

Today's Lecture

- A little bit on views
- Basic SQL queries
- Relational Algebra

Winter 2003

1

Recommended Readings

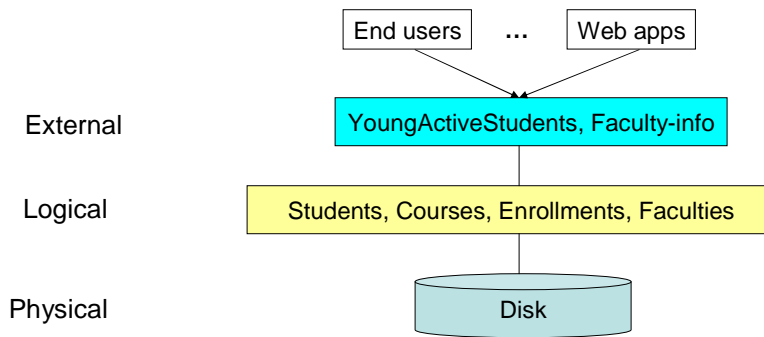
- Chapter 3
 - Section 3.6.1, 3.7
- Chapter 5
 - Section 5.2.1
- Chapter 4
 - Section 4.1, 4.2
- SQL for Web nerds – Simple queries
 - The link is on the class web page

Winter 2003

2

Views

- Physical schema
- Logical/Conceptual schema
- External schema
 - What applications can see
- Views mechanism supports logical data independence



3

Views

- External schema is created by defining views over the logical schema
- Of course, sometimes the logical schema serves well as the external schema but sometimes we need to define new “relations” over existing ones
- A view is a relation, defined in terms of other relations, but its tuples are not stored.
- Virtual views versus Materialized views

Winter 2003

4

Views

```
CREATE VIEW YoungActiveStudents (name, grade) AS  
  SELECT S.name, E.grade  
  FROM   Students S, Enrolled E  
  WHERE  S.sid = E.sid and S.age<21
```

- Views can be dropped using the DROP VIEW command. Similarly, tables can be dropped with DROP TABLE command
 - What happens if DROP TABLE is performed and there is a view defined on the table?
 - DROP TABLE command has options to let the user specify this.

Winter 2003

5

Views and Security

- Views can be used to present only the desired information, while hiding details in underlying relation(s).
 - Given YoungActiveStudents, but not Students or Enrolled, we can find young students who are enrolled, but not the *cid*'s of the courses they are enrolled in.
- Define views and allow only a group of users to access to it

Winter 2003

6

Relational Model: Summary

- A tabular representation of data.
- Simple and intuitive, currently the most widely used.
- Integrity constraints can be specified by the DBA, based on application semantics. DBMS checks for violations.
 - Two important ICs: primary and foreign keys
 - In addition, we *always* have domain constraints.
- Rules to translate ER to relational model
- How do we query relational data?

Winter 2003

7

Querying with SQL

- SQL is the most popular querying language for relational DBMS
- Basic form:

```
SELECT [DISTINCT]  $c_1, c_2, \dots, c_m$ 
FROM  $R_1, R_2, \dots, R_n$ 
[WHERE condition]
```

Attribute names

Relation names

- What is the semantics (or meaning) of such a query?

Winter 2003

8

Example 1 - a simply query

```
SELECT age
FROM Students
```

equivalent to:

```
SELECT age
FROM Students
WHERE TRUE
```

- For every tuple in Students, emit only the age component.

Students

Sid	Name	Age
1124	John	21
2187	Ann	20
8992	John	21
2346	Jane	19

Age
21
20
21
19

Note the
multiset
semantics

Winter 2003

9

Example 2 - DISTINCT

```
SELECT DISTINCT age
FROM Students
```

- For every tuple in Students, emit only the age component. Take the set of resulting values (i.e., remove duplicates)

Students

Sid	Name	Age
1124	John	21
2187	Ann	20
8992	John	21
2346	Jane	19

Age
21
20
19

Winter 2003

10

Example 3 - CONDITION

```
SELECT DISTINCT name
FROM Students
WHERE Age ≥ 20
```

- For every tuple in Students, if the value of the age value of that tuple is more than 19, emit the name component of that tuple. Remove duplicates.

Students

Sid	Name	Age
1124	John	21
2187	Ann	20
8992	John	21
2346	Jane	19

Name
John
Ann

Winter 2003

11

Example 4 - Tuple variables

```
SELECT DISTINCT *
FROM Students S ← S is a tuple variable
WHERE S.Age ≥ 20
```

- The symbol "*" is short for all columns
- The variable *S* binds to a tuple in Students
- For every tuple *S* in Students, if the value of the age column of *S* is more than 20, emit *S*. Remove duplicates.

Students

Sid	Name	Age
1124	John	21
2187	Ann	20
8992	John	21
2346	Jane	19

S.Sid	S.Name	S.Age
1124	John	21
2187	Ann	20
8992	John	21

Winter 2003

12

Example 5 - Multiple Relations in FROM clause

- What does the following query compute?

```
SELECT S.name, E.cid
FROM Students S, Enrolled E
WHERE S.sid=E.sid AND E.grade="A"
```

Sid	Name	Age
53630	Smith	21
21870	Ann	20
S 53831	John	21

sid	cid	grade
E 53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

S.name	E.cid
Smith	Topology112

Winter 2003

Condition

- The WHERE clause consists of a boolean combination of conditions using logical connectives AND, OR, NOT.
- Each condition is of the form *expression op expression*
 - *expression* is a column name, a constant, or an arithmetic or string expression
 - *op* is a comparison operator (<, ≤, =, <>, ≥, >)
 - Examples:
 - age > shoesize,
 - name <> "Jane" AND salary*0.1 > 100

Winter 2003

14

Meaning of an SQL query

```
SELECT [DISTINCT]  $c_1, c_2, \dots, c_m$   
FROM  $R_1, R_2, \dots, R_n$   
[WHERE condition]
```

UNION, DIFFERENCE

- For every n -ary tuple t (one from R_1 , one from R_2 , ..., one from R_n),
 - if t satisfies *condition* (i.e., condition evaluates to true), then emit the c_1, c_2, \dots, c_m components of t .
- Let *Result* denote the collection of all emitted results
- If DISTINCT is stated in the SELECT clause, remove duplicates in *Result* (i.e., Result is a set of tuples. Otherwise, Result is a bag of tuples)
- Note that the order of emitted results is not important
- Try out on your own on PostgreSQL!

Winter 2003

15

More on SQL

- Many more features in SQL! (look at the size of the manual)
- Before we leap ...
 - what is the foundation/properties of SQL?
- Recall that a data model consists of two parts
 - Formalism for describing data
 - Set of operations used to manipulate data
 - Relational algebra, relational calculus are two other different ways of describing operations in the relational model
- relational algebra and relational calculus are terse and formal.
 - Without the verbosity of SQL and good for reasoning
 - Form the basis of SQL

Winter 2003

16

Relational Algebra

- Queries in relational algebra are composed using basic operations (functions)
 - Selection
 - Projection
 - Set-theoretic operations:
 - Union
 - Intersection
 - Set-difference
 - Cross-product
 - Renaming
 - Joins
 - Division

Winter 2003

17

Compositionality

- Each operator takes as input one or two relations and returns a relation as output
- A complex expression is built from basic ones
- Five basic relational operations
 - Select
 - Project
 - Product
 - Union
 - Difference
- In relational algebra (or relational calculus), we assume set semantics (not bag or multiset semantics unless explicitly specified). Duplicate elimination is always performed

Winter 2003

18

Selection - $\sigma_{condition}(R)$

- Unary operation
 - Input: $R(A_1, \dots, A_n)$
 - Output relation has attributes A_1, \dots, A_n
 - Note:
 - named field notation: we have assumed that the attributes of input relation are known and “passed along” to the output. Easier to understand
 - Strictly speaking, the attributes are not known and we use position numbers instead to refer to the fields. (both are used in SQL)
 - Meaning: Takes a relation R and extracts only rows from R that satisfy the *condition*.
 - Output is always a set (no duplicates)

Winter 2003

19

Example - Select

- $\sigma_{rating > 6}$ (Hotels)
- Positional notation: $\sigma_{\$3 > 6}$ (Hotel)

Hotels

name	address	rating	capacity
Windsor	54 th ave	6.0	135
Astoria	5 th ave	8.0	231
BestInn	45 th st	6.7	28
ELogde	39 W st	5.6	45
ELodge	2nd E st	6.0	40

name	address	rating	capacity
Astoria	5 th ave	8.0	231
BestInn	45 th st	6.7	28

Winter 2003

20

Example - Select

- $\sigma_{\text{rating} > 6 \text{ AND } \text{capacity} > 50}(\text{Hotel})$

name	address	rating	capacity
Windsor	54 th ave	6.0	135
Astoria	5 th ave	8.0	231
BestInn	45 th st	6.7	28
ELodge	39 W st	5.6	45
ELodge	2nd E st	6.0	40

name	address	rating	capacity
Astoria	5 th ave	8.0	231

- Is $\sigma_{c_1}(\sigma_{c_2}(R)) = \sigma_{c_1 \text{ AND } c_2}(R)$? Prove or give a counter-example
- In class

Winter 2003

21

Projection - $\pi_{\text{attribute list}}(R)$

- Unary operation
 - Input : $R(A_1, \dots, A_n)$
 - Output relation has attributes according to *attribute list*
 - Meaning: Emits only the attributes stated in *attribute list* of every tuple in relation R.
- Eliminate duplicates.

Winter 2003

22

Example - Project

- $\pi_{\text{name, address}}(\text{Hotels})$

name	address
Windsor	54 th ave
Astoria	5 th ave
BestInn	45 th st
ELodge	39 W st
ELodge	2nd E st

- Suppose name and address form the key of Hotels relation, is the cardinality of the output relation the same as the cardinality of Hotels? Why?

Winter 2003

23

Example - Project

- $\pi_{\text{name}}(\text{Hotel})$

name
Windsor
Astoria
BestInn
ELodge

- Note that there are no duplicates

Winter 2003

24

Set Union - $R \cup S$

- Binary operation
 - Input: R and S must be union-compatible
 - They have the same set of arity, same number of columns
 - The i th column of R has the same type as the i th column of S for every column i .
 - Note that field names are not used in defining union-compatibility though we can also think that R and S is union-compatible if they having the same type (a set of record type).
 - Output has the same type as R
 - Meaning: the output consists of the set of all tuples in R and S

Winter 2003

25

Example - Set Union

- $\text{Dell_Desktops} \cup \text{IBM_Desktops}$

Dell_desktops

Harddisk	Speed	OS
20G	500Mhz	Windows
30G	1.0Ghz	Windows
20G	750Mhz	Linux

IBM_desktops

Harddisk	Speed	OS
30G	1.2Ghz	Windows
20G	500Mhz	Windows

All tuples in R occurs in $R \cup S$
 All tuples in S occurs in $R \cup S$
 $R \cup S$ contains tuples that either occur in R or S (or both).

Harddisk	Speed	OS
20G	500Mhz	Windows
30G	1.0Ghz	Windows
20G	750Mhz	Linux
30G	1.2Ghz	Windows

Winter 2003

26

Example - Set Union

- Dell_Desktops \cup IBM_Desktops

Dell_desktops

Harddisk	Speed	OS
20G	500Mhz	Windows
30G	1.0Ghz	Windows
20G	750Mhz	Linux

IBM_desktops

Harddisk	Speed	OS
30G	1.2Ghz	Windows
20G	500Mhz	Windows

$R \cup S = S \cup R$ (commutativity)
 $R \cup (S \cup T) = (R \cup S) \cup T$ (associativity)

Harddisk	Speed	OS
20G	500Mhz	Windows
30G	1.0Ghz	Windows
20G	750Mhz	Linux
30G	1.2Ghz	Windows
20G	500Mhz	Windows

Winter 2003

27

Set Difference - R - S

- Binary operation
 - Input: R and S must be union-compatible
 - Output has the same type as R
 - Meaning: output consists of all tuples in R and not in S

Winter 2003

28

Example - Difference

- Dell_Desktops - IBM_Desktops

Dell_desktops

Harddisk	Speed	OS
20G	500Mhz	Windows
30G	1.0Ghz	Windows
20G	750Mhz	Linux

IBM_desktops

Harddisk	Speed	OS
30G	1.2Ghz	Windows
20G	500Mhz	Windows

Dell_Desktops – IBM_Desktops

Harddisk	Speed	OS
30G	1.0Ghz	Windows
20G	750Mhz	Linux

Winter 2003

29

Example - Difference

- IBM_Desktops – Dell_Desktops

Dell_Desktops

Harddisk	Speed	OS
20G	500Mhz	Windows
30G	1.0Ghz	Windows
20G	750Mhz	Linux

IBM_Desktops

Harddisk	Speed	OS
30G	1.2Ghz	Windows
20G	500Mhz	Windows

IBM_Desktops – Dell_Desktops

Harddisk	Speed	OS
30G	1.2Ghz	Windows

Is it commutative? NO. We have just seen a counter-example.
Is it associative?

Winter 2003

30

Join or Cross Product - $R \times S$

- Join is a generic term for a variety of operations that connect two relations that may not be union-compatible.
 - basic operation is the product, $R \times S$
- Binary operation
 - Input: $R(A_1, \dots, A_m), S(B_1, \dots, B_n)$
 - Output is a relation with columns $A_1, \dots, A_m, B_1, \dots, B_n$
 - What happens if R and S contain common attributes?
 - Meaning: concatenates every tuple in R with every tuple in S .

Winter 2003

31

Example - Cross Product

R

A	B	C
a ₁	b ₁	c ₁
a ₂	b ₂	c ₂

S

D	E
d ₁	e ₁
d ₂	e ₂
d ₃	e ₃

$R \times S$

A	B	C	D	E
a ₁	b ₁	c ₁	d ₁	e ₁
a ₁	b ₁	c ₁	d ₂	e ₂
a ₁	b ₁	c ₁	d ₃	e ₃
a ₂	b ₂	c ₂	d ₁	e ₁
a ₂	b ₂	c ₂	d ₂	e ₂
a ₂	b ₂	c ₂	d ₃	e ₃

Is it commutative?
 Is it associative?
 Is it distributive? I.e.,
 $Is R \times (S \cup T) = (R \times S) \cup (R \times T)$

Winter 2003

32

Example - Cross Product

- What happens if R and S contain common attributes?
e.g., R(A,B,C) and S(A,E)
 - Answer varies. We may assume positional suffixes exists to distinguish among conflicts

A.1	B	C	A.2	E
a ₁	b ₁	c ₁	d ₁	e ₁
a ₁	b ₁	c ₁	d ₂	e ₂
a ₁	b ₁	c ₁	d ₃	e ₃
a ₂	b ₂	c ₂	d ₁	e ₁
a ₂	b ₂	c ₂	d ₂	e ₂
a ₂	b ₂	c ₂	d ₃	e ₃

Winter 2003

33

More complex queries

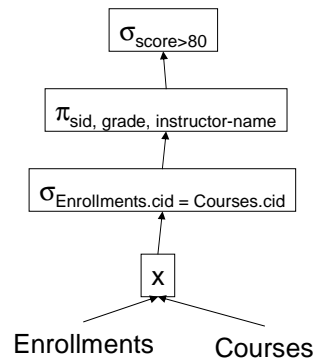
- Relational queries can be composed to form more complex queries
- Enrollments(sid, cid, score), Courses(cid, cname, instructor-name)
- $\sigma_{\text{score} > 80} (\pi_{\text{sid, grade, instructor-name}} (\sigma_{\text{Enrollments.cid} = \text{Courses.cid}} (\text{Enrollments} \times \text{Courses})))$
- We can now find out the student and instructor pairs where the student scored well (more than 80 pts) in a course taught by the instructor

Winter 2003

34

More complex queries

- Query tree or Operator tree



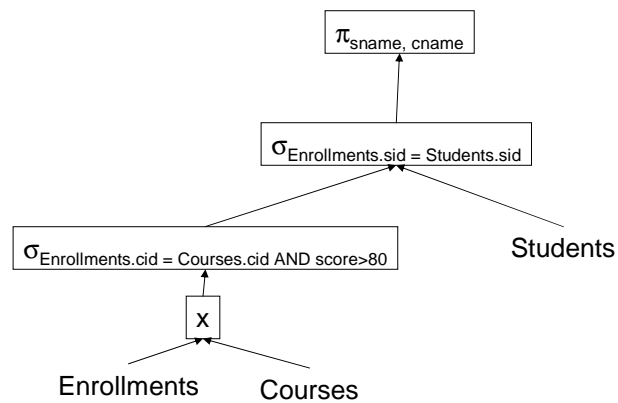
Tells us exactly the procedure to take in order to arrive at the answer. Also known as execution plan.

Winter 2003

35

More complex queries

- Find the student and course names where the student scored well (more than 80 pts) in the course



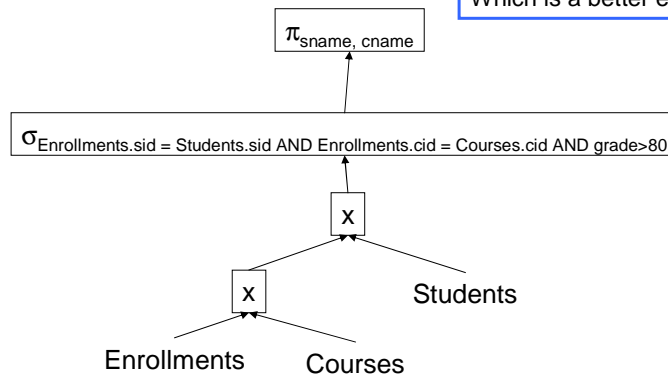
Winter 2003

36

An alternative plan

- Find the student and course names where the student scored well (more than 80 pts) in the course

Which is a better execution plan?



Winter 2003

37

Renaming - $\rho_x(E)$

- Renaming is used to give a name to the result of a relational algebra expression.
- Suppose the schema is $R(A, B, C)$, we write $\rho_{B \rightarrow D}(R)$ to change the schema to $R(A, D, C)$
- Suppose $R(A, B, C)$ and $S(D, A)$ in the previous example. We have naming conflict on attribute A in $R \times S$. We can write $\rho_{1 \rightarrow G}(R \times S)$ to correct the ambiguity. Therefore the result has schema $R \times S(G, B, C, D, A)$

Winter 2003

38