Introduction to Data Management CSE 344

Lecture 3: SQL Basics

Friday June 23

Announcements

- WQ1 due on Sunday (night
  - Any issues?
- HW1 due on next Tuesday (June 27)
- Office Hours
  - Moving to 2nd floor breakout

Announcements

- WQ1 due on Sunday night
  - Any issues?

Trevor : Monday 10:00 – 12:00

Ryan : Tuesday 11:30 – 12:30

Rob : Friday 1:00 – 2:00

Trevor : Wednesday 11:00 – 1:00 (CSE 220)

- – Moving to 2nd floor breakout

Review

- Relational data model
  - Instance and schema
- SQL for manipulating relational data
  - Create tables
  - Retrieve records from tables
  - Declare keys and foreign keys

Review

- SQL is declarative
  - Say what you want not how to do it
- Tables are FLAT
  - No nested attributes
- Tables DO NOT prescribe how they are implemented / stored on disk
  - This is called **physical data independence**

Relation Schema

- Names and types form part of the table “**schema**”:

```
Company(cname, country, no_employees, for_profit)
```

PK

```
Company(cname: varchar(30), country: char(20),
        no_employees: int, for_profit: char(1))
```

- Instance

cname	country	no_employees	for_profit
Canon	Japan	50000	Y
Hitachi	Japan	30000	Y

Adding Attributes

cname	country	no_employees	for_profit	Prods
Canon	Japan	50000	Y	(x, y, z)
Hitachi	Japan	30000	Y	(a, b, c)

- Let's add a list of product that each company produces
  - How? Recall that tables are flat!

Foreign Keys

- A column (or columns) whose value is a key of another table **(Must be unique!)**
  - i.e., a reference to another row in another table

Foreign Keys

- A column (or columns) whose value is a key of another table **(Must be unique!)**
  - i.e., a reference to another row in another table

cname	country	no_employees	for_profit
Canon	Japan	50000	Y
Hitachi	Japan	30000	Y

```
Product(pname, price, category,
        mname, mcountry,
        FOREIGN KEY (mname,mcountry)
        REFERENCES Company(cname,country)
)
```

New Relation

Foreign Keys

```
Company(cname, country, no_employees, for_profit)
```

cname	country	no_employees	for_profit
Canon	Japan	50000	Y
Hitachi	Japan	30000	Y

```
Product( pname, price, category, cname, country,
      FOREIGN KEY (cname,ccountry)
      REFERENCES Company(cname,country) )
```

pname	price	category	cname	country
SingleTouch	149.99	photography	**Canon**	**Japan**
AC	300	Appliance	**Hitachi**	**Japan**

Best Practice: Use Integer Primary Key

cid	cname	country	no_employees	for_profit
1	Canon	Japan	50000	Y
2	Hitachi	Japan	30000	Y

```
Product(pid INTEGER PRIMARY KEY, pname, price, category,
        cid REFERENCES Company.cid)
```

pid	pname	price	category	cid
1	SingleTouch	149.99	photography	**1**
2	AC	300	Appliance	**2**

"All problems in computer science can be solved by another level of indirection"

A Note On Multiple Keys

```
Company(cname: varchar(30) PRIMARY KEY NOT NULL,
          country: char(20),
           no_employees: int,
          for_profit: char(1)   NOT NULL
);
```

A Note On Multiple Keys

```
Company(cname: varchar(30) PRIMARY KEY NOT NULL,
         country: char(20),
          no_employees: int,
         for_profit: char(1)
);
```

goes away

```
 Company(cname: varchar(30) NOT NULL,
          country: char(20) NOT NULL,
           no_employees: int,
          for_profit: char(1),
          PRIMARY KEY (cname, country)
 );
```

added

Secondary keys

Same for UNIQUE and FOREIGN KEY

Today

- SQL Basics
  - Selection
  - Projection
  - Ordering
  - Joins

Demo

Setup Database

Selections in SQL

- Condition on the WHERE clause to filter returned tuples.

```
SELECT *
FROM   Product
WHERE  price > 100.0
```

<   <=   >   >=

==   !=   <>   IS   IS NOT   IN   LIKE

GLOB   MATCH   REGEXP

AND

OR

Demo

Selection

Projections in SQL

- What does the * mean in SELECT *
  - Shortcut for ALL attributes
  - What if we only want a few?

```
SELECT category
FROM   Product
```

- Can combine with selection (build complex queries)

```
SELECT category
FROM   Product
WHERE  price > 100.0
```

DISTINCT and ORDER BY

- Query results do not have to be relations
  - i.e., they can have duplicate rows
  - remove them using DISTINCT

- Result order is normally unspecified
  - choose an order using ORDER BY
  - e.g., ORDER BY country, cname
  - e.g., ORDER BY price ASC, pname DESC

Demo

Projection, Order, Distinct

Complex Queries

cname	country	no_employees	for_profit
GizmoWorks	USA	20000	Y
Hitachi	Japan	30000	Y

pname	price	category	cname $\rightarrow$ Key
SuperGizmo	250.00	Gadget	GizmoWorks
AC	300	Appliance	Hitachi

How do we get all products made in Japan?

Need information from BOTH tabels

Joins in SQL

```
Product(pname, price, category, manufacturer)
Company(cname, country)
```

```
SELECT pname, price
FROM   Product, Company   two tables
               manufacture
WHERE  Product.pname = Company.cname AND
       country='Japan' AND price < 150
```

- What does this query do?

Retrieve all Japanese products
that cost < $150

Joins in SQL

3×3

pname	price	cname
MultiTouch	199.99	Canon
SingleTouch	49.99	Canon
SuperGizmo	250.00	GizmoWorks

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

```
SELECT pname, price
FROM   Product, Company
```

where P.cname = C.cname

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

A) 3 B) 6 [✓ C) 9] D) 27

Joins in SQL

Retrieve all Japanese companies that manufacture products less than $100

pname	price	cname
MultiTouch	199.99	Canon
SingleTouch	49.99	Canon
SuperGizmo	250.00	GizmoWorks

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

```
SELECT pname, price
FROM   Product, Company
```

Joins in SQL

```
Product(pname, price, category, cname)
Company(cname, country)
```

pname	price	manufacturer
MultiTouch	199.99	Canon
SingleTouch	49.99	Canon
SuperGizmo	250.00	GizmoWorks

cname	country
GizmoWorks	USA
Canon	Japan

```
SELECT DISTINCT cname
FROM   Product, Company
WHERE  country='Japan' AND price < 100.0
       AND Product.cname = Company.cname
```

Aliases

```
SELECT country, pname, price
FROM Company LEFT JOIN Product
ON Company.cname = Product.cname
```

=

```
SELECT country, pname, price
FROM Company as c LEFT JOIN Product as p
ON c.cname = p.cname
```

Joins in SQL

- This query is called an inner join
  - Each row in the result **must come from both tables in the join**

```
SELECT DISTINCT cname
FROM   Product
INNER JOIN Company on Company.cname = Product.cname
WHERE  country='USA' AND category = 'gadget'
```

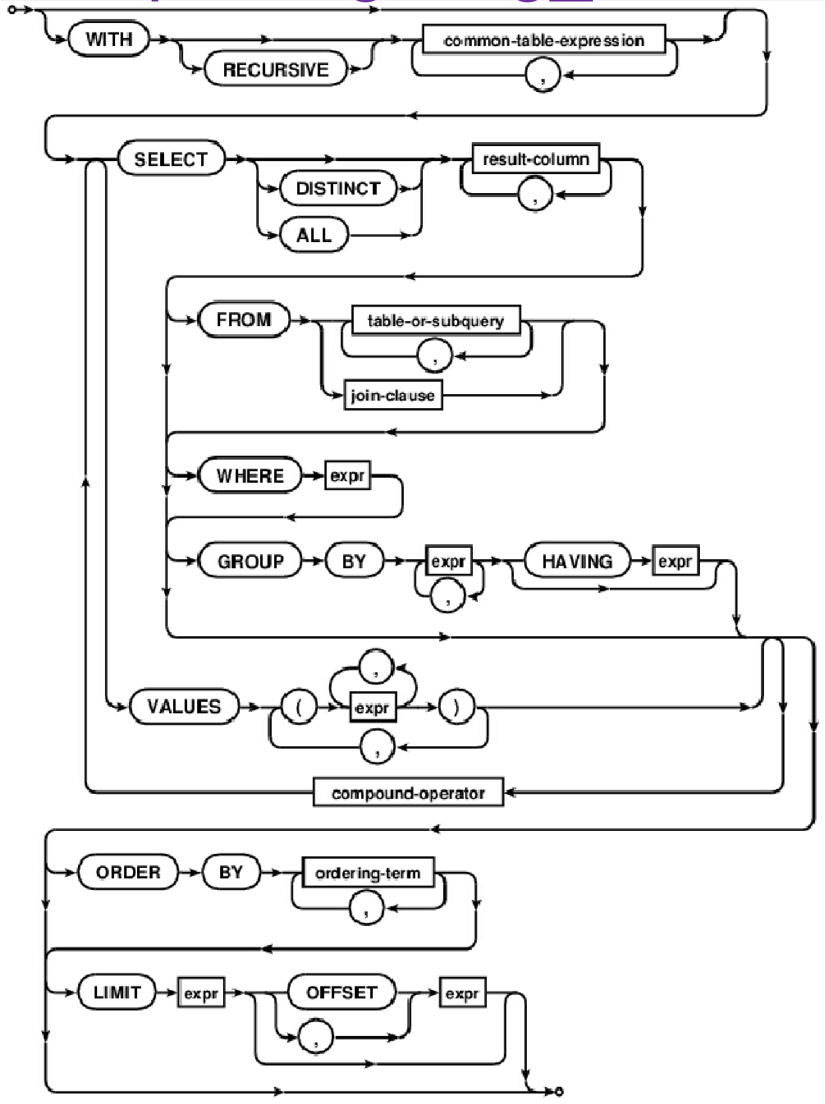
- What happens if a company makes no products?
  - Not returned in the results

Today

- SQL Basics
  - Selection
    - WHERE clause with condition
  - Projection
    - Field List or *
  - Ordering
  - Joins
    - Inner

SQLite SELECT

https://sqlite.org/lang_select.html


WITH
RECURSIVE
common-table-expression
,
SELECT
DISTINCT
ALL
result-column
,
FROM
table-or-subquery
,
join-clause
WHERE
expr
GROUP
BY
expr
,
HAVING
expr
VALUES
(
expr
,
)
,
compound-operator
ORDER
BY
ordering-term
,
LIMIT
expr
OFFSET
,
expr