关系代数与 SQL

续本达

清华大学 工程物理系

2024-07-25 清华

下载课堂练习数据并安装软件

```
wget "http://hep.tsinghua.edu.cn/~orv/pd/dataframe-practice-r1.tar.gz"
tar -xf dataframe-practice-r1.tar.gz
```

- 安装 SQLite3 和查看器，以及 CSV 查看器

```
apt install sqlite3 sqlitebrowser csvkit csvkit-doc
```

本周提要

第一周 一切工作都是差分

第二周 一切计算都是数组

第三周 一切行为都是命令

- 程序运行与文件管理：
  1. 外壳 shell: `bash`
  2. 流水线: `make`（函数式编程、描述性编程）
  3. 字符处理：正则表达式（描述性编程）

第四周 一切数据都是表格

- 学习如何有序组织数据，努力让统计分析和数据洞见更易进行
  - 关系代数，学习新语言 SQL
- 应用意义：直接应用到分布式 SQL 引擎
- 1981 年和 2014 年的两届图灵奖工作

引例：表格 vs 数组

`eid` 指事例编号 event ID，`ch` 指读数通道 `channel`。

eid	ch	wave
0	0	(0, 2, 3)
0	2	(0, 0, 1)
1	2	(3, 2, 0)
1	3	(0, 3, 1)
…		

- `eid` 的取值可达几百万 N_{E}，`ch` 的取值是 0 到 29 共 30 个。
- 表格定义：把坐标 (0, 0)、(0, 2)、(1, 2)、(1, 3) 写在列上
- 长度略小于 $N_{\mathrm{E}} \times 30$ 的表格

ch \ eid	0	1
0	(0, 2, 3)	x
2	(0, 0, 1)	(3, 2, 0)
3	x	(0, 3, 1)

- 如果行与列的标号不连续，则另行开辟 `eid` 和 `ch` 的数组，建立 0, 2, 3 与标号 0, 1, 2 的对应关系。
- 形状为 $(N_{\mathrm{E}}, 30)$ 的二维数组

概念

eid	ch	wave
0	0	(0, 2, 3)
0	2	(0, 0, 1)
1	2	(3, 2, 0)
1	3	(0, 3, 1)
...		

- 一切都是表，可以表达一切数据
  - 未必真实存在，可以是逻辑的表
- 集合运算，交、并、差 $\cap, \cup, \setminus$
  - 集合元素是表格的行。
- 线性运算，笛卡尔积、投影、选取 $\otimes, \Pi, \sigma$
  - 表格扩大，列方向与行方向缩小
- 关系运算，连接 $\bowtie$
  - 非关系代数基本运算，可由 $\otimes\sigma$ 组合得到，极常用
- 拓展运算，GroupBy（分组）$\mathcal{G}$
  - 非关系代表运算，不可由基本运算实现，极常用

以代数理论组织数据的图灵奖

Edgar F. Codd

1981 年图灵奖得主

Relational database: a practical foundation for productivity.

Michael Stonebraker

2014 年图灵奖得主

For fundamental contributions to the concepts and practices underlying modern database systems.

参考书

Hellerstein, Joseph M. and Michael Stonebraker. Readings in Database Systems.

反思：如此简明的理论为何影响深远？

1. 数据组织在物理学家眼里可能微不足道：如何组织不重要，能用就行；
2. “自然”地采用树状结构：一个高能量的粒子产生很多“次数粒子”，次数粒子诱导出光子，一个光子被放大成 1 千万个电子；
3. 争论：什么样的“树”最好？
4. 每个人有自己关注的物理问题，因此争证持续，或者走向分裂；
5. “都是那些个二货坚持用奇怪的数据格式”；
6. 反思时，发现自己在“微不足道”上花了太多精力，非常空虚。

反思：如此简明的理论为何影响深远？

1. 数据组织在物理学家眼里可能微不足道：如何组织不重要，能用就行；
2. “自然”地采用树状结构：一个高能量的粒子产生很多“次数粒子”，次数粒子诱导出光子，一个光子被放大成 1 千万个电子；
3. 争论：什么样的“树”最好？
   - 复分 vs. 能力：出身星系 vs. 天体种类；
   - 复分 vs. 能力：空间 vs. 时间；
4. 每个人有自己关注的物理问题，因此争证持续，或者走向分裂；
5. “都是那些个二货坚持用奇怪的数据格式”；
6. 反思时，发现自己在“微不足道”上花了太多精力，非常空虚。

反思：如此简明的理论为何影响深远？

1. 数据组织在物理学家眼里可能微不足道：如何组织不重要，能用就行；
2. “自然”地采用树状结构：一个高能量的粒子产生很多“次数粒子”，次数粒子诱导出光子，一个光子被放大成 1 千万个电子；
3. 争论：什么样的“树”最好？
   - 身分 vs. 能力：出身星系 vs. 天体种类；
   - 身分 vs. 能力：母粒子 vs. 粒子种类；
4. 每个人有自己关注的物理问题，因此争证持续，或者走向分裂；
5. “都是那些个二货坚持用奇怪的数据格式”；
6. 反思时，发现自己在“微不足道”上花了太多精力，非常空虚。

反思：如此简明的理论为何影响深远？

1. 数据组织在物理学家眼里可能微不足道：如何组织不重要，能用就行；
2. “自然”地采用树状结构：一个高能量的粒子产生很多“次数粒子”，次数粒子诱导出光子，一个光子被放大成 1 千万个电子；
3. 争论：什么样的“树”最好？
   - 身分 vs. 能力：出身星系 vs. 天体种类；
   - 身分 vs. 能力：母粒子 vs. 粒子种类；
4. 每个人有自己关注的物理问题，因此争证持续，或者走向分裂；
5. “都是那些个二货坚持用奇怪的数据格式”；
6. 反思时，发现自己在“微不足道”上花了太多精力，非常空虚。

反思：如此简明的理论为何影响深远？

1. 数据组织在物理学家眼里可能微不足道：如何组织不重要，能用就行；
2. “自然”地采用树状结构：一个高能量的粒子产生很多“次数粒子”，次数粒子诱导出光子，一个光子被放大成 1 千万个电子；
3. 争论：什么样的“树”最好？
   - 身分 vs. 能力：出身星系 vs. 天体种类；
   - 身分 vs. 能力：母粒子 vs. 粒子种类；
4. 每个人有自己关注的物理问题，因此争证持续，或者走向分裂；
5. “都是那些个二货坚持用奇怪的数据格式”；
6. 反思时，发现自己在“微不足道”上花了太多精力，非常空虚。

反思：如此简明的理论为何影响深远？

1. 数据组织在物理学家眼里可能微不足道：如何组织不重要，能用就行；
2. “自然”地采用树状结构：一个高能量的粒子产生很多“次数粒子”，次数粒子诱导出光子，一个光子被放大成 1 千万个电子；
3. 争论：什么样的“树”最好？
   - 身分 vs. 能力：出身星系 vs. 天体种类；
   - 身分 vs. 能力：母粒子 vs. 粒子种类；
4. 每个人有自己关注的物理问题，因此争证持续，或者走向分裂；
5. “都是那些个二货坚持用奇怪的数据格式”；
6. 反思时，发现自己在“微不足道”上花了太多精力，非常空虚。

反思：如此简明的理论为何影响深远？

1. 数据组织在物理学家眼里可能微不足道：如何组织不重要，能用就行；
2. “自然”地采用树状结构：一个高能量的粒子产生很多“次数粒子”，次数粒子诱导出光子，一个光子被放大成 1 千万个电子；
3. 争论：什么样的“树”最好？
   - 身分 vs. 能力：出身星系 vs. 天体种类；
   - 身分 vs. 能力：母粒子 vs. 粒子种类；
4. 每个人有自己关注的物理问题，因此争证持续，或者走向分裂；
5. “都是那些个二货坚持用奇怪的数据格式”；
6. 反思时，发现自己在“微不足道”上花了太多精力，非常空虚。

反思：如此简明的理论为何影响深远？

1. 数据组织在物理学家眼里可能微不足道：如何组织不重要，能用就行；
2. “自然”地采用树状结构：一个高能量的粒子产生很多“次数粒子”，次数粒子诱导出光子，一个光子被放大成 1 千万个电子；
3. 争论：什么样的“树”最好？
   - 身分 vs. 能力：出身星系 vs. 天体种类；
   - 身分 vs. 能力：母粒子 vs. 粒子种类；
4. 每个人有自己关注的物理问题，因此争证持续，或者走向分裂；
5. “都是那些个二货坚持用奇怪的数据格式”；
6. 反思时，发现自己在“微不足道”上花了太多精力，非常空虚。

研究这个问题是为了不再研究这个问题

- 按照简明的理论组织数据，从此之后忘掉它。
- 简明的理论有严格的数学体系；
- 与正则表达式一样，可以有完美的社会分工；
  - 使用者以关系代数的思想和语言描绘出自己要做什么；
  - 研发者以天才的程序技巧实现关系代数的基本运算，优化复合运算。

工业领域的某些反弹

- 2000 年左右，Web 界开始了去“关系数据库”的运动，开发 NoSQL 服务。
  - SQL 是 structured query language，关系数据库的基本语言；
  - 关系数据库是以关系代数为基础的数据管理程序，有严格的数据规范。
- NoSQL 运动的支持者认为，世界丰富多样，不应该受关系代数的限制。
  - Web 2.0 被作为商业概念炒作，从业人员四处革命
  - 关系代数让一切都成为了表格，大大限制了我们的想象力，剥夺了我们自由设计数据格式的权力
  - 于是大家开浪起来，进行了一轮新的探索，在这个过程中创造了一批卓越的分布式的数据系统
  - Google 的 MapReduce 也在此运动中，改变了整个大数据处理的格局。
- 但是，随着时间的推移，这些 NoSQL 的方案逐渐成熟，慢慢发展出关系代数的各种运算
  - 关系代数系统被重新发明一次，与分布式系统结合起来，成为数据时代的基石

工业领域的某些反弹

- 2000 年左右，Web 界开始了去“关系数据库”的运动，开发 NoSQL 服务。
  - SQL 是 structured query language，关系数据库的基本语言；
  - 关系数据库是以关系代数为基础的数据管理程序，有严格的数据规范。
- NoSQL 运动的支持者认为，世界丰富多样，不应该受关系代数的限制。
  - Web 2.0 被作为商业概念炒作，从业人员四处革命
  - 关系代数让一切都成为了表格，大大限制了我们的想象力，剥夺了我们自由设计数据格式的权力
  - 于是大家开浪起来，进行了一轮新的探索，在这个过程中创造了一批卓越的分布式的数据系统
  - Google 的 MapReduce 也在此运动中，改变了整个大数据处理的格局。
- 但是，随着时间的推移，这些 NoSQL 的方案逐渐成熟，慢慢发展出关系代数的各种运算
  - 关系代数系统被重新发明一次，与分布式系统结合起来，成为数据时代的基石

工业领域的某些反弹

- 2000 年左右，Web 界开始了去“关系数据库”的运动，开发 NoSQL 服务。
  - SQL 是 structured query language，关系数据库的基本语言；
  - 关系数据库是以关系代数为基础的数据管理程序，有严格的数据规范。
- NoSQL 运动的支持者认为，世界丰富多样，不应该受关系代数的限制。
  - Web 2.0 被作为商业概念炒作，从业人员四处革命
  - 关系代数让一切都成为了表格，大大限制了我们的想象力，剥夺了我们自由设计数据格式的权力
  - 于是大家开浪起来，进行了一轮新的探索，在这个过程中创造了一批卓越的分布式的数据系统
  - Google 的 MapReduce 也在此运动中，改变了整个大数据处理的格局。
- 但是，随着时间的推移，这些 NoSQL 的方案逐渐成熟，慢慢发展出关系代数的各种运算
  - 关系代数系统被重新发明一次，与分布式系统结合起来，成为数据时代的基石

关系代数的实现

- SQL：是 structured query language 的缩写
  - 关系代数的天然语言，语法简明
  - 描述型的语言，描述把什么样的数据取出来
  - 也可以写数据，但一般是一次写，多次读
- 关系数据库 Relational Database，使用 SQL 语言描述
  - MariaDB (MySQL 的后继)
  - PostgreSQL
  - SQLite
- 统计软件分支，比如 S 语言中的 SAS/SPSS 中的 Dataset 或 DataFrame
  - GNU R 以及对应的 R 语言，是一个 S 语言的后继，其中 DataFrame 是语言的核心
  - Pandas 受 GNU R 的影响，目标是在 Python 的语言环境中实现 DataFrame 及其基本操作
- MapReduce 分布式大数据算法也受到关系代数的影响，一般都会使用关系代数作为平台高级接口
  - Hadoop 生态圈, Spark, etc.

关系代数的实现

- SQL：是 structured query language 的缩写
  - 关系代数的天然语言，语法简明
  - 描述型的语言，描述把什么样的数据取出来
  - 也可以写数据，但一般是一次写，多次读
- 关系数据库 Relational Database，使用 SQL 语言描述
  - MariaDB (MySQL 的后继)
  - PostgreSQL
  - SQLite
- 统计软件分支，比如 S 语言中的 SAS/SPSS 中的 Dataset 或 DataFrame
  - GNU R 以及对应的 R 语言，是一个 S 语言的后继，其中 DataFrame 是语言的核心
  - Pandas 受 GNU R 的影响，目标是在 Python 的语言环境中实现 DataFrame 及其基本操作
- MapReduce 分布式大数据算法也受到关系代数的影响，一般都会使用关系代数作为平台高级接口
  - Hadoop 生态圈, Spark, etc.

关系代数的实现

- SQL：是 structured query language 的缩写
  - 关系代数的天然语言，语法简明
  - 描述型的语言，描述把什么样的数据取出来
  - 也可以写数据，但一般是一次写，多次读
- 关系数据库 Relational Database，使用 SQL 语言描述
  - MariaDB (MySQL 的后继)
  - PostgreSQL
  - SQLite
- 统计软件分支，比如 S 语言中的 SAS/SPSS 中的 Dataset 或 DataFrame
  - GNU R 以及对应的 R 语言，是一个 S 语言的后继，其中 DataFrame 是语言的核心
  - Pandas 受 GNU R 的影响，目标是在 Python 的语言环境中实现 DataFrame 及其基本操作
- MapReduce 分布式大数据算法也受到关系代数的影响，一般都会使用关系代数作为平台高级接口
  - Hadoop 生态圈, Spark, etc.

关系代数的实现

- SQL：是 structured query language 的缩写
  - 关系代数的天然语言，语法简明
  - 描述型的语言，描述把什么样的数据取出来
  - 也可以写数据，但一般是一次写，多次读
- 关系数据库 Relational Database，使用 SQL 语言描述
  - MariaDB (MySQL 的后继)
  - PostgreSQL
  - SQLite
- 统计软件分支，比如 S 语言中的 SAS/SPSS 中的 Dataset 或 DataFrame
  - GNU R 以及对应的 R 语言，是一个 S 语言的后继，其中 DataFrame 是语言的核心
  - Pandas 受 GNU R 的影响，目标是在 Python 的语言环境中实现 DataFrame 及其基本操作
- MapReduce 分布式大数据算法也受到关系代数的影响，一般都会使用关系代数作为平台高级接口
  - Hadoop 生态圈, Spark, etc.

SQL 描述性编程：定义清楚“我要什么”。

`CREATE` 创建表格
`INSERT` 插入行
`SELECT` 取得内容

SQL 关键字的大写约定

- 为了醒目地展现 SQLite 的关键字，我们约定它们全大写。

SQL 基本操作

```
CREATE TABLE A(ID integer, name text);
INSERT INTO A VALUES(1, 'Wang');
INSERT INTO A VALUES(2, 'Li');
```

```
SELECT * FROM A;
```

ID	name
1	Wang
2	Li

Python 的 SQLite 模块

- 把 SQL 完整语句内嵌到函数调用中。

```
import sqlite3

c = sqlite3.connect("ra-python.db")
cur = c.cursor()
cur.execute("CREATE TABLE A(ID integer, name text)")
cur.execute("INSERT INTO A VALUES (1, 'Wang')")
cur.execute("INSERT INTO A VALUES (2, 'Li')")
cur.execute("SELECT * FROM A")
print(cur.fetchall()) # 读取 SQL 返回值
c.close() # 关闭
```

```
[(1, 'Wang'), (2, 'Li')]
```

- 计算机的本质：把一系列指令交给另一个工具执行。

Python 循环减少重复

```python
import sqlite3

c = sqlite3.connect("ra-python.db")
cur = c.cursor()
cur.execute("CREATE TABLE A(ID integer, name text)")
for ID, name in ((1, 'Wang'), (2, 'Li')):
    cur.execute(f"INSERT INTO A VALUES ({ID}, {name})")
cur.execute("SELECT * FROM A")
print(cur.fetchall()) # 读取 SQL 返回值
c.close() # 关闭
```

集合操作

并 UNION

差 EXCEPT

交 INTERSECT

```
SELECT column_name(s) FROM table_name1
UNION
SELECT column_name(s) FROM table_name2;
```

笛卡尔积 SELECT * FROM A,B

投影 SELECT name FROM A

选择 SELECT name FROM A WHERE ID=1

再建新的表格

```
CREATE TABLE B(ID integer, name text);
INSERT INTO B VALUES(2, 'Li');
INSERT INTO B VALUES(3, 'Zhang');
```

```
SELECT * FROM B;
```

ID	name
2	Li
3	Zhang

并运算

```
SELECT * FROM B UNION
SELECT * FROM A;
```

ID	name
1	Wang
2	Li
3	Zhang

交运算

```
SELECT * FROM B INTERSECT
SELECT * FROM A;
```

ID	name
2	Li

笛卡尔积运算

```
SELECT * FROM A, B;
```

ID	name	ID	name
1	Wang	2	Li
1	Wang	3	Zhang
2	Li	2	Li
2	Li	3	Zhang

差运算

```
SELECT * FROM A
EXCEPT
SELECT * FROM B;
```

ID	name
1	Wang

投影

```
SELECT name FROM A;
```

name
Wang
Li

选择：where 代表选择条件

```
SELECT * FROM A WHERE ID=1;
```

ID	name
1	Wang

关系代数的特色

```
SELECT * FROM A JOIN B ON A.ID=B.ID;
```

ID	name	ID	name
2	Li	2	Li

- 用笛卡尔积和选择

```
SELECT * FROM A, B WHERE A.ID=B.ID;
```

ID	name	ID	name
2	Li	2	Li

左连接

- 左连接，无条件保留左边

```
SELECT * FROM A LEFT JOIN B ON A.ID=B.ID;
```

ID	name	ID	name
1	Wang		
2	Li	2	Li

- SQLite 不支持右连接

```
SELECT * FROM B LEFT JOIN A ON A.ID=B.ID;
```

ID	name	ID	name
2	Li	2	Li
3	Zhang		

GroupBy

```
INSERT INTO A VALUES(4, 'Li');
SELECT * FROM A;
```

ID	name
1	Wang
2	Li
4	Li

```
SELECT name,count(*) AS '人数' FROM A GROUP BY name;
```

name	人数
Li	2
Wang	1

成绩录入

天才少年爱迪生上课时做了一个梦，梦见自己成为了一门课的助教，协助老师向注册中心系统录成绩。但是少年爱迪生在梦境位面中的超能力减半，无法调用他强大的编程战斗值，所以他找到了你。

查看 CSV

```
csvlook dataframe-practice/students.csv | head
```

```
| 学号 | 姓名 | 性别 | 班级    | 联系方式 |
| -- | -- | -- | ---- | ------------ |
|  1 | AB | 女  | 物理71 | 50,626,922,811 |
|  2 | AC | 女  | 工物61 |  6,533,879,773 |
|  3 | AD | 男  | 物理71 | 43,865,400,582 |
|  4 | AE | 男  | 工物71 | 58,581,462,691 |
|  5 | AF | 女  | 物理72 | 45,017,508,911 |
|  6 | AG | 男  | 工物72 | 26,000,243,873 |
|  7 | AH | 男  | 物理71 |  4,295,424,729 |
|  8 | AI | 男  | 工物83 | 51,193,285,308 |
```

SQLite 读入 CSV

```
.mode csv
.import dataframe-practice/students.csv students
SELECT * FROM students LIMIT 5;
```

学号	姓名	性别	班级	联系方式
1	AB	女	物理 71	50626922811
2	AC	女	工物 61	6533879773
3	AD	男	物理 71	43865400582
4	AE	男	工物 71	58581462691
5	AF	女	物理 72	45017508911

- 也可直接使用 `dataframe-practice/people.db` 读入数据。

csvsql

可在命令行使用 SQL 语句查询 CSV 的内容。

查看已经有的表格

```
.tables
```

A B students

- 任务：读入 `scores.csv` `classes.csv`

少年爱迪生想

- 比较物理系和工物系的男女比例
- 算出大家的总评成绩：
  - 小作业权相等，总体占 65% 的成绩
  - 大作业占 30% 的成绩
  - 划分出不同的分数段，给出某分数段同学的手机号
- 画出各班平均小作业成绩的变化曲线

物理系和工物系的男女比例

```
SELECT classes.院系, students.性别, count(*) AS 人数
FROM students JOIN classes
ON students.班级 = classes.班级
GROUP BY classes.院系, students.性别;
```

院系	性别	人数
工物	女	4
工物	男	18
物理	女	10
物理	男	25

选出特定成绩段的学生查询手机号

```
SELECT 联系方式,姓名,
(`curve.fitting`+`gpa.calculator`)/2*0.65 + `大作业`*0.3 AS total
FROM students JOIN scores ON students.学号 = scores.学号
WHERE total < 60;
```

联系方式	姓名	total
26000243873	AG	22.75
11373628062	AK	45.65
11391622912	AM	38.35
9693622361	AR	55.575
73349261755	AY	53.705
99263903250	BE	53.945
99555170920	BJ	58.66
26880267330	BS	36.665
77213592969	BV	47.65

引例：小作业所有加和

```
.schema scores
```

```
CREATE TABLE `scores` (
  `学号` INTEGER,
  `self.intro` INTEGER,
  `a.b` INTEGER,
  `rank.guesser` INTEGER,
  `hdf5` INTEGER,
  `prime` INTEGER,
  `heart.curve` INTEGER,
  `gpa.calculator` REAL,
  `curve.fitting` INTEGER,
  `大作业` REAL
);
```

```
SELECT 联系方式, 姓名,
(`self.intro`+`a.b`+`rank.guesser`+hdf5+prime
+`heart.curve`+`curve.fitting`+`gpa.calculator`)/8*0.65
+ `大作业`*0.3 AS total
FROM students JOIN scores ON students.学号 = scores.学号
WHERE total < 60;
```

联系方式	姓名	total
26000243873	AG	42.575
11391622912	AM	58.0125

- 可以用双引号 " 代替斜引号 `

太丑了

- 四周 16 次作业全都写上吗?
- 不能写循环吗?
- 不能自动完成吗，一次原则在哪里?
- 问题何在?
  - 表格列的对称性。

长表

```
SELECT * FROM longscores LIMIT 5
```

学号	作业	分数	大作业
1	1	100.0	0
2	1	100.0	0
3	1	100.0	0
4	1	100.0	0
5	1	100.0	0

- “作业”一列给出作业编号
- “大作业”一列 0 代表非，1 代表是

再计算一次总评

- 用 longscores 分组大大简化

```
SELECT 学号, AVG(分数) FROM longscores GROUP BY 学号 LIMIT 5
```

学号	AVG(分数)
1	104.411111111111
2	92.3333333333333
3	94.2222222222222
4	104.033333333333
5	99.8333333333333

加上权重

- 0 到 0.65 权重，1 到 0.3 权重
- HAVING 与 groupby 连用，对 group 进行过滤。

```
SELECT 学号, SUM(权重*分数) AS 总评 FROM
(SELECT 学号, CASE 大作业
  WHEN 0 THEN 0.65
  WHEN 1 THEN 0.3
  END AS 权重, AVG(分数) AS 分数
  FROM longscores GROUP BY 学号, 大作业)
GROUP BY 学号
HAVING 总评 > 105
```

学号	总评
49	105.16375
55	105.70625

建立中间结果的表格

使用 CREATE VIEW 为中间结果命名。这是 lazy evaluation，不必担心中间变量的资源占用。

```
CREATE VIEW weights AS
SELECT 学号, CASE 大作业
  WHEN 0 THEN 0.65
  WHEN 1 THEN 0.3
  END AS 权重, AVG(分数) AS 分数
  FROM longscores GROUP BY 学号, 大作业;

CREATE VIEW finals AS
SELECT 学号, SUM(权重*分数) AS 总评 FROM weights
GROUP BY 学号 HAVING 总评 > 105;

SELECT finals.学号,总评,联系方式 FROM finals JOIN students ON finals.学号 = students
```

储存起来用于下一步使用

- CREATE VIEW 把中间结果存成虚拟表格

```
CREATE VIEW final_scores AS
SELECT 学号, SUM(权重*分数) AS 总评 FROM
(SELECT 学号, CASE 大作业
  WHEN 0 THEN 0.65
  WHEN 1 THEN 0.3
  END AS 权重, AVG(分数) AS 分数
  FROM longscores GROUP BY 学号, 大作业)
GROUP BY 学号

SELECT * FROM final_scores LIMIT 5
```

学号	总评
1	104.50375
2	87.20625
3	94.05625
4	103.265
5	98.26875

算出大家的总评成绩

```
SELECT students.学号, 总评, 联系方式
FROM final_scores JOIN students
ON final_scores.学号 = students.学号
WHERE 总评 > 105
```

学号	总评	联系方式
49	105.16375	81920572715
55	105.70625	24891470639

总评成线关联班级

```
SELECT total.班级, 院系, 姓名, 总评, 联系方式 FROM classes JOIN
(SELECT 姓名, 班级, 总评, 联系方式 FROM final_score
JOIN students on final_score.学号 = students.学号 WHERE 总评 < 60) AS total
ON total.班级 = classes.班级
```

班级	院系	姓名	总评	联系方式
工物 72	工物	AG	42.575	26000243873
物理 71	物理	AM	58.0125	11391622912

成绩按班级计算

- ☒ 比较物理系和工物系的男女比例
- ☒ 算出大家的总评成绩：
  - ☒ 小作业权相等，总体占 65% 的成绩
  - ☒ 大作业占 30% 的成绩
  - ☒ 划分出不同的分数段，给出某分数段同学的手机号
- ☐ 画出各班平均小作业成绩的变化曲线

取得班级作业平均数据

```
SELECT 班级, 作业, AVG(分数) AS 分数
FROM longscores JOIN students ON longscores.学号 = students.学号
GROUP BY 班级, 作业 LIMIT 10
```

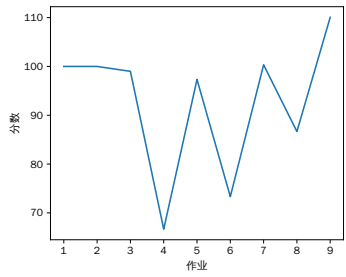
班级	作业	分数
基科 71	1	100.0
基科 71	2	100.0
基科 71	3	99.0
基科 71	4	66.6666666666667
基科 71	5	97.3333333333333
基科 71	6	73.3333333333333
基科 71	7	100.333333333333
基科 71	8	86.6666666666667
基科 71	9	110.066666666667
基科 72	1	100.0

绘图

```
from matplotlib import pyplot as plt
import pandas as pd
import sqlite3

c = sqlite3.connect("dataframe-practice/people.db")
class_score = pd.read_sql_query("""
SELECT 班级, 作业, AVG(分数) AS 分数
FROM longscores JOIN students
ON longscores.学号 = students.学号
GROUP BY 班级, 作业
""", c)

one_line = class_score.query("班级 == '基科71'")
plt.plot(one_line["作业"], one_line["分数"])
plt.xlabel("作业")
plt.ylabel("分数")
```



长表与宽表总结

- 长表，为了恢复列之间的对称性，对计算机系统更友好
- 宽表，某些时候人类易于理解。
  - 不同的列有不同的属性，例如权重
  - 相同属性的列在重复，违反一次原则