Database Management Systems

Lecture 3: SQL Basics
(Inner) Joins

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

for t1 in x1:
  for t2 in x2:
    ...
    for tm in xm:
      if Cond(t1, t2, ...):
        output(t1.a1, t2.a2, ... tm.am)
(Inner) Joins

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

```python
for t1 in x1:
    for t2 in x2:
        ...
        for tm in xm:
            if Cond(t1, t2, ...):
                output(t1.a1, t2.a2, ... tm.am)
```
(Inner) joins

Company(\texttt{cname}, country)
Product(\texttt{pname}, price, category, manufacturer)
  – manufacturer is foreign key

\begin{verbatim}
SELECT DISTINCT cname
FROM Product, Company
WHERE country = 'USA' AND category = 'gadget' AND manufacturer = cname
\end{verbatim}
(Inner) joins

```
SELECT DISTINCT cname
FROM Product, Company
WHERE country = 'USA' AND category = 'gadget' AND manufacturer = cname
```
(Inner) joins

```
SELECT DISTINCT cname
FROM Product, Company
WHERE country = 'USA' AND category = 'gadget' AND manufacturer = cname
```
(Inner) joins

```sql
SELECT DISTINCT cname
FROM Product, Company
WHERE country = 'USA' AND category = 'gadget' AND manufacturer = cname
```
(Inner) joins

```sql
SELECT DISTINCT cname
FROM Product, Company
WHERE country = 'USA' AND category = 'gadget' AND manufacturer = cname
```
(Inner) joins

```
SELECT DISTINCT cname
FROM Product, Company
WHERE country = 'USA' AND category = 'gadget' AND manufacturer = cname
```
(Inner) joins

```
SELECT DISTINCT cname
FROM Product, Company
WHERE country = 'USA' AND category = 'gadget' AND manufacturer = cname
```
(Inner) joins

```sql
SELECT DISTINCT cname
FROM Product, Company
WHERE country = 'USA' AND category = 'gadget' AND manufacturer = cname
```

```sql
SELECT DISTINCT cname
FROM Product JOIN Company
ON country = 'USA' AND category = 'gadget' AND manufacturer = cname
```
## Tuple variables

```sql
SELECT DISTINCT cname
FROM Product P1, Product P2, Company
WHERE country = 'USA' AND P1.category = 'gadget' AND P2.category = 'photo' AND P1.manufacturer = cname AND P2.manufacturer = cname;
```

<table>
<thead>
<tr>
<th>P1 (Product)</th>
<th>P2 (Product)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>pname</strong></td>
<td><strong>pname</strong></td>
</tr>
<tr>
<td><strong>category</strong></td>
<td><strong>category</strong></td>
</tr>
<tr>
<td><strong>manufacturer</strong></td>
<td><strong>manufacturer</strong></td>
</tr>
<tr>
<td>Gizmo</td>
<td>Gizmo2</td>
</tr>
<tr>
<td>gadget</td>
<td>photo</td>
</tr>
<tr>
<td>GizmoWorks</td>
<td>GizmoWorks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cname</strong></td>
</tr>
<tr>
<td>GizmoWorks</td>
</tr>
</tbody>
</table>
### Tuple variables

```
SELECT DISTINCT cname
FROM Product P1, Product P2, Company
WHERE country = 'USA' AND P1.category = 'gadget' AND P2.category = 'photo' AND P1.manufacturer = cname AND P2.manufacturer = cname;
```

<table>
<thead>
<tr>
<th>P1 (Product)</th>
<th>P2 (Product)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>pname</strong></td>
<td><strong>category</strong></td>
</tr>
<tr>
<td>Gizmo</td>
<td>gadget</td>
</tr>
<tr>
<td>Gizmo2</td>
<td>photo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cname</strong></td>
</tr>
<tr>
<td>GizmoWorks</td>
</tr>
</tbody>
</table>
### Tuple variables

**SELECT DISTINCT** `cname`  
**FROM**  
Product P1, Product P2, Company  
**WHERE**  
country = 'USA' AND P1.category = 'gadget' AND P2.category = 'photo' AND P1.manufacturer = `cname` AND P2.manufacturer = `cname`;

<table>
<thead>
<tr>
<th>P1 (Product)</th>
<th>P2 (Product)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>pname</strong></td>
<td><strong>pname</strong></td>
</tr>
<tr>
<td><strong>category</strong></td>
<td><strong>category</strong></td>
</tr>
<tr>
<td><strong>manufacturer</strong></td>
<td><strong>manufacturer</strong></td>
</tr>
<tr>
<td>Gizmo</td>
<td>Gizmo2</td>
</tr>
<tr>
<td>gadget</td>
<td>photo</td>
</tr>
<tr>
<td>GizmoWorks</td>
<td>GizmoWorks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cname</strong></td>
</tr>
<tr>
<td>GizmoWorks</td>
</tr>
</tbody>
</table>
Outer joins

Product(name, category)
Purchase(prodName, store) -- prodName is foreign key

An “inner join”:

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

Same as:

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

But some Products are not listed! Why?
Outer joins

Product(name, category)
Purchase(prodName, store) -- prodName is foreign key

If we want to include products that never sold, then we need an “outerjoin”:

```
SELECT Product.name, Purchase.store
FROM Product LEFT OUTER JOIN Purchase ON
  Product.name = Purchase.prodName
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ProdName</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
</tr>
<tr>
<td>Camera</td>
<td>Ritz</td>
</tr>
<tr>
<td>Camera</td>
<td>Wiz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
</tr>
<tr>
<td>Camera</td>
<td>Ritz</td>
</tr>
<tr>
<td>Camera</td>
<td>Wiz</td>
</tr>
<tr>
<td>OneClick</td>
<td>NULL</td>
</tr>
</tbody>
</table>
Outer Joins

• Left outer join:
  – Include the left tuple even if there’s no match
• Right outer join:
  – Include the right tuple even if there’s no match
• Full outer join:
  – Include both left and right tuples even if there’s no match
Aggregation in SQL

>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
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 apply to a single attribute
Aggregates and NULL Values

Null values are not used in aggregates

```sql
insert into Purchase
values(12, 'gadget', NULL, NULL, 'april')
```

Let's try the following

```sql
select count(*) from Purchase
select count(quantity) from Purchase
select sum(quantity) from Purchase
select sum(quantity) from Purchase
where quantity is not null;
```
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
```
More Examples

\[ \text{SELECT } \text{Sum(price } \times \text{ quantity)} \text{ FROM Purchase} \]

\[ \text{SELECT } \text{Sum(price } \times \text{ quantity)} \text{ FROM Purchase WHERE product = 'bagel'} \]

What do they mean?
Simple Aggregations

**Purchase**

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>Bagel</td>
<td>1.50</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>0.5</td>
<td>50</td>
</tr>
<tr>
<td>Banana</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Banana</td>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>

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

90 (= 60 + 30)
### Simple Aggregations

#### Purchase

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>Bagel</td>
<td>1.50</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>0.5</td>
<td>50</td>
</tr>
<tr>
<td>Banana</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Banana</td>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>

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

90 (= 60+30)
Grouping and Aggregation

Purchase(product, price, quantity)

Find total quantities for all sales over $1, by product.

```
SELECT product, Sum(quantity) AS TotalSales
FROM Purchase
WHERE price > 1
GROUP BY product
```

Let’s see what this means…
Grouping and Aggregation

1. Compute the **FROM** and **WHERE** clauses.

2. Group by the attributes in the **GROUP BY**

3. Compute the **SELECT** clause:
   grouped attributes and aggregates.

FWGS
1&2. FROM-WHERE-GROUPBY

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>Bagel</td>
<td>1.50</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>0.5</td>
<td>50</td>
</tr>
<tr>
<td>Banana</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Banana</td>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>

WHERE price > 1

FWGS
3. SELECT

```
SELECT product, Sum(quantity) AS TotalSales
FROM Purchase
WHERE price > 1
GROUP BY product
```

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>Bagel</td>
<td>1.50</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>0.5</td>
<td>50</td>
</tr>
<tr>
<td>Banana</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Banana</td>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product</th>
<th>TotalSales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>40</td>
</tr>
<tr>
<td>Banana</td>
<td>20</td>
</tr>
</tbody>
</table>
Other Examples

Compare these two queries:

```
SELECT product, count(*)
FROM Purchase
GROUP BY product
```

```
SELECT month, count(*)
FROM Purchase
GROUP BY month
```

```
SELECT product,
    sum(quantity) AS SumQuantity,
    max(price) AS MaxPrice
FROM Purchase
GROUP BY product
```

What does it mean?
Need to be Careful…

SELECT product, max(quantity) 
FROM Purchase 
GROUP BY product

SELECT product, quantity 
FROM Purchase 
GROUP BY product

sqlite is WRONG on this query. (why?)

Advanced DBMS (e.g. SQL Server) gives an error