟在 group 子句后，不可以单独使用

- Where 条件优先于 having，一般先用 Where 子句限定元组，然后分组，最后再用 having 子句限定分组

使用 GROUP BY 子句需注意：

1. 不能 GROUP BY text、image 或 bit 数据类型的列

2.select 列表中的每一列必须出现在 GROUP BY 子句中，除非这列是用于聚合函数或者常量。【举例】

3. 不能 GROUP BY 列的别名。这是说 GROUP BY 字段列表中的所有字段必须是实际存在于 FROM 子句中指定的表中的列

4. 进行分组前可以使用 WHERE 子句消除不满足条件的行。使用 GROUP BY 子句返回的组没有特定的顺序。可以使用 ORDER BY 子句指定想要的排序次序

5.GROUP BY 字段中的 Null 值将被分组，而不会省略。但是，在任何 SQL 聚合函数中都不会计算 Null 值。

2.3 数据更新

1. 插入记录

Insert into < 表名 > values （元组值）

2. 删除记录

delete from < 表名 > where < 条件表达式 >

3. 修改记录

update < 表名 > set < 置换内容 > where < 条件 >

3. R VS 数据库

为什么要用 R 连接数据库呢？

- 编程时可能用的是来自数据库的数据
- 编程软件所提供的数据库接口使让我们能够方便地从编程软件中调取数据库中的数据，而不需要先从数据库中导出数据到外部文件，在将外部文件导入到 R 中
- 直接用编程软件和数据库的接口调取数据，速度快且更新及时！

总之，使用 R 来访问存储在外部数据库中的数据是一种分析大数据集的有效手段，并且能够发挥 SQL 和 R 各自的优势

3.1 R 与数据库连接

R 中有多种面向关系型数据库管理系统（DBMS）的接口，包括 Microsoft SQL Server、Microsoft Access、MySQL、Oracle、PostgreSQL、DB2、Sybase、Teradata 以及 SQLite。其中一些包通过原生的数

据库驱动来提供访问功能，另一些则是通过 ODBC 或 JDBC 来实现访问的。

简单来说，这些驱动能完成下列三件事：

- 同一个数据库建立连接
- 向数据库发送 SQL 语句
- 处理数据库返回的结果

1 ODBC 接口

在 R 中通过 RODBC 包访问一个数据库可能是最常用的方式，这种方式允许 R 连接到任意一种拥有 ODBC 驱动的数据库，其实几乎就是市面上的所有数据库。

第一步是针对你的系统和数据库类型安装和配置合适的 ODBC(Open Database Connectivity，开放数据库连接) 驱动 ——它们并不是 R 的一部分。如果你的机器尚未安装必要的驱动，上网搜索一下应该就可以找到。

针对选择的数据库安装并配置好驱动后，还需要安装 RODBC 包。你可以使用命令 install.packages("RODBC") 来安装它。下面列出 RODBC 中的函数：

函　数	描　述
odbcConnect(*dsn*,uid="",pwd="")	建立一个到ODBC数据库的连接
sqlFetch(*channel*,*sqltable*)	读取ODBC数据库中的某个表到一个数据框中
sqlQuery(*channel*,*query*)	向ODBC数据库提交一个查询并返回结果
sqlSave(*channel*,*mydf*,tablename=*sqtable*,append=FALSE)	将数据框写入或更新（append=TRUE）到ODBC数据库的某个表中
sqlDrop(*channel*,*sqtable*)	删除ODBC数据库中的某个表
close(*channel*)	关闭连接

RODBC 包允许 R 和一个通过 ODBC 连接的 SQL 数据库之间进行双向通信。这意味着我们不仅可以读取数据库中的数据到 R 中，同时也可以使用 R 修改数据库中的内容。

2 DBI 相关包

DBI 包为访问数据库提供了一个通用且一致的客户端接口。构建于这个框架之上的 RJDBC 包提供了通过 JDBC 驱动访问数据库的方案。使用时请确保安装了针对你的系统和数据库的必要 JDBC 驱动。其他有用的、基于 DBI 的包有 RMySQL（操作 MySQL）、ROracle（操作 Oracle）、RPostgreSQL（）、RSQLServer（操作 Microsoft SQL）和 RSQLite（操作 SQLite）等。这些包都为对应的数据库提供了原生的数据库驱动，但可能不是在所有系统上都可用。详情请参阅 CRAN（http://cran.r-project.org　　。

本次课程为了让大家也能跟着练练手，我们使用最简单的数据库，SQLite——轻量级数据库引擎为大家讲解。

在 R 中操作 SQLite 数据库

3 R 语言连接数据库通用步骤

尽管连接不同的数据库需要使用不同的包，但是一些基本的步骤都是通用的，下面我们以连接 SQLite 数据库为例演示如何用 R 连接数据库

- Step1 加载相关的包，操作 SQLite 数据库文件需要加载 RSQLite 包：

```
# 如果还没安装包，需要安装一下 ~
# 安装 RSQLite 包的同时也会安装 DBI 包 ~
install.packages("RSQLite")
library(RSQLite)
```

- Step2 创建数据库连接：

```
con <- dbConnect(SQLite( ), "data/example.sqlite")
```

数据连接是介于用户和系统中间的一层，我们通过它连接到数据库，并通过这个连接实现查询、读取以及更新数据。

- Step3 通过数据库连接操作数据库
- Step4 使用完毕，断开数据库连接

```
dbDisconnect(con)
```

连接不同数据库在某些细节上可能略有差异，大家工作中需要用到具体数据库的时候再去查就可以啦 ~

3.2 DBI 包自带数据库操作函数

DBI 包中提供一些函数，使我们可以不必通过 SQL 就能对数据库进行一些简单的操作，以下介绍一些比较常用的：

```
dbGetInfo(con)# 获取数据库基本信息
dbListTables(con)# 列出数据库中的所有表
dbExistsTable(con, "students")# 检查数据库中是否存在某张表
dbListFields(con, "student")# 显示数据库中某张表的列名（字段）
studentInfo <- dbReadTable(con,"student")# 将整张表读入一个数据框
# 除了以上简单查询，还能对数据进行一些修改 ~
# 温馨提醒：修改数据需谨慎！
dbWriteTable(con,"studentcopy",studentInfo)# 用于向一个数据库写入表格，或者追加一些数据
dbRemoveTable(con,"studentcopy")# 删除数据库中指定表
```

3.3 用 SQL 对关系型数据库进行查询

这部分的关键在 SQL 语句，R 代码其实很简单：

```
dbGetQuery(con,"sqlStatement")#"sqlStatement" 处为具体的 SQL 语句
```

3.4 分块提取查询结果

有时候，我们需要检查的数据量超过了计算机内存容量。显然，不能把所有数据载入内存，所以必须逐块处理。

绝大部分关系型数据支持逐块提取查询数据。在接下来的例子中，我们会用 dbSendQuery( ) 进行查询，而不是用 dbGetQuery( )。然后，我们重复地从查询结果中提取出一块数据（几行记录），直到遍历查询结果。通过这种方式，逐块地处理数据，便不需要用到很大的内存空间。

```
con <- dbConnect(SQLite( ), "data/example2.sqlite")
res <- dbSendQuery(con,
                   "select carat, cut, color, price from diamonds
                   where cut = 'Ideal' and color = 'E' ")
while(!dbHasCompleted(res)){
  chunk <- dbFetch(res, 800)
  cat(nrow(chunk), "records fetched\n")
}
dbClearResult(res)
dbDisconnect(con)
```

实践中，数据库中可能有数十亿条记录。查询结果有可能达到千万条。如果用 dbGetQuery( ) 一次性取出所有查询结果，内存可能吃不消。但是，如果容许分块处理数据来完成任务，那么上述方法不失为一个好的选择。

4 参考文献

- 数据库系统基础教程 (第 3 版)
- Learning R Programing
- R in Action