DS 1300 - Introduction to SQL
Part 2 – Multi-table Queries

By Michael Hahsler
based on slides for CS145 Introduction to Databases (Stanford)

What you will learn about in this section

1. Foreign key constraints
2. Joins: basics
3. Joins: SQL semantics
4. Activities: Multi-table queries

Foreign Key Constraints

- Suppose we have the following schema:

  Students(sid: *text*, name: *text*, gpa: *real*)
  Enrolled(student_id: text, cid: *text*, grade: *real*)

- And we want to impose the following constraint:
  - 'Only existing students may enroll in courses' i.e. a student must appear in the Students table to enroll in a class

Note: student_id alone is not a key- what is?

Students

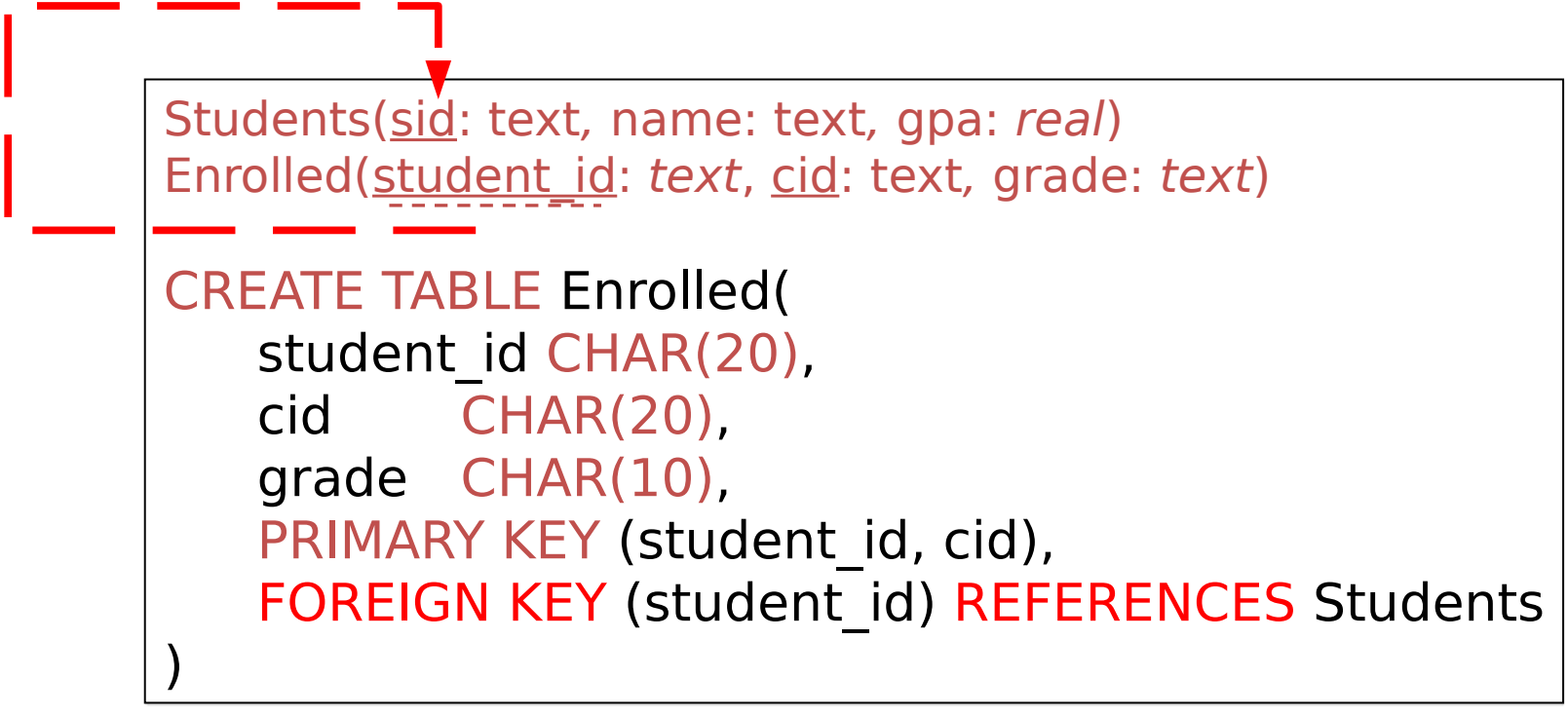
sid	name	gpa
101	Bob	3.2
123	Mary	3.8

Enrolled

student_id	cid	grade
123	564	A
123	537	A+

We say that student_id is a **foreign key** that refers to Students

Declaring Foreign Keys



Foreign Keys and Update Operations

Students(sid: *text*, name: *text*, gpa: *real*)
Enrolled(student_id: text, cid: *text*, grade: *text*)

- What if we insert a tuple into Enrolled, but no corresponding student?
  - INSERT is rejected (foreign keys are constraints)!

- What if we delete a student?
  1. **Disallow the delete**
  2. Remove all of the courses for that student
  3. *SQL allows a third via NULL (not yet covered)*

SQLite: Enable foreign keys with PRAGMA foreign_keys = ON;

DB Browser: check "Foreign Keys" in "Edit Pragma"

Keys and Foreign Keys

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

What is a foreign key vs. a key here?

Product

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

Keys and Foreign Keys

Company(CName, StockPrice, Country)
Product(PName, Price, Category, Manufacturer)

- This example uses **natural keys**.
- Often **surrogate keys** are used instead:

Company(Cnumber, CName, StockPrice, Country)
Product(Pnumber, Pname, Price, Category, ManufNumber)

- Why?
- Why do we use SMU IDs and Social Security Numbers?

Joins

Product(PName, Price, Category, Manufacturer)
Company(CName, StockPrice, Country)

Ex: Find all products under $200 manufactured in Japan;
return their names and prices.

This will need information from both tables…

Joins

Product(PName, Price, Category, Manufacturer)
Company(CName, StockPrice, Country)

Ex: Find all products under $200 manufactured in Japan;
return their names and prices.

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

A **join** between tables returns all unique combinations of their tuples **which meet the join condition**

Joins

Product(PName, Price, Category, Manufacturer)
Company(CName, StockPrice, Country)

Several equivalent ways to write a basic join in SQL:

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

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

Joins

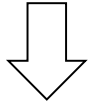
Product

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

Company

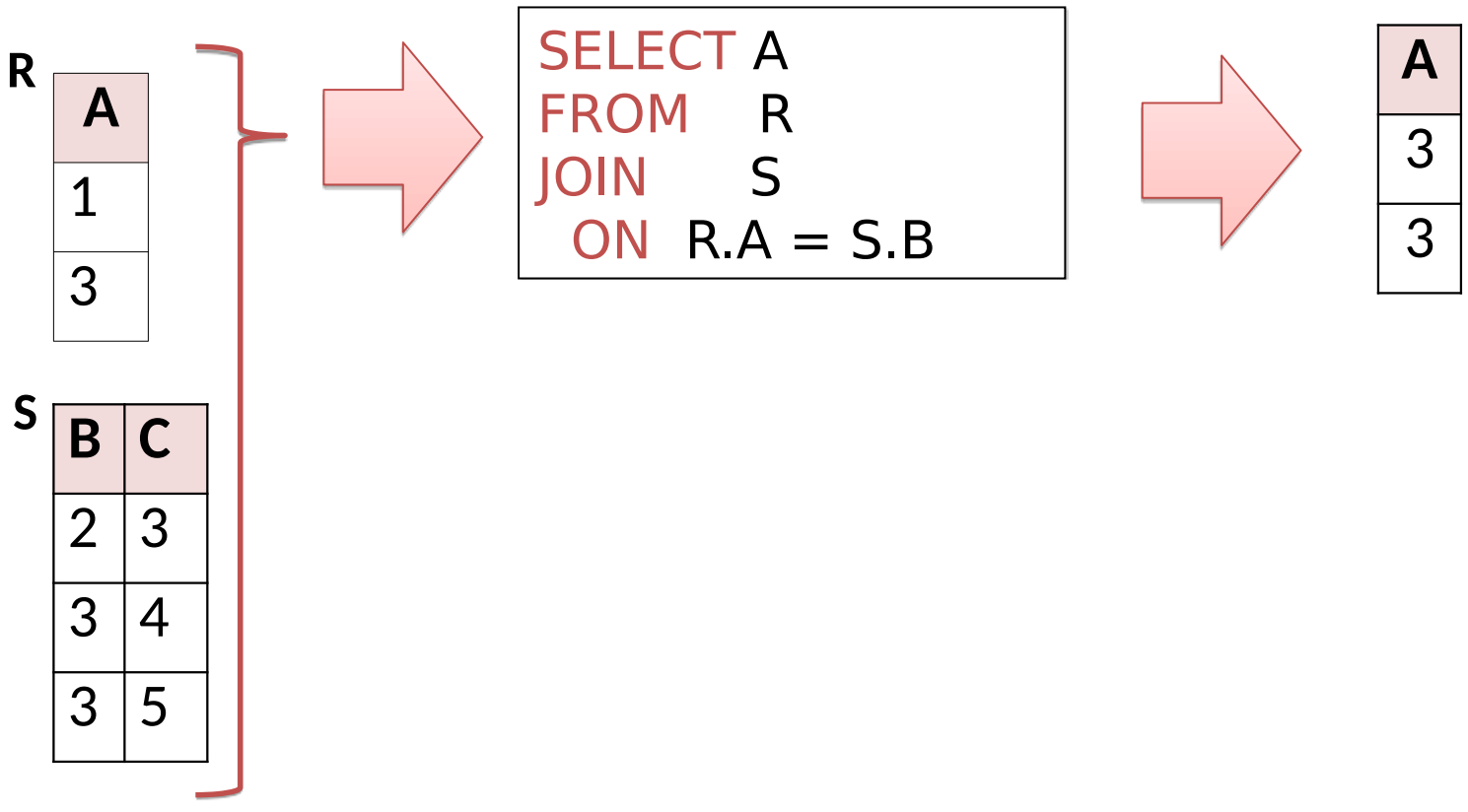
Cname	Stock	Country
GWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

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

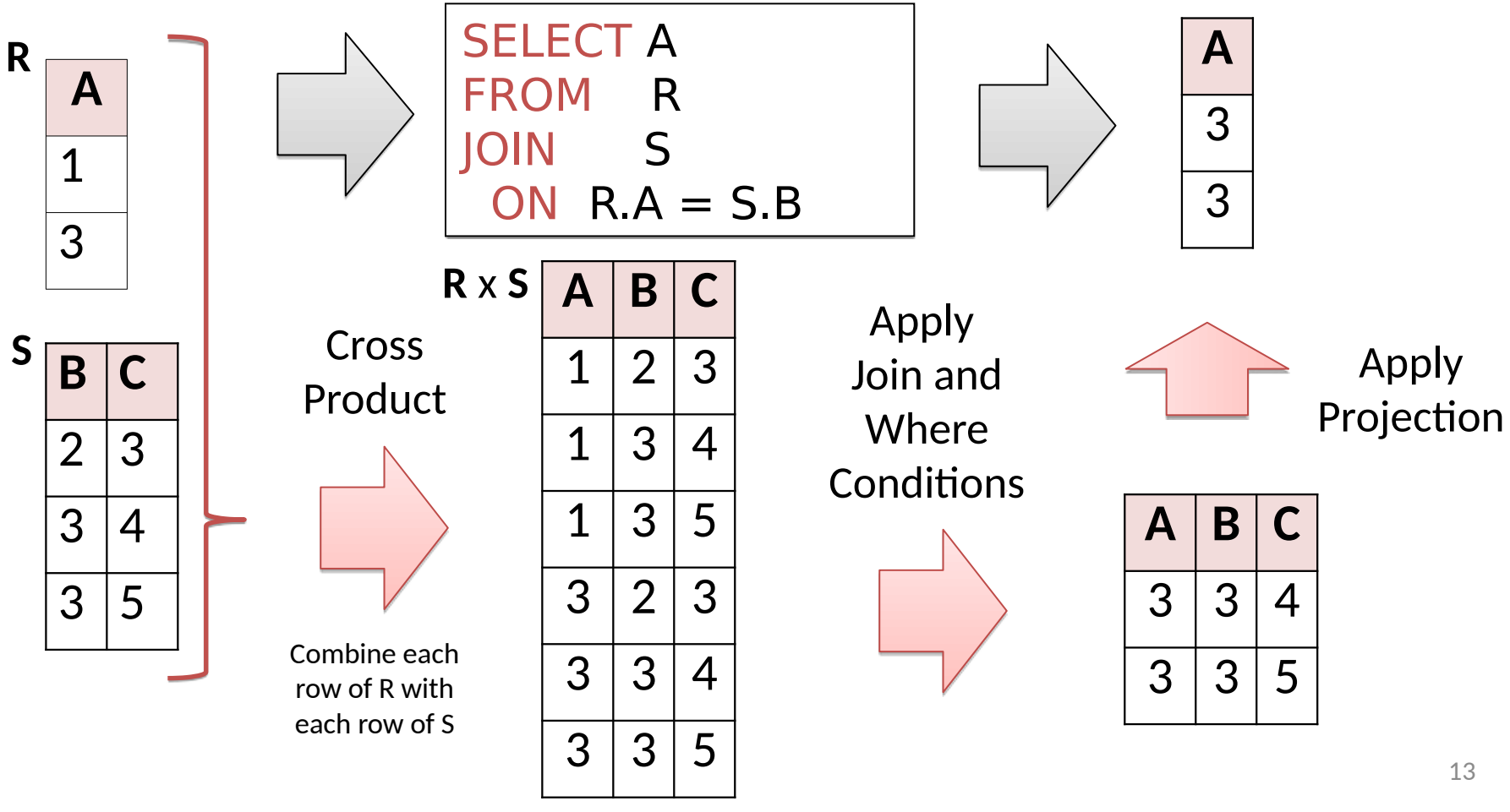


PName	Price
SingleTouch	$149.99

An Example of SQL Semantics



An Example of SQL Semantics



Tuple Variable Ambiguity in Multi-Table

A person works for a company

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

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

Which "address" does this refer to?

Which "name"s??

Tuple Variable Ambiguity in Multi-Table

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

Both equivalent ways to resolve variable ambiguity

```
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
```

A Note on Semantics

- “semantics” is not equal to “execution order”
- The preceding slides show *what a join means*
- Not actually how the DBMS executes it under the covers

Activities

1. Create the product/company database from the slide set. Add the following relation

   Purchase(id, product, buyer).

   with the appropriate foreign key constraints and add some data.

2. Find all countries that manufacture some product in the ‘Gadgets’ category (shows each country only once).

3. Find all products that are manufactured in the US sorted by price.

4. For a given buyer, in how many different countries are the products she purchases manufactured?