SQL学习

第2部分
Part-II

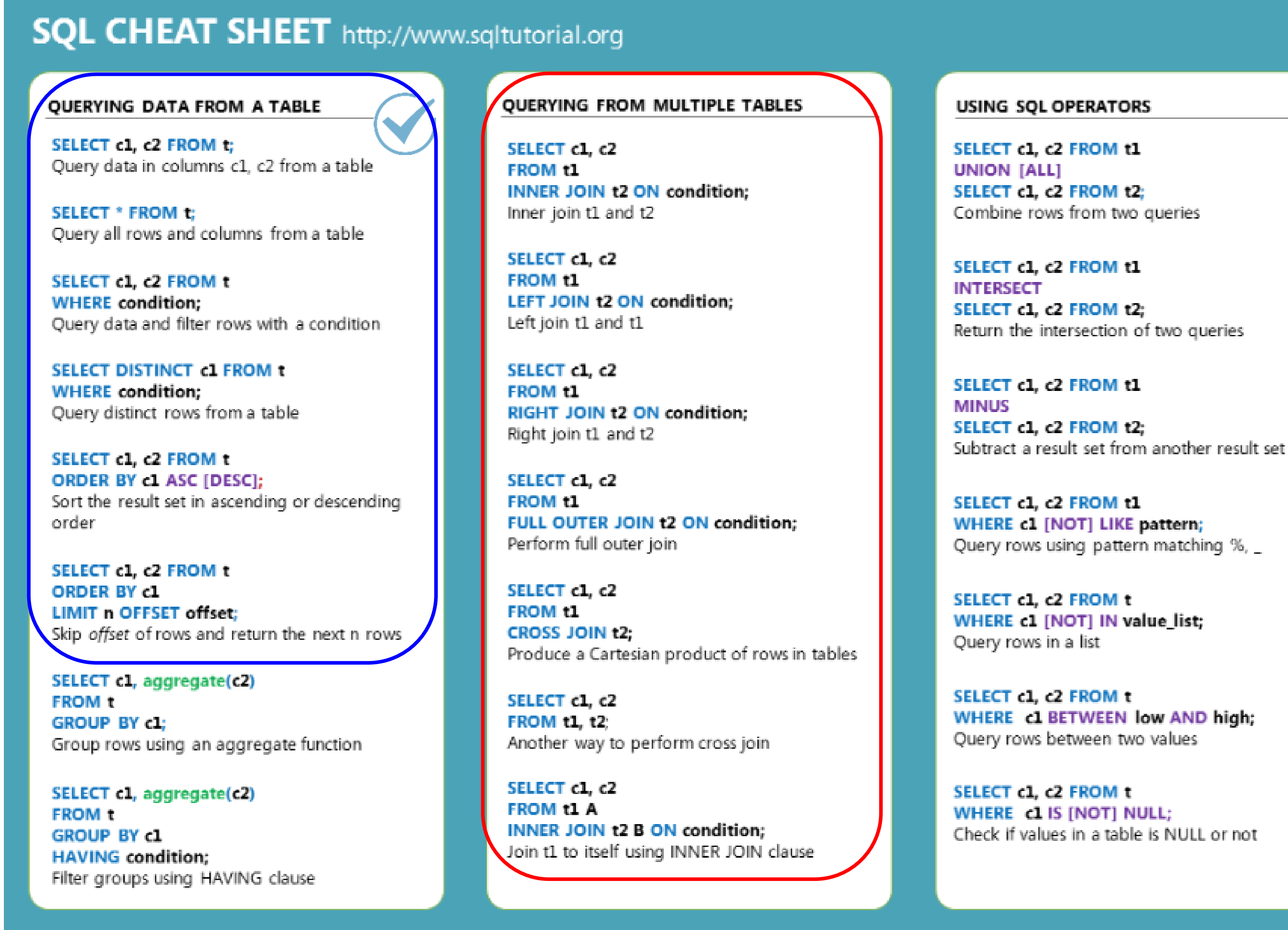
SQL 语言language

简览preview

Preview简览

SQL queries
SQL查询语句

sqltutorial.org/sql-cheat-sheet



SQL CHEAT SHEET http://www.sqltutorial.org

QUERYING DATA FROM A TABLE

```
SELECT c1, c2 FROM t;
```
Query data in columns c1, c2 from a table

```
SELECT * FROM t;
```
Query all rows and columns from a table

```
SELECT c1, c2 FROM t
WHERE condition;
```
Query data and filter rows with a condition

```
SELECT DISTINCT c1 FROM t
WHERE condition;
```
Query distinct rows from a table

```
SELECT c1, c2 FROM t
ORDER BY c1 ASC [DESC];
```
Sort the result set in ascending or descending order

```
SELECT c1, c2 FROM t
ORDER BY c1
LIMIT n OFFSET offset;
```
Skip *offset* of rows and return the next n rows

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1;
```
Group rows using an aggregate function

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition;
```
Filter groups using HAVING clause

QUERYING FROM MULTIPLE TABLES

```
SELECT c1, c2
FROM t1
INNER JOIN t2 ON condition;
```
Inner join t1 and t2

```
SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition;
```
Left join t1 and t1

```
SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition;
```
Right join t1 and t2

```
SELECT c1, c2
FROM t1
FULL OUTER JOIN t2 ON condition;
```
Perform full outer join

```
SELECT c1, c2
FROM t1
CROSS JOIN t2;
```
Produce a Cartesian product of rows in tables

```
SELECT c1, c2
FROM t1, t2;
```
Another way to perform cross join

```
SELECT c1, c2
FROM t1 A
INNER JOIN t2 B ON condition;
```
Join t1 to itself using INNER JOIN clause

USING SQL OPERATORS

```
SELECT c1, c2 FROM t1
UNION [ALL]
SELECT c1, c2 FROM t2;
```
Combine rows from two queries

```
SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2;
```
Return the intersection of two queries

```
SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2;
```
Subtract a result set from another result set

```
SELECT c1, c2 FROM t1
WHERE c1 [NOT] LIKE pattern;
```
Query rows using pattern matching %, _

```
SELECT c1, c2 FROM t
WHERE c1 [NOT] IN value_list;
```
Query rows in a list

```
SELECT c1, c2 FROM t
WHERE c1 BETWEEN low AND high;
```
Query rows between two values

```
SELECT c1, c2 FROM t
WHERE c1 IS [NOT] NULL;
```
Check if values in a table is NULL or not

目录 Table of Contents

练习代码：SQL-1.ipynb

SQL Definitions
SQL 定义

Principles

基本原理

What you will learn about in this section 以下学习内容

1. 什么是SQL？ What is SQL?
2. 基本的模式定义 Basic schema definitions
3. 键和约束 Keys & constraints intro

SQL简介SQL Introduction

- SQL is a standard language for querying and manipulating data

SQL是一种是查询和操作数据的标准语言

- SQL is a **very high-level** programming language

This works because it is optimized well!
SQL是一种高级编程语音，底层做了很多优化工作。

- Many standards out there:
- SQL有众多的标准

ANSI SQL, SQL92 (a.k.a. SQL2), SQL99 (a.k.a. SQL3),
SQL2003, 2006, 2008, 2011, 2016, ….

SQL stands for
Structured
Query
Language

SQL语言是… SQL is a…

- 数据定义语言 Data Definition Language (DDL)

    定义关系模式 Define relational schemata

    创建修改删除数据表和属性 Create/alter/delete tables and their attributes

- 数据操作语言 Data Manipulation Language (DML)

    查询单表或多表 Query one or more tables

    插入删除修改数据表的数据 Insert/delete/modify tuples in tables

集合代数 Set algebra

多集 （multiset），一个允许有重复的集合

连表 List:	[1, 1, 2, 3]
集合 Set:	{1, 2, 3}
多集 Multiset:	{1, 1, 2, 3}

A **multiset** is an unordered list (or: a set with multiple duplicate instances allowed)
一个多集是无序连表（允许有多个重复的集合）

合并运算 UNIONs

Set: {1, 2, 3} U { 2 } = { 1, 2, 3 }
Multiset: {1, 1, 2, 3} U { 2 } = { 1, 1, 2, 2, 3 }

交叉积 Cross-product

{1, 1, 2, 3} * { y, z } =
{ <1, y>, <1, y>, <2, y>, <3, y>,
<1, z>, <1, z>, <2, z>, <3, z>
}

i.e. no *next()*, etc. methods!
即，没有next()方法

数据表 Tables in SQL

关系或 表（Relation/Table）

Product

PName	Price	Manuf
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A **relation** or **table** is a multiset of tuples having the attributes specified by the schema
一个**关系或表**是元组组成的多集，元组包含了模式定义的属性。

Let's break this definition down
让我们进一步分解下去。

数据表 Tables in SQL

Product

属性或 列（Attribute/Column）

PName	Price	Manuf
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

An **attribute** (or **column**) is a typed data entry present in each tuple in the relation
属性或**列**是元组中的类型化的数据项。

*NB: Attributes must have an **atomic** type in standard SQL, i.e. not a list, set, etc.*
注意：在标准的SQL中，属性必须包含原子类型，即，不能是连表，集合等

数据表 Tables in SQL

Product

PName	Price	Manuf
Gizmo	**$19.99**	**GizmoWorks**
Powergizmo	**$29.99**	**GizmoWorks**
SingleTouch	**$149.99**	**Canon**
MultiTouch	**$203.99**	**Hitachi**

元组 或行或记录（Tuple/Row/Record）

A **tuple** or **row** is a single entry in the table having the attributes specified by the schema
元组或**行**是一个数据表中的包含由模式指定的属性的单个数据条目

Also referred to sometimes as a ***Record***
元组有时又称为**记录**

数据表 Tables in SQL

Product

PName	Price	Manuf
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

势 和元数 (cardinality/arity)

The number of tuples is the **cardinality** of the relation
一个关系中的元组的数目，被称为势（类似集合术语）

The number of attributes is the **arity** of the relation
一个关系中属性的个数被称为元数（arity）

数据类型 Data Types in SQL

原子数据类型 Atomic types:

- 字符串Characters: CHAR(20), VARCHAR(50)
- 数值Numbers: INT, BIGINT, SMALLINT, FLOAT
- 其它Others: 货币MONEY, 日期DATETIME, …

Every attribute must have an atomic type

每一个属性必须具有一个原子类型

- Hence tables are flat
- 因此表是平的（不能再分解）

数据表模式 Table Schemas

The **schema** of a table is the table name, its attributes, and their types:

数据表模式是指表名称，表的属性和属性的类型

Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

A **key** is an attribute whose values are unique; we underline a key

键是指值是唯一的属性，键加下划线

Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

键的约束 Key constraints

A **key** is a **minimal subset of attributes** that acts as a unique identifier for tuples in a relation
键是属性的最小子集可以作为关系中的元组的唯一标识符

- A key is an implicit constraint on which tuples can be in the relation
- 键是一个对于关系中元组的隐性的约束
- i.e. if two tuples agree on the values of the key, then they must be the same tuple!
- 也就是说，如果有两个元组的键的值是相等的，那么他们一定是同一个元组

Students(sid:string, name:string, gpa: float)

1. Which would you select as a key?
你会选择哪些属性作为键
2. Is a key always guaranteed to exist?
是否一定存在一个键
3. Can we have more than one key?
是否可以有一个到多个键

模式申明 Declaring Schema

Students(sid: *string,* name: *string,* gpa: *float*)

```
CREATE TABLE Students (
  sid CHAR(20),
  name  VARCHAR(50),
  gpa  float,
  PRIMARY KEY (sid)
)
```

空值和非空值NULL and NOT NULL

- To say “don’t know the value” we use NULL
- 对于某个属性，我们不知道它的值，称为NULL

NULL has (sometimes painful) semantics, more detail later

NULL是一种语义（有时代价颇大），后续还会介绍

Students(sid:string, name:string, gpa: float)

sid	name	gpa
123	Bob	3.9
143	Jim	NULL

Say, Jim just enrolled in his first class.
比如说，Jim刚选了第一门课，当然没有学分绩

In SQL, we may constrain a column to be NOT NULL, e.g., “name” in this table
在SQL中，我们可以增加约束某一列属性为非空，比如表中的“name”属性。

空值Null Values

Unexpected behavior:
意想不到的行为

```
SELECT *
FROM   Person
WHERE  age < 25 OR age >= 25
```

Some Persons are not included !
有些“人员”没有包含进来，为什么？

空值Null Values

可以显示测试是否为空值:
Can test for NULL explicitly:

- x IS NULL
- x IS NOT NULL

```
SELECT *
FROM   Person
WHERE  age < 25 OR age >= 25
    OR age IS NULL
```

Now it includes all Persons!
现在包含了所有“人员”

单选题 1分

设置

SQL查询操作的底层的数据结构是?

A 连表List

B 集合Set

[✓ C] 多集MultiSet

D 数组Array

提交

2. Single - table queries
单表查询

What you will learn about in this section
以下学习内容

The SFW(Select-From-Where expression) query

SFW(Select-From-Where 表达式) 查询

选择与投射操作

Other useful operators: LIKE, DISTINCT, ORDER BY

其它有用的算子： LIKE, DISTINCT, ORDER BY

SQL 查询Query

- Basic form (there are many many more bells and whistles)
- 基本形式

```
SELECT <attributes>

FROM   <one or more relations>

WHERE  <conditions>
```

SQL 查询的基本结构由三个子句（suclause）构成：
select、from 和where.
称为SFW查询 Call this a **SFW** query.

简单SQL查询：选择
Simple SQL Query: Selection

Selection is the operation of filtering a relation's tuples on some condition

选择是一种操作过滤出符合条件的行

PName	Price	Category	Manuf
Gizmo	$19.99	Gadgets	GWorks
Powergizmo	$29.99	Gadgets	GWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT *
FROM   Product
WHERE  Category = 'Gadgets'
```

⇩

PName	Price	Category	Manuf
Gizmo	$19.99	Gadgets	GWorks
Powergizmo	$29.99	Gadgets	GWorks

简单SQL查询：投射
Simple SQL Query: Projection

<u>Projection</u> is the operation of producing an output table with tuples that have a subset of their prior attributes

投射是一种操作生成属性集的子集

PName	Price	Category	Manuf
Gizmo	$19.99	Gadgets	GWorks
Powergizmo	$29.99	Gadgets	GWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT Pname, Price, Manufacturer

FROM   Product

WHERE  Category = ‘Gadgets’
```

PName	Price	Manuf
Gizmo	$19.99	GWorks
Powergizmo	$29.99	GWorks

术语Notation

Input Schema
输入模式

Product(<u>PName</u>, Price, Category, <u>Manufacturer</u>)

```
SELECT Pname, Price, Manufacturer
FROM   Product
WHERE  Category = 'Gadgets'
```

Output Schema
输出模式

Answer(PName, Price, Manfacturer)

一些细节 A Few Details

SQL **commands** are case insensitive:

Same: SELECT, Select, select

Same: Product, product

SQL 命令是大小写无关

Values are **not:**

Different: ‘Seattle’, ‘seattle’

属性值是大小写敏感的

Use single quotes for constants:

‘abc’ - yes

“abc” - no

一些细节 A Few Details

SQL 语句可以写成一行，也可以分写为多行。

多条 SQL 语句必须以分号（;）分隔。

SQL 语句时，所有空格都被忽略。

SQL 支持三种注释

```
## 注释
-- 注释
/* 注释 */
```

LIKE：简单的字符串匹配
LIKE: Simple String Pattern Matching

```
SELECT *
FROM   Products
WHERE PName LIKE '%gizmo%'
```

s **LIKE** p: pattern matching on strings
p may contain two special symbols:

- % = any sequence of characters
- _ = any single character

消除重复DISTINCT: Eliminating Duplicates

```
SELECT DISTINCT Category
FROM   Product
```

Category
Gadgets
Photography
Household

Versus

```
SELECT Category
FROM   Product
```

Category
Gadgets
Gadgets
Photography
Household

ORDER BY:对查询结果进行排序
ORDER BY: Sorting the Results

```
SELECT      PName, Price, Manufacturer
FROM         Product
WHERE       Category='gizmo' AND Price > 50
ORDER BY  Price, PName
```

Ordering is ascending, unless you specify the DESC keyword.
顺序默认是升序，除非用DESC申明是降序

Ties are broken by the second attribute on the ORDER BY list, etc.
排序会根据第二个属性，打破平局

3. Multiple - table queries: JOIN
多表查询：JOIN

What you will learn about in this section
以下学习内容

关联查询:

连接JOINs

内连接Inner JOINs

外连接Outer JOINs

连接Joins

Product(<u>PName</u>, Price, Category, Manufacturer)

Company(<u>CName</u>, StockPrice, Country)

Ex: Find all products under $200 manufactured in Japan; return their names and prices.
找出所有在日本生成的，价格200美元以下的产品的名称和价格

连接Joins

Product(<u>PName</u>, Price, Category, Manufacturer)

Company(<u>CName</u>, StockPrice, Country)

Several equivalent ways to write a basic join in SQL:

一些等价的写法进行基本连接操作

```
SELECT  PName, Price
FROM    Product
JOIN    Company
ON      Manufacturer = Cname
WHERE   Price <= 200
             AND Country=‘Japan’
```

```
SELECT  PName, Price
FROM    Product, Company
WHERE   Manufacturer = CName
        AND Country=‘Japan’
            AND Price <= 200
```

A few more later on

连接Joins

Product

PName	Price	Category	Manufacturer
Gizmo	$19	Gadgets	GizmoWorks
Powergizmo	$29	Gadgets	GizmoWorks
SingleTouch	$149	Photography	Canon
MultiTouch	$203	Household	Hitachi

Company

CName	Stock Price	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

```
SELECT PName, Price
FROM   Product, Company
WHERE  Manufacturer = CName
          AND Country='Japan'
                AND Price <= 200
```

PName	Price
SingleTouch	$149

Tuple Variable Ambiguity in Multi-Table
多表查询中元组变量的含糊性

Person(name, address, worksfor)

Company(name, address)

```
1.  SELECT DISTINCT name, address
2.  FROM  Person, Company
3.  WHERE     worksfor = name
```

Which "address" does this refer to**?**

"address"是指那个表的属性

Which name"s??

"name"是指那个表中的属性

Tuple Variable Ambiguity in Multi-Table 多表查询中元组变量的含糊性

```
Person(name, address, worksfor)
Company(name, address)
```

```
SELECT DISTINCT Person.name, Person.address
FROM           Person, Company
WHERE            Person.worksfor = Company.name
```

```
SELECT DISTINCT p.name, p.address
FROM           Person p, Company c
WHERE            p.worksfor = c.name
```

Both equivalent ways to resolve variable ambiguity
以上两种方法都可以解决含糊性问题

Semantics of JOINs
连接的语义

```
SELECT $x_1.a_1$, $x_2.a_2$, ⋯, $x_n.a_k$
FROM   $R_1$ AS $x_1$, $R_2$ AS $x_2$, ⋯, $R_n$ AS $x_n$
WHERE  Conditions($x_1$,⋯, $x_n$)
```

Note:
This is a ***multiset*** union

注意:
这是多集的并集操作

```
Answer = {}
for $x_1$ in $R_1$ do
        for $x_2$ in $R_2$ do
        .....
        for $x_n$ in $R_n$ do
                if Conditions($x_1$,…, $x_n$)
                then Answer = Answer $\cup$ {($x_1.a_1$, $x_1.a_2$, …, $x_n.a_k$)}
return Answer
```

Semantics of JOINs
连接的语义

```
SELECT R.A
FROM   R, S
WHERE  R.A = S.B
```

- Take **cross product**

  X = R x S

  交叉积运算

- Apply **selections/conditions**

  Y = {(r, s) in X | r.A == s.B}

  应用选择，进行条件过滤

- Apply **projections** to get final output

  Z = (y.A) for y in Y

  应用投射，获得最终结果

Recall: 回忆
Cross product (A X B) is the set of all unique tuples in A,B
A和B交叉积
Ex: {a,b,c} X {1,2} = {(a,1), (a,2), (b,1), (b,2), (c,1), (c,2)}

= Filtering!
过滤

= Returning only *some* attributes
仅仅返回某些属性

Remembering this order is critical to understanding the output of certain queries (see later on…)
记住连接语义的这个内部操作顺序，它是理解某些查询输出的关键（以后就会看到）

An example of JOIN semantics
一个连接语义示例

```
SELECT R.A
FROM   R, S
WHERE  R.A = S.B
```

R

A
1
3

S

B	C
2	3
3	4
3	5

Cross Product

A	B	C
1	2	3
1	3	4
1	3	5
3	2	3
3	3	4
3	3	5

Apply Selections / Conditions

A	B	C
3	3	4
3	3	5

Apply Projection

Output

A
3
3

课堂内小任务1

- 创建完成两张表，产品表Product和采购表Purchase，自行填充数据
  - 1.完成内连接（Inner Join）
  - 2.完成左外连接（Left Outer Join）

RECAP: Joins
复习：连接

By default, joins in SQL are **“inner joins”**:
缺省情况下，SQL是默认内连接

```
Product(name, category)
Purchase(prodName, store)
```

1

```
SELECT Product.name, Purchase.store
FROM   Product
  JOIN Purchase ON Product.name = Purchase.prodName
```

2

```
SELECT Product.name, Purchase.store
FROM   Product, Purchase
WHERE  Product.name = Purchase.prodName
```

Both equivalent:
Both INNER JOINS!
以上情况是等价的:
都是内连接

内连接 INNER JOIN

Product

name	category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

prodName	store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

3

```
SELECT Product.name, Purchase.store
FROM   Product
  INNER JOIN Purchase
                    ON Product.name = Purchase.prodName
```

name	store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Note: another equivalent way to write an INNER JOIN!
注意：另一种写内连接的等价方法

外连接 Outer Joins

```
SELECT Product.name, Purchase.store
FROM   Product
  LEFT OUTER JOIN Purchase ON
                    Product.name = Purchase.prodName
```

Left outer joins in SQL:

Now we'll get products even if they didn't sell
现在，我们将获得所有产品，即使它们没有销售

- An **outer join** returns tuples from the joined relations that don't have a corresponding tuple in the other relations
- 外连接返回元组，该元组在连接的关系里一方没有对应的元组
  - I.e. If we join relations A and B on a.X = b.X, and there is an entry in A with X=5, but none in B with X=5…
  - A LEFT OUTER JOIN will return a tuple (a, NULL)!
  - 比如，连接关系relations A 和 B on a.X = b.X, 其中A里有属性值 X=5, 但是在B中没有对应的属性值 X=5…
  - 左外连接将返回元组(a, NULL)!

LEFT OUTER JOIN
左外连接

Product

name	category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

prodName	store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

```
SELECT Product.name, Purchase.store
FROM   Product
  LEFT OUTER JOIN Purchase
                ON Product.name = Purchase.prodName
```

→

name	store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
OneClick	NULL

单选题 1分

设置

JOIN连接操作基于的数学运算是?

A 内积(Inner product)

[✓ B 交叉积(Cross Product)]

C 卷积

提交

Other Outer Joins
其它外连接

- Left outer join:
  - Include the left tuple even if there’s no match
- 左外连接
  - 包括左元组，即使没有匹配的

- Right outer join:
  - Include the right tuple even if there’s no match
- 右外连接
  - 包括右元组，即使没有匹配的

- Full outer join:
  - Include the both left and right tuples even if there’s no match
- 全外连接
  - 包括左元组和右元组，即使没有匹配的

多选题 1分

设置

多表查询的（ JOIN ）连接操作有哪些?

A Inner JOIN

B Left JOIN

C Right JOIN

D Full JOIN

提交

参考资料

- CS145: Data Management and Data Systems
- CS245: Principles of Data-Intensive Systems

谢谢指正！