Introduction to Data Management
CSE 344

Lecture 4: Joins and Aggregates

Monday June 26

Announcements

- HW1 is due tomorrow at 11pm

- WQ2 is out – due Sunday (July 2)
- HW2 will be out Wednesday.
  - Write queries using real-world dataset
  - Due Wednesday July 5

- There will be class on July 3rd

Today

- Inner joins (6.2)
- Outer joins (6.3.8)
- Aggregations (6.4.3 – 6.4.6)

Our SQL Tool Box

- Selection (WHERE)
- Projection (Attribute List)
- Ordering and distinct
- Join (FROM tableA , tableB)

Can compose these into complex queries

Example Relations and Data

```
Company(name VARCHAR PRIMARY KEY, country VARCHAR, size INT)
Product(name VARCHAR, type VARCHAR, company REFERENCES
        Company.name, PRIMARY KEY (name, company) )
```

name	country	size
Tata	India	660,000
Hitachi	Japan	335,000
Therano	USA	500

name	type	company
Estate	Car	Tata
OneClick	Camera	Hitachi
Nano	Car	Tata
Echelon	Medical	Hitachi

Review: Joins in SQL

Implicit
Cross Join

name		ze
Tat		000
		5 000
Th		500

name	type	company
Estate	Car	Tata
OneClick	Camera	Hitachi
Nano	Car	Tata
Echelon	Medical	Hitachi

3 X 4

```
SELECT P.name, C.country
FROM   Product as P, Company as C
```

What is the **cardinality** of this query?

A) 3 **B)** 4 **C)** 8 **D)** 12

(Inner) joins

```
SELECT P.type
FROM   Product as P, Company as C
WHERE C.country='Japan' AND P.type LIKE 'C%'
       AND P.company = C.name
```

Product

name	type	company
Estate	Car	Tata
OneClick	Camera	Hitachi
Nano	Car	Tata
Echelon	Medical	Hitachi

Company

name	country	size
Tata	India	660,000
Hitachi	Japan	335,000
Therano	USA	500

name	type	company	name	country	size
OneClick	Camera	Hitachi	Tata	India	660,000

12 Rows

(Inner) joins

```
SELECT P.type
FROM   Product as P, Company as C
WHERE  country='Japan' AND type LIKE 'C%'
       AND P.company = C.name
```

Selection: P.comapny = C.name

name	type	company	name	country	size
Estate	Car	Tata	Tata	India	660,000
OneClick	Camera	Hitachi	Hitachi	Japan	335,000
Nano	Car	Tata	Tata	India	660,000
Echelon	Medical	Hitachi	Hitachi	Japan	335,000

(Inner) joins

```
SELECT P.type
FROM   Product as P, Company as C
WHERE  country='Japan' AND type LIKE 'C%'
       AND P.company = C.name
```

More Selection: country = ‘Japan’ and type LIKE ‘C%’

name	type	company	name	country	size
OneClick	Camera	Hitachi	Hitachi	Japan	335,000

Projection

type
Camera

(Inner) joins

```
SELECT P.type
FROM   Product as P, Company as C
WHERE  country= 'Japan' AND type LIKE 'C%'
       AND P.company = C.name
```

Alternative syntax:

```
SELECT P.type
FROM Product  as P INNER JOIN Company as C
ON P.company = C.name
WHERE country = 'Japan' AND type LIKE 'C%'
```

Emphasizes that the predicate is part of the join.

(Inner) Joins Semantics

```
SELECT   x1.a1, x2.a2, … xm.am
FROM     R1 as x1, R2 as x2, … Rm as xm
WHERE    Cond
```

Nested loop semantics

```
for x1 in R1:
  for x2 in R2:
    ...
      for xm in Rm:
          if Cond(x1, x2…):
              output(x1.a1, x2.a2, … xm.am)
```

Joining Practice

```
Product(name, type, company)
Company(name, country, size)
-- Product.company is foreign key to Company.name
```

Retrieve all Japanese companies that manufacture products in both 'Camera' and 'Medical' categories (type)

From P, C Whe P.company = C.name

What is the FROM clause

C.country = Japan; and
P.type = camera and P.type = Medical

Joining Practice

```
Product(name, type, company)
Company(name, country, size)
-- Product.company is foreign key to Company.name
```

Retrieve all Japanese companies that manufacture products in both 'Camera' and 'Medical' categories (type)

```
SELECT DISTINCT C.name
FROM Product as P1, Product as P2, Company as C
WHERE country = 'Japan' AND P1.type = 'Camera'
      AND P2.type = 'Meidcal'
      AND P1.company = C.name
      AND P2.company = C.name;
```

} 2 FK conditions

Self-Joins and Tuple Variables

- Find all companies that manufacture both products in the ‘gadgets’ and ‘photo’ category
- Joining Product with Company is insufficient: need to join Product, with Product, and with Company
- When a relation occurs twice in the FROM clause we call it a *self-join*
  - in that case we must use tuple variables (why?)

Tuple Variables also make complex queries shorter.

Self-joins

```
SELECT DISTINCT C.name
FROM Product as P1, Product as P2, Company as C
WHERE country = 'Japan' AND P1.type = 'Camera'
      AND P2.type = 'Meidcal'
      AND P1.company = C.name
      AND P2.company = C.name;
```

Product

name	type	company
te	Car	Tata
OneClick	Camera	Hitachi
o	Car	Tata
Echelon	Medical	Hitachi

P1

P2

P2.type != 'Medical'

Company

name	country	size
Tata	India	660,000
Hitachi	Japan	335,000
Therano	USA	500

C

restrict to country = 'Japan'

Self-joins

```
SELECT DISTINCT C.name
FROM Product as P1, Product as P2, Company as C
WHERE country = 'Japan' AND P1.type = 'Camera'
      AND P2.type = 'Meidcal'
      AND P1.company = C.name
      AND P2.company = C.name;
```

Product

name	type	company
te	Car	Tata
OneClick	Camera	Hitachi
Nano	Car	Tata
elon	Medical	Hitachi

P1

P2

P2.company != C.name

Company

name	country	size
Tata	India	660,000
Hitachi	Japan	335,000
Therano	USA	500

C

restrict to country = ‘Japan’

Self-joins

```
SELECT DISTINCT C.name
FROM Product as P1, Product as P2, Company as C
WHERE country = 'Japan' AND P1.type = 'Camera'
      AND P2.type = 'Meidcal'
      AND P1.company = C.name
      AND P2.company = C.name;
```

Product

name	type	company
te	Car	Tata
OneClick	Camera	Hitachi
o	Car	Tata
Echelon	Medical	Hitachi

P1

P2

All conditions are True

Company

name	country	size
Tata	India	660,000
Hitachi	Japan	335,000
Therano	USA	500

C

restrict to country = ‘Japan’

Self-joins

```
SELECT DISTINCT C.name
FROM Product as P1, Product as P2, Company as C
WHERE country = 'Japan' AND P1.type = 'Camera'
      AND P2.type = 'Meidcal'
      AND P1.company = C.name
      AND P2.company = C.name;
```

Product

name	type	company
P1 te	Car	Tata
OneClick	Camera	Hitachi
P2 no	Car	Tata
Echelon	Medical	Hitachi

Company

name	country	size
Tata	India	660,000 C
Hitachi	Japan	335,000
Therano	USA	500

P1.name	P1.type	P1.company	P2.Name	P2.type	P2.company	C.name	c.country	c.size
OneClick	Camera	Hitachi	Echelon	Medical	Hitachi	Hitachi	Japan	3350,00

Self-joins

Product

name	type	company
Estate	Car	Tata
OneClick	Camera	Hitachi
Nano	Car	Tata
Echelon	Medical	Hitachi

Company

name	country	size
Tata	India	660,000
Hitachi	Japan	335,000
Therano	USA	500

4 x 4 x 3

```
SELECT P1.name , P2.name, C.name, C.country
FROM Product as P1, Product as P2, Company as C
```

What is the **cardinality** of this query?

A) 4 **B)** 8 **C)** 24 **D)** 48

Joins: Missing Data

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

-- prodName is foreign key
```

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

We want to include products that are never sold, but these are never listed. Why?

Outer joins

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

-- prodName is foreign key
```

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

Left Outer Join includes all data from first table.
Even if there is no match.

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

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 JOIN Purchase ON
       Product.name = Purchase.prodName
```

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 JOIN Purchase ON
       Product.name = Purchase.prodName
```

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Output

Name	Store
Gizmo	Wiz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Output

Name	Store
Gizmo	Wiz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Output

Name	Store
Gizmo	Wiz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Output

Name	Store
Gizmo	Wiz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Output

Name	Store
Gizmo	Wiz
Camera	Ritz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Output

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Output

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

no match

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Output

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Output

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
OneClick	NULL

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
Phone	Foo

Output

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
OneClick	NULL

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

Purchase.prodName

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
Phone	Foo

Output

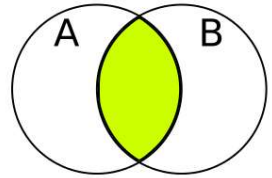
Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
OneClick	NULL
NULL	Foo

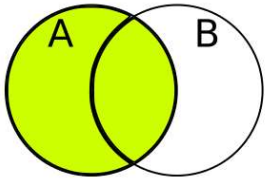
Phone

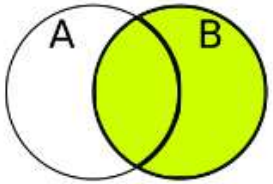
SQL Joins

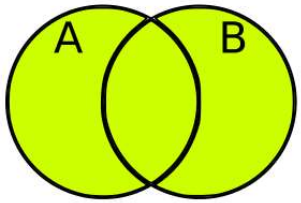
```
FROM A (LEFT/RIGHT/FULL) (OUTER/INNER) JOIN B ON p
```

- **Inner Join:**
  - Only tuples in `A` & `B`
- **Left Outer Join:**
  - All tuples from `A` even if not match
- **Right Outer Join**:
  - Only tuples from `B` even if no match
- **Full Outer Join:**
  - All tuples from both even if no match
- In all cases:
  - Patch tuples without matches using **`NULL`**









RIGHT and FULL JOINS are not implemented in SQLite

Simple Aggregations

Five basic aggregate operations in SQL

```
select count(*) from Purchase
select sum(quantity) from Purchase
select avg(price) from Purchase
select max(quantity) from Purchase
select min(quantity) from Purchase
```

Except count, all aggregations require an attribute

```
select count(price) from Purchase
select sum(quantity * price) from Purchase
```

Aggregates and NULL Values

Null values are not used in aggregates

```
insert into Purchase
(pid, product, price, quantity, month)
values(12, 'gadget', NULL, NULL, 'april')
```

Let's try the following

- `select count(*) from Purchase` 1
- `select count(quantity) from Purchase` 0

- `select sum(quantity) from Purchase` 0
- `select count(*) from Purchase where quantity is not null;` 0

Counting Duplicates

`COUNT` applies to duplicates, unless otherwise stated:

```
SELECT count(product)
FROM   Purchase
WHERE  price > 4.99
```

same as `count(*)` if no nulls

We probably want:

```
SELECT count(DISTINCT product)
FROM   Purchase
WHERE  price > 4.99
```

Loading Data into SQLite

```
>sqlite3 lecture04

sqlite> create table Purchase
        (pid int primary key,
         product text,
         price float,
         quantity int,
         month varchar(15));

sqlite> -- download data.txt
sqlite> .import lec04-data.txt Purchase
```

Specify a filename where the database will be stored

Other DBMSs have other ways of importing data

Comment about SQLite

- Cannot load NULL values such that they are actually loaded as null values

- So we need to use two steps:
  - Load null values using some type of special value
  - Update the special values to actual null values

```
update Purchase
  set price = null
  where price = 'null'
```

not string

string

Aggregates Example

See: lec04-sql-aggregates.sql
Use: lec04-data.txt

More Examples

What do they mean ?

```
SELECT  Sum(price * quantity)
FROM    Purchase
```

Total Revenue

```
SELECT  Sum(price * quantity)
FROM    Purchase
WHERE   product = 'bagel'
```

Revenue from Bagels