

Answering Queries Using Views: Introduction and Cost-Based Approaches

CPS 296.1
Topics in Database Systems

Introduction

- Given a query Q and a set of views \mathcal{V}
 - Is it possible to answer Q using only the views in \mathcal{V} ?
 - If not, what is the maximal set of tuples in the answer of Q that we can obtain from the views in \mathcal{V} ?
 - If we can access both the views and base tables, what is the cheapest query execution plan for answering Q ?
- Applications in query optimization, database design, data integration, data warehouse design, semantic data caching, ...

➤ Halevy. "Answering Queries Using Views: A Survey." *VLDB Journal*, 2001

2

Query optimization example (slide 1)

- Schema
 - Prof(name, area)
 - Course(c-number, title)
 - Teaches(prof, c-number, quarter)
 - Registered(student, c-number, quarter)

3

Query optimization example (slide 2)

- Query: my 3xx courses and students in them
 $Q :=$

```
select Course.title, Registered.student
from Teaches, Course, Registered
where Teaches.prof = 'Jun'
and Teaches.c-number = Course.c-number
and Teaches.c-number = Registered.c-number
and Teaches.c-number >= 300;
```
- Materialized view: graduate courses and students in them
 $Grad :=$

```
select Course.c-number, Course.title, Registered.student
from Course, Registered
where Course.c-number = Registered.c-number
and Course.c-number >= 200;
```
- Faster to answer the query using the materialized view
 $Q =$

```
select Grad.title, Grad.student
from Teaches, Grad
where Teaches.prof = 'Jun'
and Teaches.c-number = Grad.c-number
and Teaches.c-number >= 300;
```

4

Data integration example (slide 1)

- Global schema
 - Teaches(prof, c-number, quarter, univ)
 - Course(c-number, title, univ)
- Source contents described as views
 $DB-Courses :=$

```
select Course.title, Teaches.prof, Course.c-number, Course.univ
from Teaches, Course
where Teaches.c-number = Course.c-number
and Teaches.univ = Course.univ
and Course.title = 'Database Systems';
```

 $Duke-Grad :=$

```
select Course.title, Teaches.prof, Course.c-number, Course.univ
from Teaches, Course
where Teaches.c-number = Course.c-number
and Teaches.univ = Course.univ
and Course.univ = 'Duke'
and Course.c-number >= 200;
```

➤ Local-as-view approach (versus global-as-view)

5

Data integration example (slide 2)

- Query: "Database Systems" course at Duke
 - Complete answer at source DB-Courses

```
select * from DB-Courses where univ = 'Duke';
```
 - Incomplete answer at source Duke-Grad

```
select * from Duke-Grad where title = 'Database Systems';
```
- Query: all profs at Duke
 - Incomplete answer at both DB-Courses and Duke-Grad

```
select prof from DB-Courses where univ = 'Duke';
select prof from Duke-Grad;
```
 - The best we can do is to union the two answers

6

Other applications

- Access path/index selection
 - Access paths and indexes = materialized views with binding patterns
- Data warehousing
 - Warehouse data = materialized views
- Semantic data caching
 - Cached data = materialized views
- Use materialized views to improve query performance

7

Containment and equivalence

- Containment: Q_1 contains Q_2 if for all database instance, the result of Q_1 contains the result of Q_2
- Equivalence: Q_1 is equivalent to Q_2 if Q_1 contains Q_2 and Q_2 contains Q_1
- Important: “for all database instance”
 - $R = \{ (1), (2), (3) \}$
 - $Q_1 = \sigma_{A>1} R$ and $Q_2 = \sigma_{A<3} R$ return the same result
 - But Q_1 and Q_2 are not equivalent in general

8

Equivalent and maximal rewritings

- Given a query Q and a set of views \mathcal{V}
- Q' is an equivalent rewriting of Q using \mathcal{V} if
 - Q' refers only to the views in \mathcal{V} , and
 - Q' is equivalent to Q
- Q' is a maximally-contained rewriting of Q using \mathcal{V} (w.r.t. some query language) if
 - Q' refers only to the views in \mathcal{V} , and
 - Q' is contained in Q , and
 - There is no Q'' (written in the same query language) such that Q'' is contained in Q and Q' is strictly contained in Q''
 - That is, Q' is (one of) the best we can do with a given language
 - There may be multiple maximally-contained rewritings

9

Finding maximal rewriting...

- Source 1: list all SIGMOD papers
- Source 2: given a paper, list all papers cited by it
- Source 3: given a paper, return its rating (1-10)
 - Sources 2 and 3 are views with binding patterns
- Query: find all papers with rating higher than 9

10

... is not easy

- From Source 1, find all SIGMOD papers
- From Source 2, find all papers cited by SIGMOD papers
- From Source 2, find all papers cited by (papers cited by SIGMOD papers)
- From Source 2, find all papers cited by (papers cited by (papers cited by SIGMOD papers))
- ...
- Repeat until no more papers can be found
- From Source 3, determine which papers have rating higher than 8

11

Certain answers

- Maximally-contained rewriting depends on the expressive power of the query language
- What is the best we can do (regardless of the query language)?
 - Find all certain answers of a query using views
- Tuple t is a certain answer to Q if t is in the result of Q for any database instance that is “consistent” with the given view contents

12

Closed- vs. open-world assumption

- Closed-world assumption: Views contain complete answers
- Open-world assumption: Views may contain incomplete answers
- Example: $R(A, B)$, $V_1 = \pi_A R$, $V_2 = \pi_B R$, $Q = R$
 - Suppose V_1 contains a single tuple (a) , and V_2 contains a single tuple (b)
 - Under closed-world assumption, (a, b) is a certain answer to Q
 - Under open-world assumption, (a, b) is not certain₃

Approaches to answering queries using views

- Cost-based rewriting
 - Query optimization, access path selection, data warehousing, semantic caching
 - Focuses on finding an efficient execution plan for an equivalent rewriting
 - Often uses SQL, relational/bag algebra
- Logical rewriting
 - Data integration
 - Focuses on finding a maximally-contained rewriting, or as many certain answers as possible
 - Often uses Datalog (a Prolog-like query language)₄

Roadmap

- Cost-based rewriting for SQL query optimization
 - Basic question: When is a view usable for a query?
 - Transformational approach
 - Selinger-style (System-R) approach
- Logical rewriting for data integration using Datalog

15

When is a view usable for a query (slide 1)

- Each occurrence of a table in V must be matched with an occurrence of the same table in Q

```

Advises(prof, student)
Teaches(prof, c-number, quarter)
Registered(student, c-number, quarter)

Q: select Advises.prof, Advises.student, Registered.quarter
    from Registered, Teaches, Advises
    where Registered.c-number = Teaches.c-number
    and Registered.quarter = Teaches.quarter
    and Advises.prof = Teaches.prof
    and Advises.student = Registered.student
    and Registered.quarter >= 'win98';

V: select Registered.student, Teaches.prof, Registered.quarter
    from Registered, Teaches
    where Registered.c-number = Teaches.c-number
    and Registered.quarter = Teaches.quarter
    and Registered.quarter >= 'win97';
    
```

16

When is a view usable for a query (slide 2)

- Intuition: If a table is joined in V but not in Q , then V is unusable because the additional join may filter out some V tuples that could contribute to Q

```

Q: select Advises.prof, Advises.student, Registered.quarter
    from Registered, Teaches, Advises
    where Registered.c-number = Teaches.c-number
    and Registered.quarter = Teaches.quarter
    and Advises.prof = Teaches.prof
    and Advises.student = Registered.student
    and Registered.quarter >= 'win98';

V: select Advises.prof, Advises.student, Registered.quarter
    from Registered, Teaches, Advises, Area
    where Registered.c-number = Teaches.c-number
    and Registered.quarter = Teaches.quarter
    and Advises.prof = Teaches.prof
    and Advises.student = Registered.student
    and Registered.quarter >= 'win97'
    and Teaches.prof = Area.prof;
    
```

17

When is a view usable for a query (slide 3)

- V must either apply the join and selection predicates in Q , or apply a logically weaker predicate (or not applying it at all) and preserve the attributes on which the predicates still need to be applied (unless these attributes can be recovered somehow)

```

Q: select Advises.prof, Advises.student, Registered.quarter
    from Registered, Teaches, Advises
    where Registered.c-number = Teaches.c-number
    and Registered.quarter = Teaches.quarter
    and Advises.prof = Teaches.prof
    and Advises.student = Registered.student
    and Registered.quarter >= 'win98';

V: select Registered.student, Teaches.prof, Registered.quarter
    from Registered, Teaches
    where Registered.c-number = Teaches.c-number
    and Registered.quarter = Teaches.quarter
    and Registered.quarter >= 'win97';
    
```

18

When is a view usable for a query (slide 4)

- Intuition: A stronger predicate may filter out some V tuples that could contribute to Q ; a weaker predicate means the original predicate must be re-applied (so attributes must be preserved)

```
Q: select Advises.prof, Advises.student, Registered.quarter
    from Registered, Teaches, Advises
    where Registered.c-number = Teaches.c-number
    and Registered.quarter = Teaches.quarter
    and Advises.prof = Teaches.prof
    and Advises.student = Registered.student
    and Registered.quarter >= 'win98';
```

```
V': select Registered.student, Teaches.prof
     from Registered, Teaches
     where Registered.c-number = Teaches.c-number
     and Registered.quarter = Teaches.quarter
     and Registered.quarter >= 'win99';
```

19

When is a view usable for a query (slide 5)

- V must not project out attributes that are selected by Q (unless they can be recovered somehow)

```
Q: select Advises.prof, Advises.student, Registered.quarter
    from Registered, Teaches, Advises
    where Registered.c-number = Teaches.c-number
    and Registered.quarter = Teaches.quarter
    and Advises.prof = Teaches.prof
    and Advises.student = Registered.student
    and Registered.quarter >= 'win98';
```

```
V: select Registered.student, Teaches.prof, Registered.quarter
     from Registered, Teaches
     where Registered.c-number = Teaches.c-number
     and Registered.quarter = Teaches.quarter
     and Registered.quarter >= 'win97';
```

20

Transformational query optimizer

- Start with a query execution plan P
- Repeat until some stopping condition (e.g., time runs out):
 - Apply a random transformation to P
 - Hopefully, the cost of P decreases after the transformation (but this is not necessary for some algorithms)

➤ Usual tricks for searching apply: iterative improvement, simulated annealing, etc.

21

Transformational optimization using views

- Add a transformation that rewrites the query to use a view
 - Sometimes an entire subplan can be replaced by a view exactly, but sometimes additional processing is needed to use the view correctly
 - Special indexing structures are often used to help determine which views are relevant to the query
 - Remember the directory index in DynaMat?
- Easy to incorporate into an existing optimizer
- Implemented in SQL Server, DB2, Oracle

22

Selinger-style query optimizer

Basic ideas

- Bottom-up generation of plans
 - An n -way join plan can be constructed by joining a k -way join plan with an $(n - k)$ -way join plan
- Pruning of plans
 - A plan is pruned if its cost is higher than another plan that joins the same set of table and produces the answers in the same or a more “interesting” order

23

Selinger-style query optimization

- Pass 1: Find all single-table plans; prune
- Pass 2: Find 2-way join plans by joining the best single-table plans (found in Pass 1); prune
- ...
- Pass n : Find n -way join plans by joining the best k -table plans (found in Pass k) with the best $(n - k)$ -table plans (found in Pass $n - k$); prune
 - If the query has only n tables, stop
- ...

24

Example

V_1 (students and their majors):

$\pi_{\text{student, dept}} \text{ Major}$

V_2 (students in theory courses):

$\pi_{\text{student, c-number}} \sigma_{\text{title like '%theory%'}}$
(Registered $\triangleright \triangleleft$ c-number Course)

V_3 (majors of students in 3xx courses):

$\pi_{\text{dept, c-number}} \sigma_{\text{c-number} \geq 300}$
(Registered $\triangleright \triangleleft$ student Major)

Q (students and their majors in 3xx theory courses):

$\pi_{\text{student, dept}} \sigma_{\text{c-number} \geq 500 \text{ and title like '%theory%'}}$
(Course $\triangleright \triangleleft$ c-number Registered $\triangleright \triangleleft$ student Major) 25

Partial and complete plans

- Complete plans return the final result of the query
- Partial plans still need additional processing
- Example plans for Q
 - Partial: $V_1 \triangleright \triangleleft$ student V_2
 - Complete: $\pi_{\text{student, dept}} \sigma_{\text{c-number} \geq 500} (V_1 \triangleright \triangleleft$ student $V_2)$
 - Partial: $V_3 \triangleright \triangleleft$ c-number V_2
 - Complete: $\pi_{\text{student, dept}} (V_3 \triangleright \triangleleft$ c-number $V_2 \triangleright \triangleleft$ student, dept $V_1)$
 - Seems redundant, but may in fact be a winning plan

26

Selinger-style optimization using views (slide 1)

- Bottom-up generation of plans
 - Partial plans can be combined to form bigger ones
 - Partial plans can be patched with additional selection and projection to obtain complete ones
 - Complete plans should not need to be combined
- Pruning of plans
 - A plan is pruned if its cost is higher than another plan that has greater or equal contribution to the query (e.g., covers more joins in the query)
- Termination testing
 - No more partial plans left unexplored

27

Selinger-style optimization using views (slide 2)

- Pass 1
 - Find all views relevant to the query
 - Distinguish between partial and complete plans
 - Prune
- ...
- Pass n
 - Consider joining the best partial plans found in previous passes
 - Distinguish between partial and complete plans
 - Prune
 - If there are no partial plans left to explore, stop
- ...

28