Lecture 3 Additional Slides

CSE 344, Winter 2014

Sudeepa Roy

- Note:

  These slides mostly contain the same material as the lecture notes, and were used as a substitute for a whiteboard in class.

- Also please go over all the sql commands and the comments in lecture02 and lecture03 notes and let us know if any of the queries or explanations is not clear.

- A few very nice observations from you in class!
  - Check if you using the right quote ’ and not an automatic correction by an editor ‘
  - Check if the font size is proper in sqlite window or whether it is truncating any field: “photography” vs. “photograph”

Ex. In SqLite

Product

pname	price	category	manufacturer
Gizmo	19.99	gadget	GizmoWorks
PowerGizmo	29.99	gadget	GizmoWorks
SingleTouch	149.99	photography	Canon
MultiTouch	199.99	photography	Hitachi
SuperGizmo	49.99	gadget	Hitachi

Company

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan
Apple	USA

manufacturer
references **cname**

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

Selection queries

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

- Select a subset of rows
- Condition specified by WHERE clause
- Ex:
  - select *
  - From Product
  - where price > 100.0;
- Ex:
  - select *
  - From Product
  - Where pname like '%e%';

Projection queries

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

- Keep a subset of the attributes/columns
- Attributes specified by SELECT clause
- Ex:
  - select price, category
  - from Product;

DISTINCT

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

- Duplicates:
  - select category
  - from Product;
- Eliminates duplicates
  - select distinct category
  - from Product;
- Once again, set vs. bag

Order By

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

- order alphabetically by name:
- order by price descending
- order by manufacturer, then price descending

BASIC SQL Query Evaluation

SELECT <attr>

FROM <reln>

WHERE <condn>

(optional)

ORDER BY <attr2>
<asc/desc>

Sequence in evaluation

1. FROM: for each tuple in <reln>
2. WHERE: apply <condn>
3. SELECT: <attr>

ORDER BY and DISTINCT - 1

SELECT <attr>

FROM <reln>

WHERE <condn>

(optional)

ORDER BY <attr2>
<asc/desc>

Sequence in evaluation

1. FROM: for each tuple in <reln>
2. WHERE: apply <condn>
3. ORDER BY: <attr2>
4. SELECT: <attr>

ORDER BY and DISTINCT - 2

- What happens if we order on an attribute that we do NOT return ?

- First, let's try:
  - select * from Product order by manufacturer;

- Now, let's try:
  - select category from Product order by manufacturer;

- What happens if we also do DISTINCT ?
  - select distinct category from Product order by manufacturer;

- In SQL, all attributes in ORDER BY must appear in SELECT if DISTINCT is used, should have been an error. Sqlite does not enforce this (another alert).

JOINS

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

- What should the following query return?
  - select pname, price
  - from Product P, Company C
  - where P.**manufacturer=C.cname** and country='Japan' and price < 150;

Join predicate

- Your answer in class ☺
- Single touch, 149.99
- Supergizmo, 49.99

Ex. 1: Retrieve all American company names that manufacture products in the 'gadget' category

- Product(pname, price, category, manufacturer)
- Company(cname, country)
- **Your answer in class ☺**

```
SELECT distinct P.manufacturer
FROM  Product P, Company C
WHERE P.manufacturer = C.cname
And C.country = 'USA'
And P.category = 'gadget'
```

Ex. 2: Retrieve all Japanese company names that manufacture products in both the 'gadget' and the photography category

- Product(pname, price, category, manufacturer)
- Company(cname, country)
- **Your answer in class ☺**

```
SELECT distinct cname
FROM Product P1,
Company, Product P2
WHERE country = 'Japan' AND
P1.category = 'gadget' AND
P2.category = 'photography'
And P1.manufacturer = cname
AND P2.manufacturer = cname
```

Note:

1. The third condition P1.manufacturer = P2.manufacturer is not needed
2. We could replace the last condition by P1.manufacturer = P2.manufacturer
3. i.e. only two equality checks are needed and not three
4. Why? See next 4 slides and think!

Join as a cartesian product followed by selection and projection (NEW SLIDE-1)

- You should think this way when writing complicated SQL queries or finding answers to a given query.
- More on this when we learn Relational Algebra (Lecture 9)
- Consider this example.

```
select R.a
from R, S, T
where R.a=S.a AND S.a <> b
```

R

A
1
2
3

S

A
2
3
4

T

B
1
2
4

Join as a cartesian product followed by selection and projection (NEW SLIDE-2)

R

A
1
2

S

A
2
3

T

B
3

```
select R.a
from R, S, T
where R.a=S.a AND S.a <> b
```

R.A	S.A	T.B
1	2	3
1	3	3
2	2	3
2	3	3

$R \times S \times T$

- STEP 1: The cartesian product of these tables will be computed (see the nested loop semantics later)
- NOTE: DBMSs will almost never evaluate queries in this inefficient way. More on this when we learn query plans

Join as a cartesian product followed by selection and projection (NEW SLIDE-3)

R

A
1
2

S

A
2
3

T

B
3

```
select R.a
from R, S, T
where R.a=S.a AND S.a <> b
```

- STEP 2: Apply condition in WHERE clause
- Only one tuple satisfies the condition (highlighted)

R.A	S.A	T.B
1	2	3
1	3	3
2	**2**	**3**
2	3	3

$R \times S \times T$

Join as a cartesian product followed by selection and projection (NEW SLIDE-4)

R

A
1
2

S

A
2
3

T

B
3

```
select R.a
from R, S, T
where R.a=S.a AND S.a <> b
```

R.A	S.A	T.B
1	2	3
1	3	3
2	**2**	**3**
2	3	3

$R \times S \times T$

- STEP 3: Now project on to the attributes in SELECT clause
- Final answer:

  2

Now think!

- why the other two answers for Ex 2 that we considered in class did not work
  - (Japanese companies for both gadgets and photography)

- Option 1: country = 'Japan' and (category = 'gadget' OR category = 'photography')
  - Ans: If any of these two categories exists in the cartesian product with country = Japan, it will be returned
- Option 2: country = 'Japan' and (category = 'gadget' AND category = 'photography')
  - Ans: Category for any tuple in the cartesian product cannot be both gadget and photography

Joins may introduce duplicates

- Try:
  - select country
  - from Product, Company
  - where manufacturer=cname and category='gadget';

- Easy fix: USE DISTINCT
  - select distinct country
  - from Product, Company
  - where manufacturer=cname and category='gadget';

Ex. 3: Find all countries that manufacture both a product under $25 and a product over $25 (Aliases)

- Product(pname, price, category, manufacturer)
- Company(cname, country)
- Answer :
- (First try yourself and then see the answer in the notes.)

JOINS: Nested Loop Semantics for SQL

- Query:
  - SELECT a1, a2, ..., ak
  - FROM R1 AS x1, R2 AS x2, ..., Rm AS xm
  - WHERE Cond
- Semantics:
  - for a1 in R1 do
  - for a2 in R2 do
  - for a3 in R3 do –
  - ...
  - for an in Rm do
  - if Cond(a1, …ak) is true
  - then output(a1,...,ak)

Although the quer processor will ALMOST NEVER evaluate the query this way!

FROM

WHERE

SELECT

What does this query compute?

R

A
1
2
3

S

A
2
3
4

T

A
1
2
4

```
select distinct R.a
from R, S
where R.a=S.a;
```

What does this query compute?

R

A
1
2
3

S

A
2
3
4

T

A
1
2
4

ANS: R intersects S

```
select distinct R.a
from R, S
where R.a=S.a;
```

What does this query compute?

R

A
1
2
3

S

A
2
3
4

T

A
1
2
4

select distinct T.a

from R, S, T

where R.a=T.a or S.a = T.a

What does this query compute?

R

A
1
2
3

S

A

T

A
1
2
4

- you might think it is: (R union S) intersect T
- but think again!
- what happens if say S = ϕ, i.e. no tuples in S
- The result should have been (R intersect T)
- But we get empty set.
- Nested loop semantics explains this!

select distinct T.a

from R, S, T

where R.a=T.a or S.a = T.a

Ans: the query returns
(R union S) intersect T
if R,S are non-empty.
otherwise it returns the empty set

NULL In SqLite

Product

pname	price	category	manufacturer
Gizmo	19.99	gadget	GizmoWorks
PowerGizmo	29.99	gadget	GizmoWorks
SingleTouch	149.99	photography	Canon
MultiTouch	199.99	photography	Hitachi
SuperGizmo	49.99	gadget	Hitachi
iPad 5	**NULL**	**gadget**	**Apple**

Company

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan
Apple	USA

SELECT gadegts with price < 25 and >=25
Ipad is nowhere!

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

Conditions involving NULL

- We need to evaluate in SQL conditions like this:
- (price < 25) and (category = 'gadget') or (manufacturer = 'Apple')
- Suppose price = 19, category = NULL, and manufacturer = NULL
- Is the predicate true or false?

3-valued logic

- FALSE = 0 E.g. price<25 is FALSE when price=99
- UNKNOWN = 0.5 E.g. price<25 is UNKNOWN when price=NULL
- TRUE = 1 E.g. price<25 is TRUE when price=19

- C1 AND C2 means min(C1,C2)
- C1 OR C2 means max(C1,C2)
- not C means 1-C

Compute the truth value

- (price < 25) and (category = 'gadget') or (manufacturer = 'Apple')
- Suppose price = 19, category = NULL, and manufacturer = NULL

- Answer:

Output of a query for NULL

- The rule for SELECT ... FROM ... WHERE C is the following:
  - if C = TRUE then include the row in the output
  - if C = FALSE or C = unknown then do not include it

Outer JOIN

More on outer join in Lec 4 (Monday)

- JOIN = INNER JOIN
- LEFT OUTER JOIN:
  - include everything on the left, fill in the right part with NULL values
- Similarly (FULL)/RIGHT OUTER JOIN

pname	price	category	manufacturer
Gizmo	19.99	gadget	GizmoWorks
PowerGizmo	29.99	gadget	GizmoWorks
SingleTouch	149.99	photography	Canon
MultiTouch	199.99	photography	Hitachi
SuperGizmo	49.99	gadget	Hitachi
iPad 5	NULL	gadget	Apple

Product

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan
Apple	USA
Google	**USA**

Company