

CSE 444: Database Internals

Lecture 2 Review of the Relational Model

CSE 444 - Spring 2019

1

Announcements

- Lab 1 part 1 is due on Monday at 11pm
 - Lab 1 in full is due on April 17th
 - "git commit -a" and "git push" often!
- HW1 is due next week on Friday
 - Print out PDF and hand in completed version
- 544M first paper review is also due next week
 - Can hand in the report to me in class
 - Deadlines are flexible for graduate readings

CSE 444 - Spring 2019

2

Agenda

- Review Relational Model
- Review Queries (will skip some slides)
 - Relational Algebra
 - SQL
- Review translation SQL \rightarrow RA
 - Needed for HW1

CSE 444 - Spring 2019

3

Database/Relation/Tuple

- A **Database** is collection of relations
- A **Relation** R is subset of $S_1 \times S_2 \times \dots \times S_n$
 - Where S_i is the domain of attribute i
 - n is number of attributes of the relation
 - A relation is a set of tuples
- A **Tuple** t is an element of $S_1 \times S_2 \times \dots \times S_n$
 - Other names: relation = **table**; tuple = **row**

CSE 444 - Spring 2019

4

Discussion

- **Rows** in a relation:
 - Ordering immaterial (a relation is a set)
 - All rows are distinct – **set semantics**
 - Query answers may have duplicates – **bag semantics**
- **Columns** in a tuple:
 - Ordering is significant
 - Applications refer to columns by their names
- **Domain** of each column is a primitive type

Data independence!

CSE 444 - Spring 2019

5

Schema

- **Relation schema**: describes column heads
 - Relation name
 - Name of each field (or column, or attribute)
 - Domain of each field
- **Degree (or arity) of relation**: # attributes
- **Database schema**: set of all relation schemas

CSE 444 - Spring 2019

6

Instance

- **Relation instance**: concrete table content
 - Set of tuples (also called records) matching the schema
- **Cardinality of relation instance**: # tuples
- **Database instance**: set of all relation instances

CSE 444 - Spring 2019

7

What is the schema? What is the instance?

Supplier

sno	sname	scity	sstate
1	s1	city 1	WA
2	s2	city 1	WA
3	s3	city 2	MA
4	s4	city 2	MA

CSE 444 - Spring 2019

8

What is the schema? What is the instance?

Relation schema

Supplier(sno: integer, sname: string, scity: string, sstate: string)

Supplier

sno	sname	scity	sstate
1	s1	city 1	WA
2	s2	city 1	WA
3	s3	city 2	MA
4	s4	city 2	MA

} instance

CSE 444 - Spring 2019

9

Integrity Constraints

- Condition specified on a database schema
- Restricts data that can be stored in db instance
- DBMS enforces integrity constraints
 - Ensures only legal database instances exist
- Simplest form of constraint is domain constraint
 - Attribute values must come from attribute domain

CSE 444 - Spring 2019

10

Key Constraints

- **Super Key**: “set of attributes that functionally determines all attributes”
- **Key**: Minimal super-key; a.k.a. “candidate key”
- **Primary key**: One minimal key can be selected as primary key

CSE 444 - Spring 2019

11

Foreign Key Constraints

- A relation can refer to a tuple in another relation
- **Foreign key**
 - Field that refers to tuples in another relation
 - Typically, this field refers to the primary key of other relation
 - Can pick another field as well

CSE 444 - Spring 2019

12

Key Constraint SQL Examples

```
CREATE TABLE Part (
    pno integer,
    pname varchar(20),
    psize integer,
    pcolor varchar(20),
    PRIMARY KEY (pno)
);
```

CSE 444 - Spring 2019

13

Key Constraint SQL Examples

```
CREATE TABLE Supply(
    sno integer,
    pname varchar(20),
    pno integer,
    qty integer,
    price integer
);
```

```
CREATE TABLE Part (
    pno integer,
    pname varchar(20),
    psize integer,
    pcolor varchar(20),
    PRIMARY KEY (pno)
);
```

CSE 444 - Spring 2019

14

Key Constraint SQL Examples

```
CREATE TABLE Supply(
    sno integer,
    pno integer,
    qty integer,
    price integer,
    PRIMARY KEY (sno,pno)
);
```

```
CREATE TABLE Part (
    pno integer,
    pname varchar(20),
    psize integer,
    pcolor varchar(20),
    PRIMARY KEY (pno)
);
```

CSE 444 - Spring 2019

15

Key Constraint SQL Examples

```
CREATE TABLE Supply(
    sno integer,
    pno integer,
    qty integer,
    price integer,
    PRIMARY KEY (sno,pno),
    FOREIGN KEY (sno) REFERENCES Supplier,
    FOREIGN KEY (pno) REFERENCES Part
);
```

```
CREATE TABLE Part (
    pno integer,
    pname varchar(20),
    psize integer,
    pcolor varchar(20),
    PRIMARY KEY (pno)
);
```

CSE 444 - Spring 2019

16

Key Constraint SQL Examples

```
CREATE TABLE Supply(
    sno integer,
    pno integer,
    qty integer,
    price integer,
    PRIMARY KEY (sno,pno),
    FOREIGN KEY (sno) REFERENCES Supplier
        ON DELETE NO ACTION,
    FOREIGN KEY (pno) REFERENCES Part
        ON DELETE CASCADE
);
```

```
CREATE TABLE Part (
    pno integer,
    pname varchar(20),
    psize integer,
    pcolor varchar(20),
    PRIMARY KEY (pno)
);
```

CSE 444 - Spring 2019

17

General Constraints

- Table constraints serve to express complex constraints over a single table

```
CREATE TABLE Part (
    pno integer,
    pname varchar(20),
    psize integer,
    pcolor varchar(20),
    PRIMARY KEY (pno),
    CHECK ( psize > 0 )
);
```

Note: Also possible to create constraints over many tables
Best to use database triggers for that purpose

CSE 444 - Spring 2019

18

Relational Query Languages

CSE 444 - Spring 2019

19

Relational Query Language

- **Set-at-a-time:**
 - Query inputs and outputs are relations
- Two variants of the query language:
 - Relational algebra: specifies order of operations
 - Relational calculus / SQL: declarative

CSE 444 - Spring 2019

20

Note

- We will go very quickly in class over the Relational Algebra and SQL
- Please review at home:
 - Read the slides that we skipped in class
 - Review material from 344 as needed

CSE 444 - Spring 2019

21

Relational Algebra

- **Queries specified in an operational manner**
 - A query gives a step-by-step procedure
- **Relational operators**
 - Take one or two relation instances as argument
 - Return one relation instance as result
 - Easy to compose into **relational algebra expressions**

CSE 444 - Spring 2019

22

Five Basic Relational Operators

- **Selection:** $\sigma_{\text{condition}}(S)$
 - Condition is Boolean combination (\wedge, \vee) of atomic predicates ($<, <=, =, \neq, >=, >$)
- **Projection:** $\pi_{\text{list-of-attributes}}(S)$
- **Union** (\cup)
- **Set difference** ($-$),
- **Cross-product/cartesian product** (\times),
Join: $R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$

CSE 444 - Spring 2019

23

```
Supplier(sno, sname, scity, sstate)
Supply(sno, pno, qty, price)
Part(pno, pname, psize, pcolor)
```

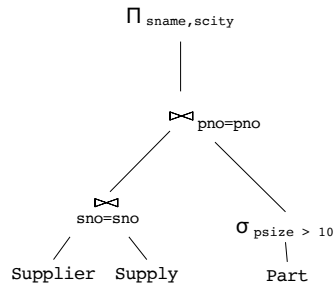
Logical Query Plans

CSE 444 - Spring 2019

24

Supplier(sno, sname, scity, sstate)
 Supply(sno, pno, qty, price)
 Part(pno, pname, psize, pcolor)

Logical Query Plans



What does this query compute?

CSE 444 - Spring 2019

25

Selection & Projection Examples

Patient

no	name	zip	disease
1	p1	98125	flu
2	p2	98125	heart
3	p3	98120	lung
4	p4	98120	heart

$\pi_{zip, disease}(Patient)$

zip	disease
98125	flu
98125	heart
98120	lung
98120	heart

$\sigma_{disease='heart'}(Patient)$

no	name	zip	disease
2	p2	98125	heart
4	p4	98120	heart

$\pi_{zip}(\sigma_{disease='heart'}(Patient))$

zip
98120
98125

CSE 444 - Spring 2019

26

Cross-Product Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

Voters V

name	age	zip
p1	54	98125
p2	20	98120

$P \times V$

P.age	P.zip	disease	name	V.age	V.zip
54	98125	heart	p1	54	98125
54	98125	heart	p2	20	98120
20	98120	flu	p1	54	98125
20	98120	flu	p2	20	98120

CSE 444 - Spring 2019

27

Different Types of Join

- **Theta-join:** $R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$
 - Join of R and S with a join condition θ
 - Cross-product followed by selection θ
- **Equijoin:** $R \bowtie_{\theta} S = \pi_A(\sigma_{\theta}(R \times S))$
 - Join condition θ consists only of equalities
 - Projection π_A drops all redundant attributes
- **Natural join:** $R \bowtie S = \pi_A(\sigma_{\theta}(R \times S))$
 - Equijoin
 - Equality on **all** fields with same name in R and in S

CSE 444 - Spring 2019

28

Theta-Join Example

AnonPatient P

age	zip	disease
50	98125	heart
19	98120	flu

Voters V

name	age	zip
p1	54	98125
p2	20	98120

$P \bowtie_{P.zip = V.zip \text{ and } P.age \leq V.age + 1 \text{ and } P.age \geq V.age - 1} V$

P.age	P.zip	disease	name	V.age	V.zip
19	98120	flu	p2	20	98120

CSE 444 - Spring 2019

29

Equijoin Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

Voters V

name	age	zip
p1	54	98125
p2	20	98120

$P \bowtie_{P.age=V.age} V$

age	P.zip	disease	name	V.zip
54	98125	heart	p1	98125
20	98120	flu	p2	98120

CSE 444 - Spring 2019

30

Natural Join Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

Voters V

name	age	zip
p1	54	98125
p2	20	98120

$P \bowtie V$

age	zip	disease	name
54	98125	heart	p1
20	98120	flu	p2

CSE 444 - Spring 2019

31

More Joins

- **Outer join**

- Include tuples with no matches in the output
- Use NULL values for missing attributes

- **Variants**

- Left outer join
- Right outer join
- Full outer join

CSE 444 - Spring 2019

32

Outer Join Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu
33	98120	lung

Voters V

name	age	zip
p1	54	98125
p2	20	98120

$P \bowtie V$

age	zip	disease	name
54	98125	heart	p1
20	98120	flu	p2
33	98120	lung	null

CSE 444 - Spring 2019

33

Example of Algebra Queries

Q1: Names of patients who have heart disease

$\pi_{\text{name}}(\text{Voter} \bowtie (\sigma_{\text{disease}=\text{'heart'}}(\text{AnonPatient})))$

CSE 444 - Spring 2019

34

More Examples

Relations

Supplier(sno, sname, scity, sstate)

Part(pno, pname, psize, pcolor)

Supply(sno, pno, qty, price)

Q2: Name of supplier of parts with size greater than 10

$\pi_{\text{sname}}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{\text{psize}>10}(\text{Part})))$

Q3: Name of supplier of red parts or parts with size greater than 10

$\pi_{\text{sname}}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{\text{psize}>10}(\text{Part}) \cup \sigma_{\text{pcolor}=\text{'red'}}(\text{Part})))$

(Many more examples in the book)

CSE 444 - Spring 2019

35

Extended Operators of Relational Algebra

- **Duplicate elimination (δ)**

- Since commercial DBMSs operate on multisets not sets

- **Aggregate operators (γ)**

- Min, max, sum, average, count

- **Grouping operators (γ)**

- Partitions tuples of a relation into "groups"
- Aggregates can then be applied to groups

- **Sort operator (τ)**

CSE 444 - Spring 2019

37

Structured Query Language: SQL

- Declarative query language, based on the relational calculus (see 344)
- Data definition language
 - Statements to create, modify tables and views
- Data manipulation language
 - Statements to issue queries, insert, delete data

CSE 444 - Spring 2019

38

SQL Query

Basic form: (plus many many more bells and whistles)

```
SELECT <attributes>
FROM   <one or more relations>
WHERE  <conditions>
```

CSE 444 - Spring 2019

39

```
Supplier(sno, sname, scity, sstate)
Supply(sno, pno, qty, price)
Part(pno, pname, psize, pcolor)
```

Quick Review of SQL

CSE 444 - Spring 2019

40

```
Supplier(sno, sname, scity, sstate)
Supply(sno, pno, qty, price)
Part(pno, pname, psize, pcolor)
```

Quick Review of SQL

```
SELECT DISTINCT z.pno, z.pname
FROM   Supplier x, Supply y, Part z
WHERE  x.sno = y.sno and y.pno = z.pno
       and x.scity = 'Seattle' and y.price < 100
```

What does
this query
compute?

CSE 444 - Spring 2019

41

```
Supplier(sno, sname, scity, sstate)
Supply(sno, pno, qty, price)
Part(pno, pname, psize, pcolor)
```

Quick Review of SQL

What about
this one?

```
SELECT z.pname, count(*) as cnt, min(y.price)
FROM   Supplier x, Supply y, Part z
WHERE  x.sno = y.sno and y.pno = z.pno
GROUP BY z.pname
```

CSE 444 - Spring 2019

42

Simple SQL Query

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

```
SELECT *
FROM   Product
WHERE  category='Gadgets'
```



PName	Price	Category	Manufacturer
Gizmo	\$19.99	Gadgets	GizmoWorks
Powergizmo	\$29.99	Gadgets	GizmoWorks

"selection"

CSE 444 - Spring 2019

43

Simple SQL Query

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

```
SELECT PName, Price, Manufacturer
FROM Product
WHERE Price > 100
```



"selection" and
"projection"

PName	Price	Manufacturer
Single Touch	\$149.99	Canon
MultiTouch	\$203.99	Hitachi

CSE 444 - Spring 2019

44

Details

- Case insensitive:
 - Same: SELECT Select select
 - Same: Product product
 - Different: 'Seattle' 'seattle'
- Constants:
 - 'abc' - yes
 - "abc" - no

CSE 444 - Spring 2019

45

Eliminating Duplicates

```
SELECT DISTINCT category
FROM Product
```

Category
Gadgets
Photography
Household

Compare to:

```
SELECT category
FROM Product
```

Category
Gadgets
Gadgets
Photography
Household

CSE 444 - Spring 2019

46

Ordering the Results

```
SELECT pname, price, manufacturer
FROM Product
WHERE category='gizmo' AND price > 50
ORDER BY price, pname
```

Ties are broken by the second attribute on the ORDER BY list, etc.

Ordering is ascending, unless you specify the DESC keyword.

CSE 444 - Spring 2019

47

Joins

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

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

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

CSE 444 - Spring 2019

48

Tuple Variables

Person(pname, address, worksfor)
Company(cname, address)

```
SELECT DISTINCT pname, address
FROM Person, Company
WHERE worksfor = cname
```

Which
address ?

```
SELECT DISTINCT Person.pname, Company.address
FROM Person, Company
WHERE Person.worksfor = Company.cname
```

```
SELECT DISTINCT x.pname, y.address
FROM Person AS x, Company AS y
WHERE x.worksfor = y.cname
```

49

Nested Queries

- **Nested query**
 - Query that has another query embedded within it
 - The embedded query is called a **subquery**
- Why do we need them?
 - Enables to refer to a table that must itself be computed
- Subqueries can appear in
 - **WHERE** clause (common)
 - **FROM** clause (less common)
 - **HAVING** clause (less common)

CSE 444 - Spring 2019

50

Subqueries Returning Relations

Company(name, city)
Product(pname, maker)
Purchase(id, product, buyer)

Return cities where one can find companies that manufacture products bought by Joe Blow

```
SELECT Company.city
FROM Company
WHERE Company.name IN
    (SELECT Product.maker
     FROM Purchase, Product
     WHERE Product.pname=Purchase.product
     AND Purchase.buyer = 'Joe Blow');
```

Subqueries Returning Relations

You can also use: s > **ALL** R
s > **ANY** R

EXISTS R
Product (pname, price, category, maker)

Find products that are more expensive than all those produced By "Gizmo-Works"

```
SELECT name
FROM Product
WHERE price > ALL (SELECT price
                   FROM Purchase
                   WHERE maker='Gizmo-Works')
```

Correlated Queries

Movie (title, year, director, length)

Find movies whose title appears more than once.

```
SELECT DISTINCT title
FROM Movie AS x
WHERE year <> ANY
    (SELECT year
     FROM Movie
     WHERE title = x.title);
```

Note (1) scope of variables (2) this can still be expressed as single SFW

CSE 444 - Spring 2019

53

Aggregation

```
SELECT avg(price)
FROM Product
WHERE maker="Toyota"
```

```
SELECT count(*)
FROM Product
WHERE year > 1995
```

SQL supports several aggregation operations:
sum, count, min, max, avg

Except count, all aggregations apply to a single attribute

CSE 444 - Spring 2019

54

Grouping and Aggregation

```
SELECT S
FROM R1, ..., Rn
WHERE C1
GROUP BY a1, ..., ak
HAVING C2
```

Conceptual evaluation steps:

1. Evaluate FROM-WHERE, apply condition C₁
2. Group by the attributes a₁, ..., a_k
3. Apply condition C₂ to each group (may have aggregates)
4. Compute aggregates in S and return the result

Read more about it in the book...

CSE 444 - Spring 2019

55

From SQL to RA

CSE 444 - Spring 2019

56

From SQL to RA

Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)

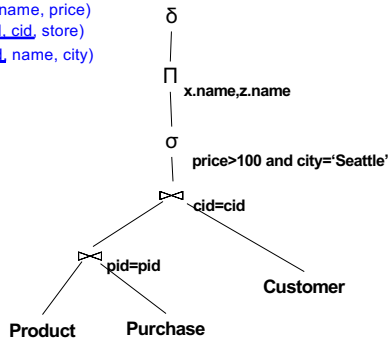
```
SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = y.cid and
      x.price > 100 and z.city = 'Seattle'
```

CSE 444 - Spring 2019

57

From SQL to RA

Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)

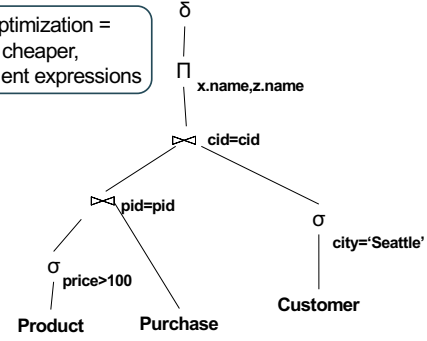


CSE 444 - Spring 2019

58

An Equivalent Expression

Query optimization =
finding cheaper,
equivalent expressions



CSE 444 - Spring 2019

59

Extended RA: Operators on Bags

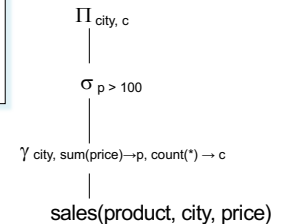
- Duplicate elimination δ
- Grouping γ
- Sorting τ

CSE 444 - Spring 2019

60

Logical Query Plan

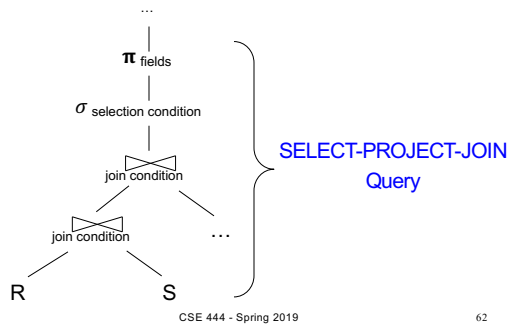
```
SELECT city, count(*)
FROM sales
GROUP BY city
HAVING sum(price) > 100
```



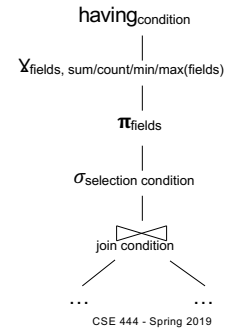
CSE 444 - Spring 2019

61

Typical Plan for Block (1/2)



Typical Plan For Block (2/2)



Benefits of Relational Model

- **Physical data independence**
 - Can change how data is organized on disk without affecting applications
- **Logical data independence**
 - Can change the logical schema without affecting applications (not 100%... consider updates)

CSE 444 - Spring 2019

64

Query Evaluation Steps Review

