

## RA to DRC

*Every relational algebra query can be translated into a domain relational calculus query.*

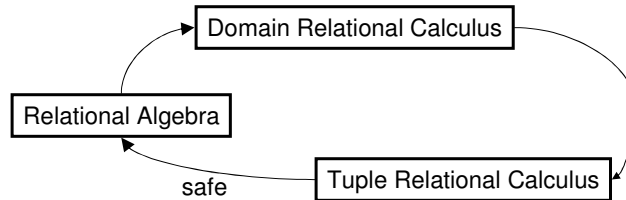
- Proof by induction on the structure of the relational algebra query
- In class

## DRC to TRC

- Every DRC query can be translated into a tuple relational calculus query.
- Proof by induction on the structure of the DRC query.
- In class

## TRC to RA

- A safe TRC query can be translated into a RA query
- Therefore, we have the following picture so far



- More precisely,  $RA \equiv \text{safe DRC} \equiv \text{safe TRC}$

Winter 2003

3

## Big Deal?

Relational Algebra and Relational Calculus greatly influenced the design of the first relational databases

- SQL, a relationally complete language, bears strong resemblance to TRC.
  - `SELECT x FROM R r WHERE C` roughly corresponds to  $\{ x \mid \exists r \in R, C \}$ !
- Relational Algebra:
  - used for optimization of SQL queries. Many equivalent RA queries for the same SQL query. Specialized and efficient algorithms for various relational operators. The relational optimizer picks the best among the possible plans.
  - Performance of relational engines are hard to beat!

Winter 2003

4

## Big Deal?

- Relational Calculus:
  - tells us what SQL can or cannot express. E.g., Known that FO logic cannot express recursive queries. E.g. compute the ancestor relation of relation PC(parent, child).
  - To write more complex queries such as recursive queries, extensions are added to SQL. Make a DB talk to host languages etc. The price you pay: your query may not be easily optimizable

Winter 2003

5

## Back to SQL

- Previous lectures
  - Data Manipulation Language (DML)
    - Insert, delete, modify rows, pose queries
  - Data Definition Language (DDL)
    - Create, delete, modify definitions for tables and views
    - Integrity constraints
- We will see other aspects of SQL Tentative
  - More advanced constructs of SQL
  - Triggers and Advanced Integrity Constraints
  - Embedded and Dynamic SQL

Winter 2003

6

## Basic form of SQL

- We have already seen the basic form of SQL:

```
SELECT    [DISTINCT] target-list
FROM      relation-list
[WHERE    qualification]
```

- *relation-list*: A list of relation names (possibly with a range-variable or tuple variable after each name). E.g., FROM Sailors s, Reserves r
- *target-list*: A list of attributes of relations in *relation-list*. \* can be used to denote all attributes. E.g., SELECT s.sname, SELECT \*
- *qualification*: Comparisons (Attr *op* const or Attr1 *op* Attr2, where *op* is one of <, >, =, ≤, ≥, ≠ combined using AND, OR and NOT.
- DISTINCT (optional) keyword indicates that the answer should not contain duplicates. Default is that duplicates are **not** eliminated!

Winter 2003

7

## Semantics of the SQL query

- Informally:
  - Result1: Compute the cartesian product of all relations in the relation list
  - Result2: For every tuple in Result1, apply the condition stated in the qualification list. Eliminate the tuple if the condition is not satisfied.
  - Result3: Retrieve only the required components from each tuple in Result2 according to target list
  - Result: Eliminate duplicate rows in Result3 if DISTINCT keyword is present in the SELECT clause

Winter 2003

8

## Example Database

Customers

sid	cname	level	type	age
2336	Cho	Beginner	snowboard	18
2334	Luke	Inter	snowboard	25
1887	Ice	Advanced	ski	20
2339	Paul	Beginner	ski	33

Activities

sid	slope-id	day
2336	s3	01/05/03
2336	s1	01/06/03
2336	s1	01/07/03
1887	s2	01/07/03
1887	s1	01/07/03
2334	s2	01/05/03

Slopes

slope-id	name	color
s1	Mountain Run	blue
s2	Olympic Lady	black
s3	Magic Carpet	green

Winter 2003

9

## Basic SQL Examples

Q1: Find all snowboarders  
 SELECT \*  
 FROM Customers c  
 WHERE c.type = "snowboard"

sid	cname	level	type	age
2336	Cho	Beginner	snowboard	18
2334	Luke	Inter	snowboard	25

Q2: Find the names of all  
 customers on Mountain Run  
 slope on the day 01/07/03  
 SELECT sname  
 FROM Customer c, Activities a,  
 Slopes s  
 WHERE c.sid = a.sid AND  
 s.slope-id = a.slope-id AND  
 a.day="01/07/03" AND s.name  
 = "Mountain Run"

cname
Cho
Ice

Winter 2003

10

# DISTINCT

Q3: Return the names of customers who are on the slope for at least one day

```
SELECT c.sname
FROM Customers c, Activities a
WHERE c.sid=a.sid
```

cname
Cho
Luke
Ice
Cho
Cho
Ice

```
SELECT DISTINCT c.cid
FROM Customers c, Activities a
WHERE c.sid=a.sid
```

cname
Cho
Luke
Ice

Winter 2003

11

# UNION

- Return the names of customers who either went on some slope on 11/05/03 or skied on Mountain Run

```
SELECT c.cname
FROM Customers c, Activities a
WHERE c.sid=a.sid AND a.day="11/05/03"
```

UNION

```
SELECT c.cname
FROM Customers c, Activities a, Slopes s
WHERE c.sid=a.sid AND
      s.slope-id=a.slop-id AND
      s.sname="Mountain Run"
```

cname
Cho
Luke
Ice

- The subqueries must be union-compatible
- Note that duplicates are eliminated even though there is no keyword DISTINCT!
- UNION has set semantics.

Winter 2003

12

## UNION ALL

```
SELECT c.cname
FROM Customers c, Activities a
WHERE c.sid=a.sid AND a.day="11/05/03"
UNION ALL
SELECT c.cname
FROM Customers c, Activities a, Slopes s
WHERE c.sid=a.sid AND
      s.slope-id=a.slop-id AND
      s.sname="Mountain Run"
```

cname
Cho
Luke
Cho
Ice
Cho

- Duplicates are not eliminated
- UNION ALL has multiset or bag semantics

Winter 2003

13

## UNION ALL

- What about the following query?

```
SELECT DISTINCT c.cname
FROM Customers c, Activities a
WHERE c.sid=a.sid AND a.day="11/05/03"
UNION ALL
SELECT DISTINCT c.cname
FROM Customers c, Activities a, Slopes s
WHERE c.sid=a.sid AND
      s.slope-id=a.slop-id AND
      s.sname="Mountain Run"
```

cname
Cho
Luke
Cho
Ice

- INTERSECT and INTERSECT ALL

Winter 2003

14

# EXCEPT

- Set difference
- Find the names of all customers who have been to Mountain Run but not Olympic Lady

cname
Cho

```
SELECT c.cname
FROM Customers c, Activities a, Slopes s
WHERE c.sid=a.sid AND s.slope-id=a.slop-id
      AND s.sname="Mountain Run"
EXCEPT
SELECT c.cname
FROM Customers c, Activities a, Slopes s
WHERE c.sid=a.sid AND s.slope-id=a.slop-id
      AND s.sname="Olympic Lady"
```

- The subqueries must be union-compatible
- Like before, duplicates are eliminated!
- EXCEPT has set-semantics
- EXCEPT ALL will return the name "Cho" twice

Winter 2003

15

# Summary

- SELECT ... FROM ... WHERE
- UNION, UNION ALL, INTERSECT, INTERSECT ALL, EXCEPT, EXCEPT ALL
- Can you translate the SQL queries with set semantics shown in this lecture into relational algebra queries?

Winter 2003

16