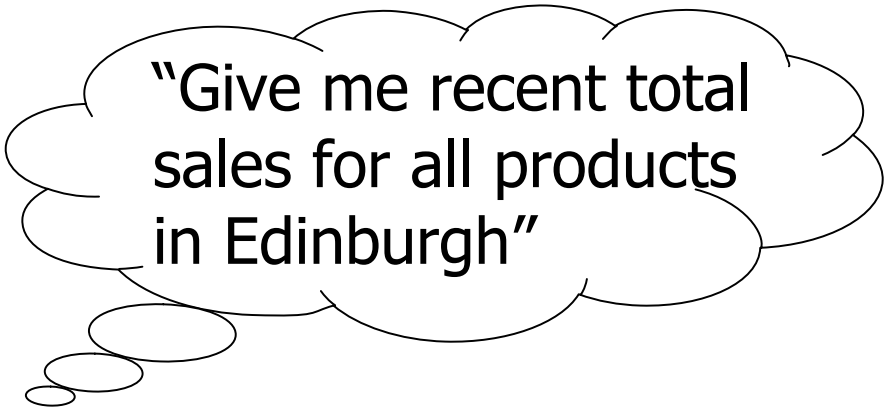
Selecting and Using Views To Compute Aggregate Queries

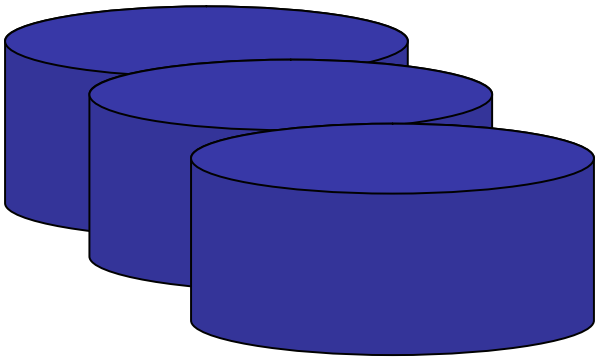
Foto Afrati (NTUA Greece)
and Rada Chirkova (NC State University)

What this Talk Is About

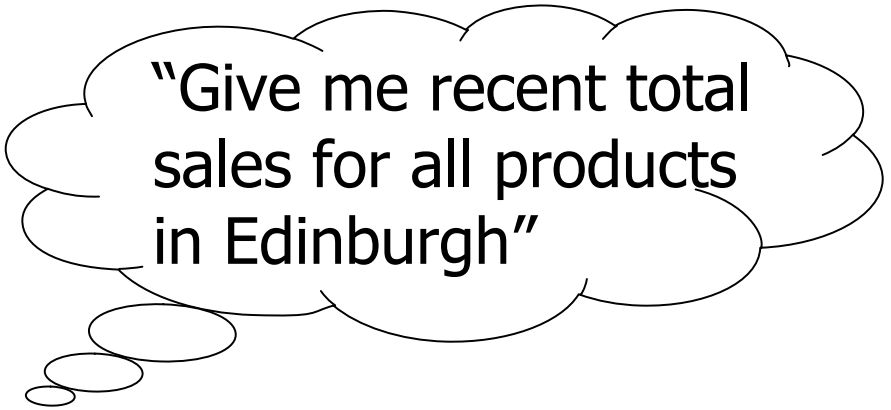
- Topic: using materialized views to improve query-evaluation performance
- Context:
  - Workloads of queries with or without aggregation
  - Designing optimal views under constraints
- Results:
  - Formats of equivalent rewritings of queries using views: sufficient and necessary conditions
  - Designing and using views: complexity and algorithms

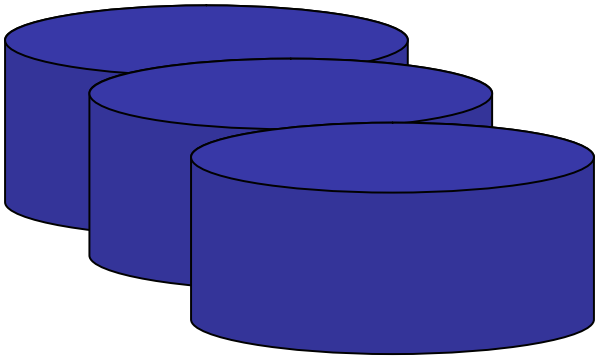
Using Databases: Asking Queries ...

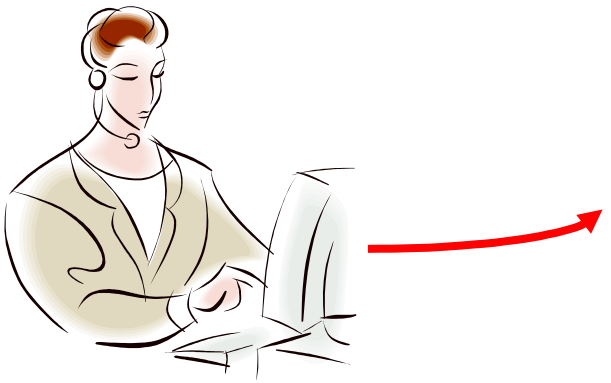




Using Databases: Asking Queries …

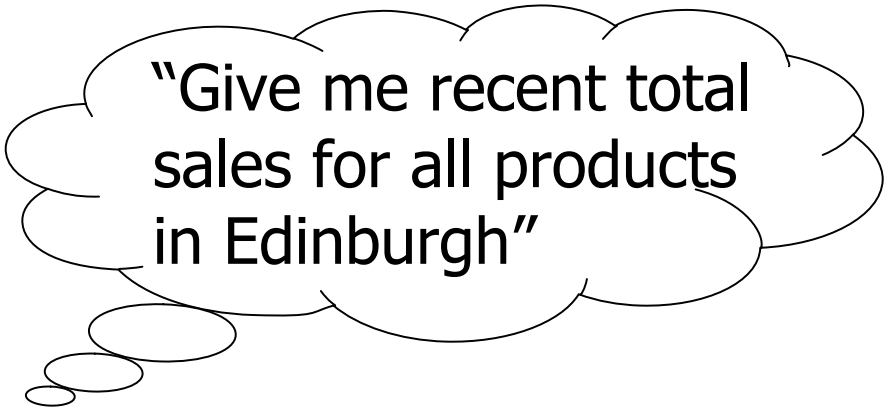


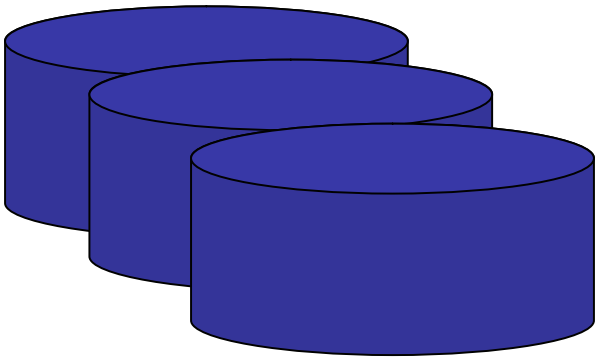




```
SELECT Product, SUM(Sales)
FROM SalesOfProducts
WHERE Location = 'Edinburgh'
AND Date > '12/31/2003'
GROUP BY Product;
```

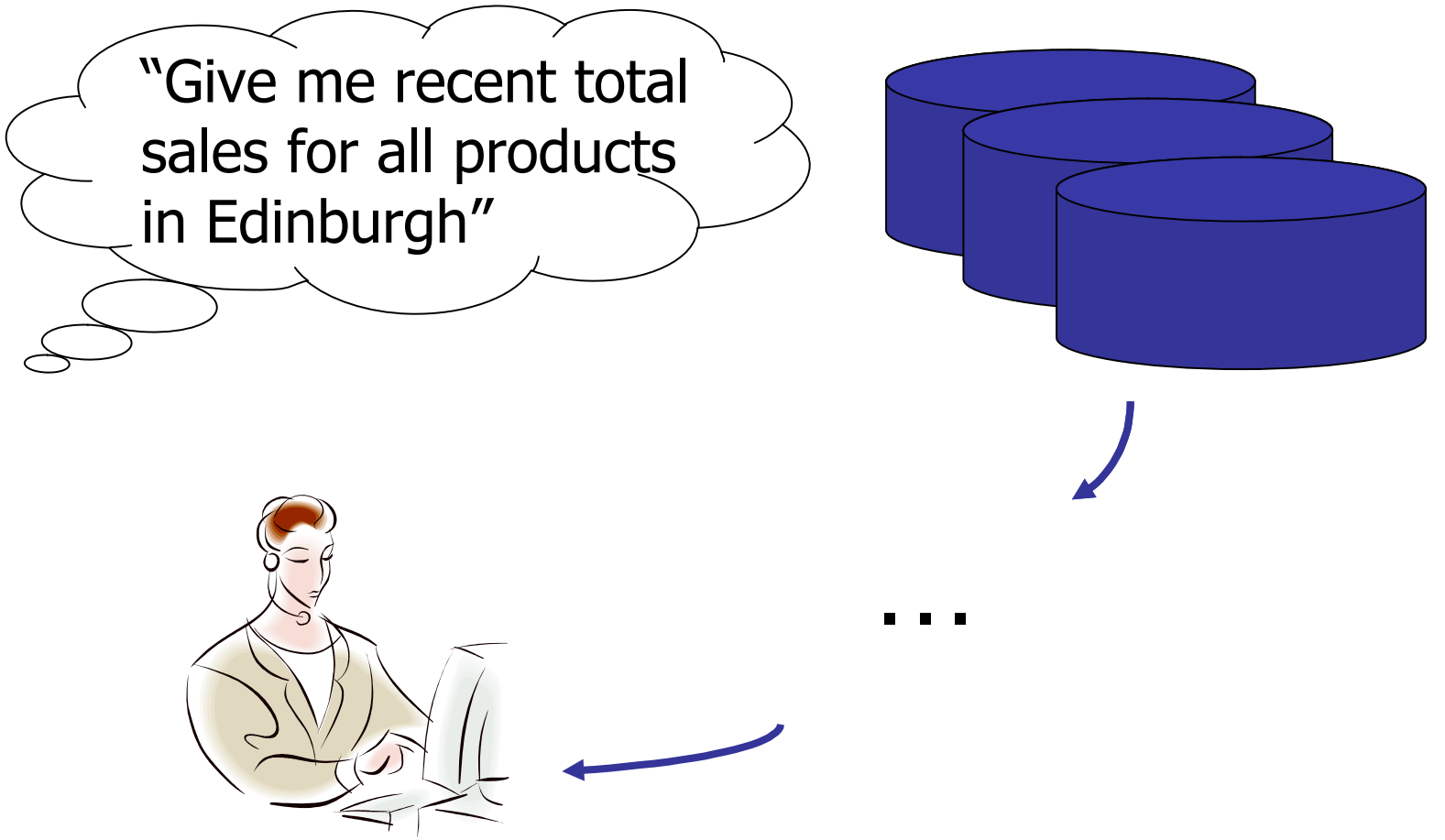
Using Databases: Asking Queries ...



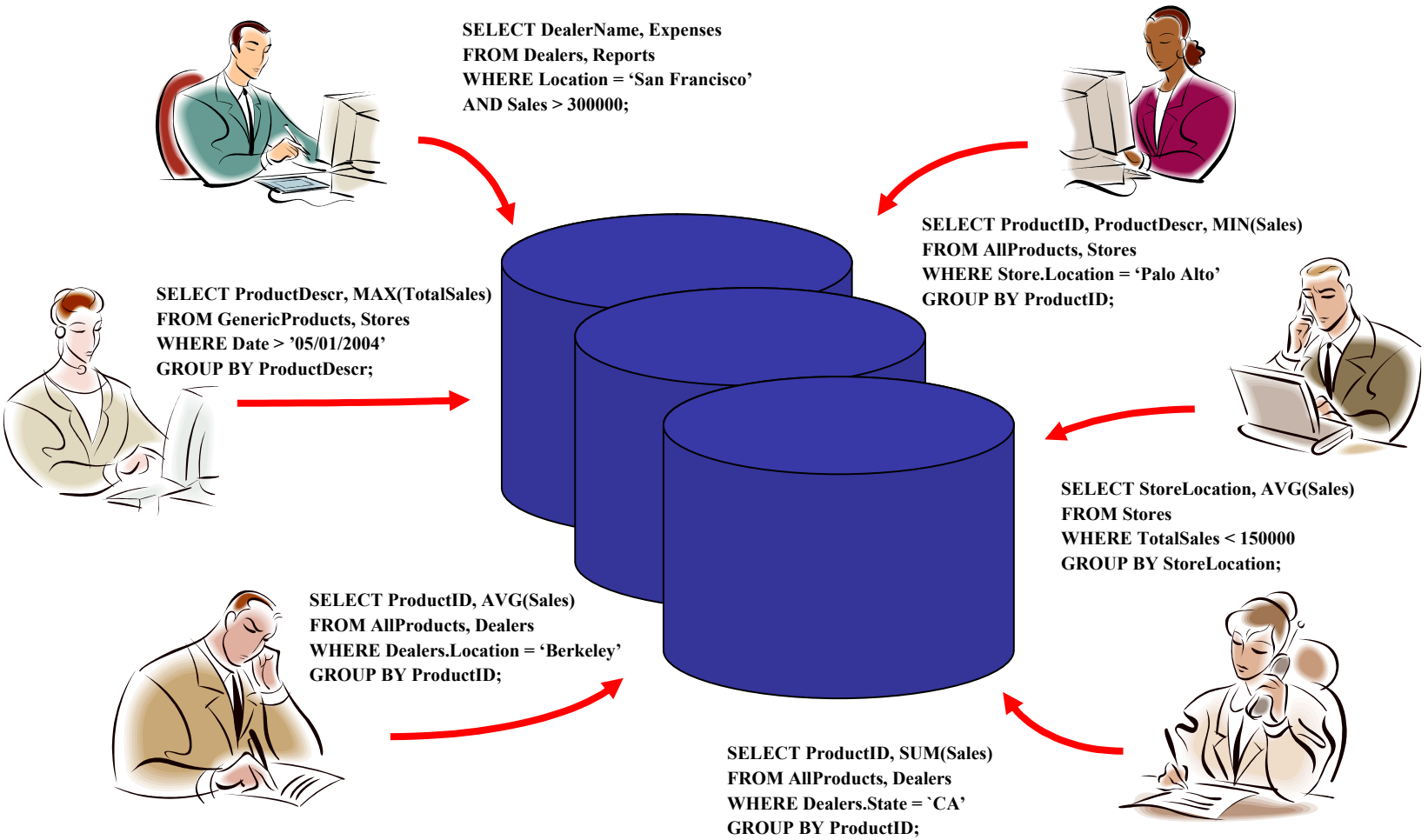


```
SELECT Product, SUM(Sales)
FROM SalesOfProducts
WHERE Location = 'Edinburgh'
AND Date > '12/31/2003'
GROUP BY Product;
```

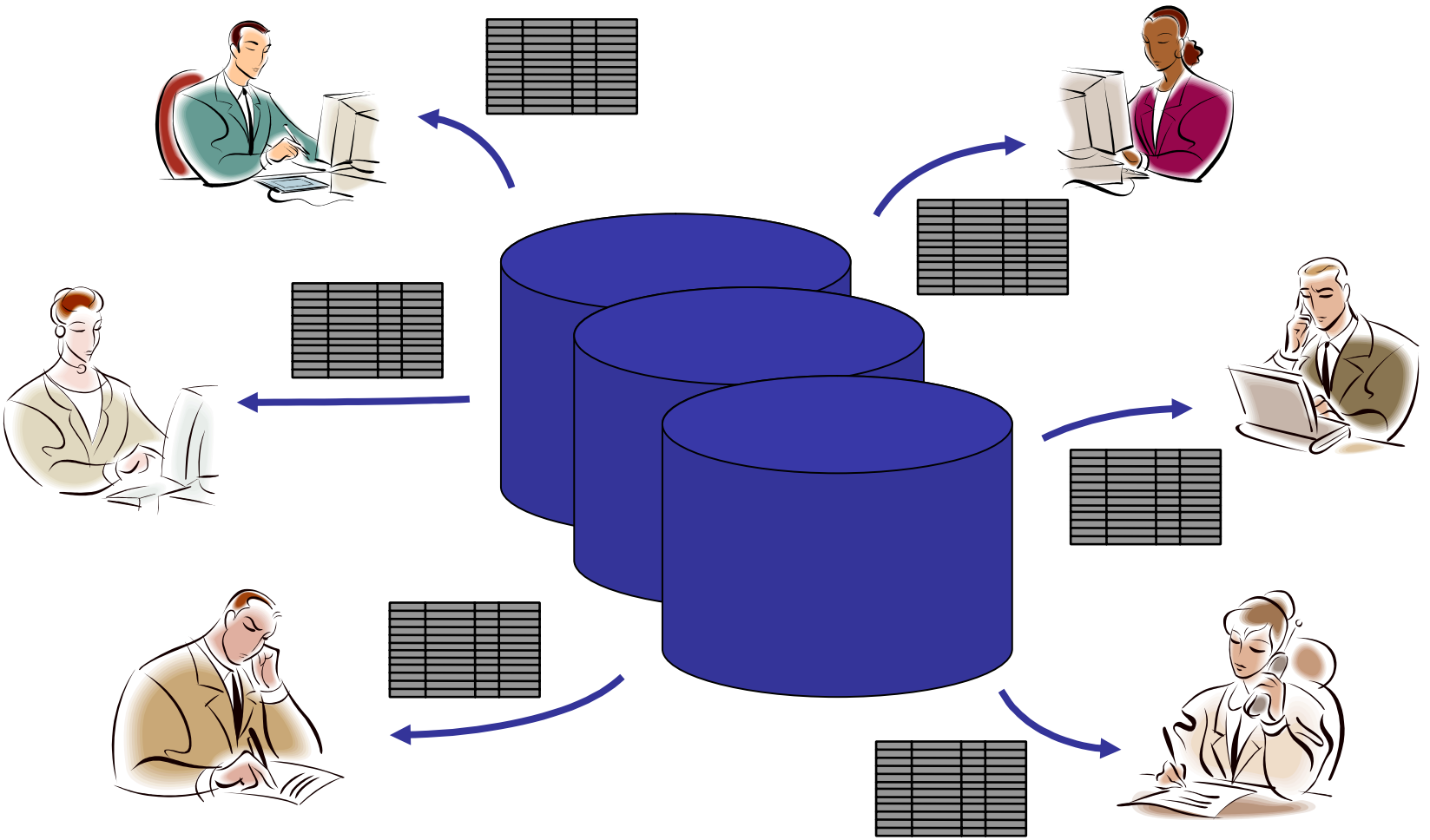
… And Getting Answers, Without Delay



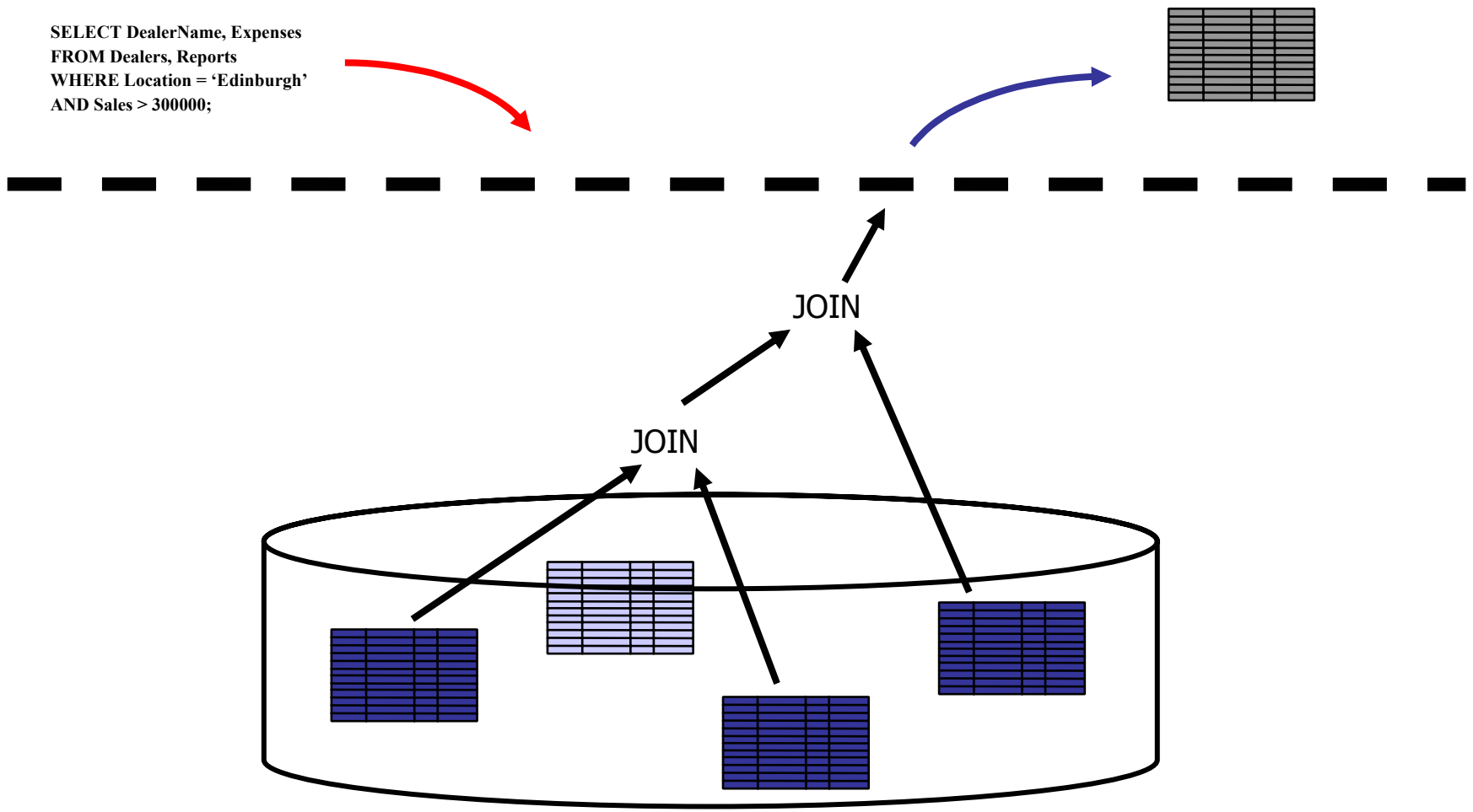
When Users Ask Complex Queries …



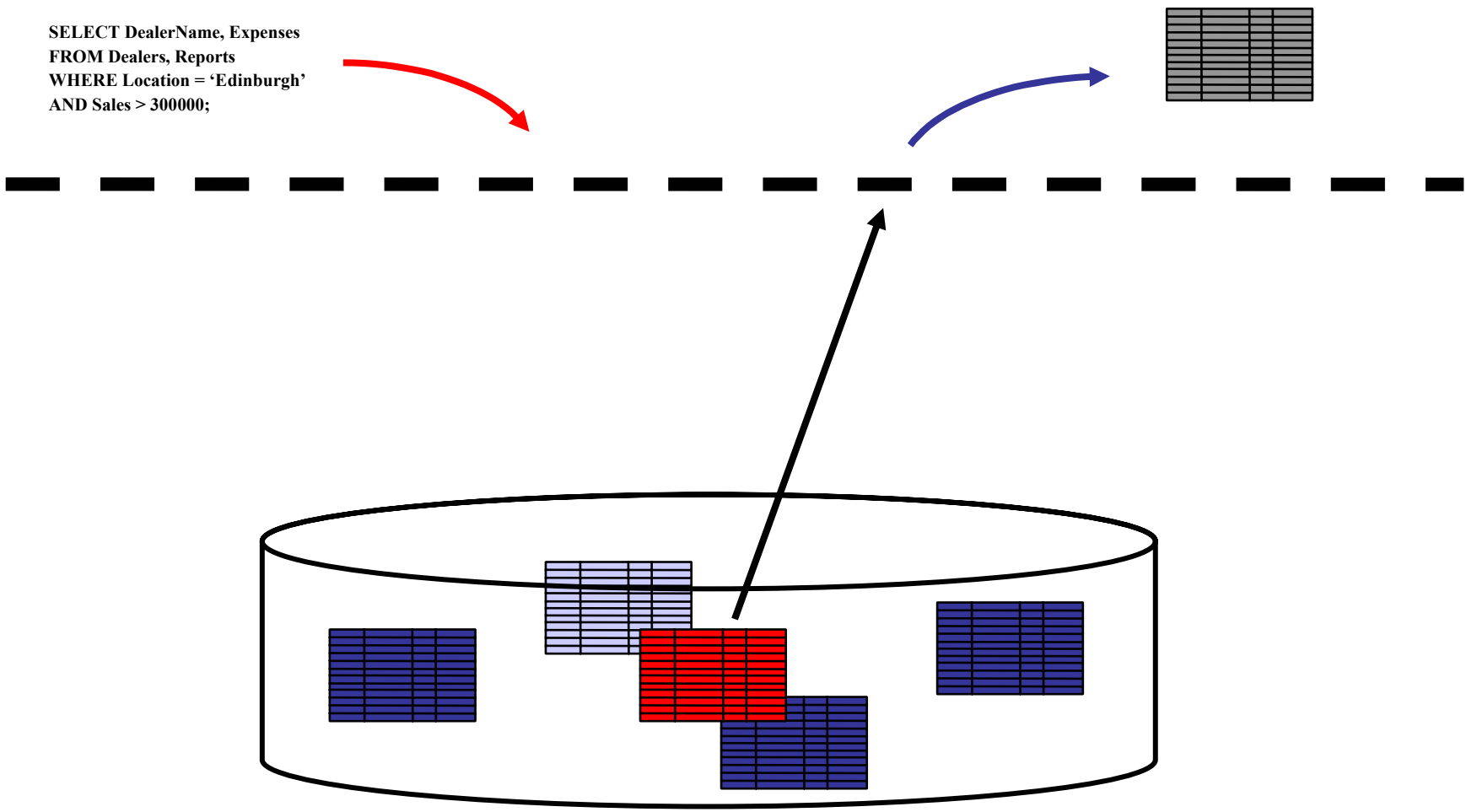
… Can We Reduce Response Times?



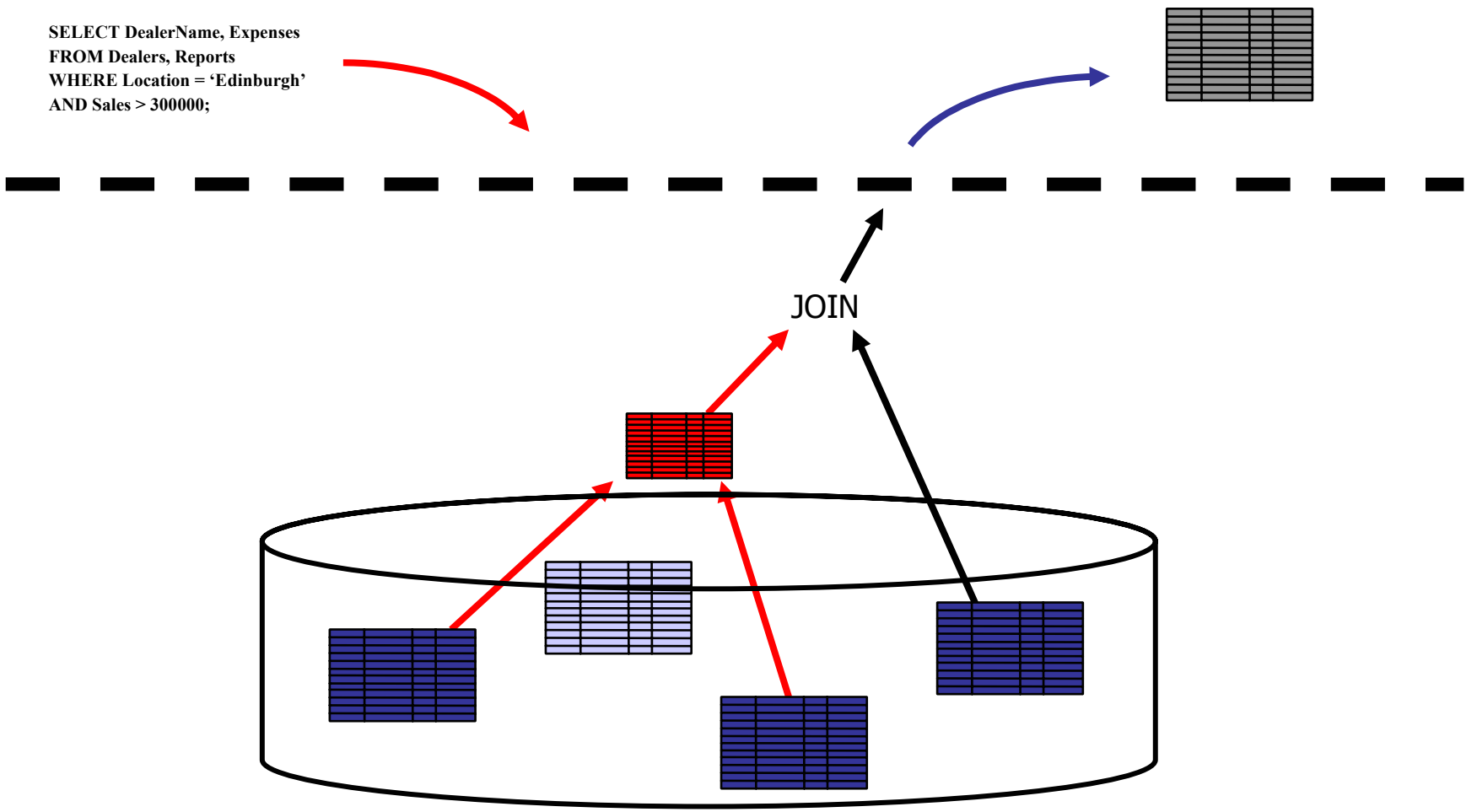
Computing a Complex Query



Can Be Done with Maximum Speedup



Or with at Least Some Speedup



Our Goals and Approach

- Improving query-evaluation performance
- By using derived data (materialized views)
- Working with a query workload at a time
- Method: designing and precomputing derived data in advance

Example: Rewriting Aggregate Queries [ACGL05]

Sales(CustID, DateID, ProductID, QtySold, TotalAmount, Discount)
Customer(CustID, CustName, Address, City, State, RegistrDateID)
Time(DateID, Month, Year)

```
Q1: SELECT c.CustID, SUM (QtySold)
    FROM Sales s, Customer c, Time t
    WHERE s.DateID = t.DateID
    AND s.CustID = c.CustID
    AND Month >= 10 AND Month <= 12
    AND Year = 2004
    GROUP BY c.CustID;
```

```
Q2: SELECT t.Year, MAX (QtySold)
    FROM Sales s, Customer c, Time t
    WHERE s.DateID = t.DateID
    AND s.CustID = c.CustID
    AND Year > 1997
    AND State = 'NC'
    GROUP BY t.Year;
```

Defining a View on the *Sales* Relation

```
Q1: SELECT c.CustID, SUM (QtySold)
    FROM Sales s, Customer c, Time t
    WHERE s.DateID = t.DateID
    AND s.CustID = c.CustID AND Month >= 10
    AND Month <= 12 AND Year = 2004
    GROUP BY c.CustID;
```

```
Q2: SELECT t.Year, MAX (QtySold)
    FROM Sales s, Customer c, Time t
    WHERE s.DateID = t.DateID
    AND s.CustID = c.CustID
    AND Year > 1997 AND State = 'NC'
    GROUP BY t.Year;
```

```
V1: SELECT CustID, DateID,
    SUM(QtySold) AS SumQS,
    MAX(QtySold) AS MaxQS
    FROM Sales
    GROUP BY CustID, DateID;
```

```
R1: SELECT c.CustID, SUM(SumQS)
    FROM V1, Customer c, Time t
    WHERE V1.DateID = t.DateID
    AND V1.CustID = c.CustID
    AND Month >= 10 AND Month <= 12
    AND Year = 2004
    GROUP BY c.CustID;
```

V1 can also be used to evaluate the query Q2

Problem: View Selection For Queries in Presence of Aggregation

- Input:
  - a workload ***Q*** of queries (some may be without aggregation)
  - an oracle ***O*** that gives view sizes
  - a constraint ***L*** on the views to be materialized
- Output: a set of views, with or without aggregation, that
  - satisfies the constraint ***L,***
  - produces equivalent rewritings of the queries in ***Q***, and
  - minimizes the total evaluation costs of ***Q***

The Setting

- Queries, views, and rewritings: SQL select-project-join (*conjunctive*) with equality selection conditions, with or without aggregation (without HAVING)
- Aggregate functions: *max, min, sum, count*
- Sum cost model for query evaluation
- Constraint on views: storage limit
- Rewriting format: *central* rewritings

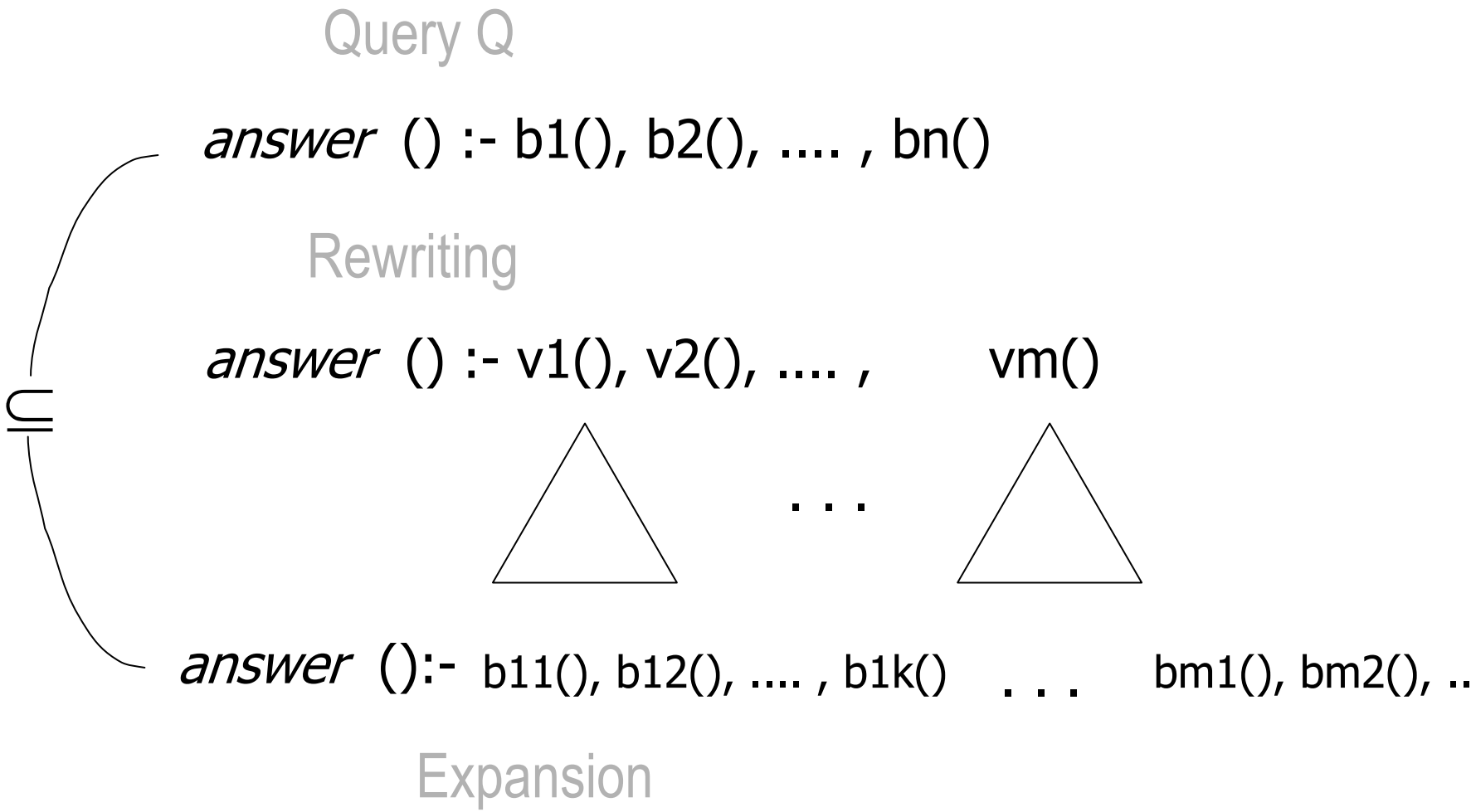
Why Consider Rewriting Formats

Example (5.1 in [CNS99])

Q(x,count) :- P(x,y).
V(x,count) :- P(x,s), P(x,t).
R(x, √z) :- V(x,z).

R is an equivalent rewriting of Q using V [CNS99, NSS98].

Rewriting Queries Using Views



Query Equivalence Modulo Views

Definition. *Let Q be a query defined on a database schema S, and let* ***V*** *be a set of views defined on S; let R be a query defined using the views in* ***V****. Then Q and R are* equivalent modulo ***V*** *if and only if for any database D, Q(D) is equivalent to* $R(D_{\mathbf{V}})$.

Here, $D_{\mathbf{V}}$ is the database obtained by computing all the view relations in **V** on D.

Evaluating an Aggregate Query Q On Database D

1. Compute the *core* Q' of Q on D as a bag B.
2. Grouping: Form equivalence classes { B' } in B based on the grouping arguments of Q.
3. Aggregation: With each equivalence class B', associate a value that is the aggregate function of Q computed on a bag of all values of the input argument of the aggregated attribute of Q in B'.

Equivalence between Aggregate Queries [NSS98]

- Definition. *Two aggregate queries are* compatible *if the tuples of arguments in their heads are identical.*
- Definition. *For two compatible aggregate queries Q and Q', Q and Q' are* equivalent *if Q(D) = Q'(D) for every database D.*

Classes of Aggregate Functions

- An aggregate function α is *duplicate-insensitive* [GHQ95] if the result of α computed over a bag of values is the same as the result of α computed over the core set of the bag. Otherwise α is *duplicate sensitive.*
- An aggregate function α is *distributive* [GCB+97] if there is a function γ such that

$$\alpha(\{\{ X_{ij} \}\}) = \gamma(\{\{ \alpha(\{\{ X_{ij} \}\} \mid i = 1, \ldots, I) \mid j = 1, \ldots, J \}\})$$

Results on Equivalence of Compatible Aggregate Queries [CNS99, NSS98]

- Two conjunctive queries are bag-set equivalent iff they are isomorphic after duplicate subgoals are removed.
- Equivalence of sum- and count-queries can be reduced to bag-set equivalence of their cores.
- Equivalence of max- and min-queries can be reduced to set equivalence of their cores.

Central Rewritings and Unfoldings

$R\ (x_1, \ldots, x_n, \alpha(y))$:- $V_0\ (x_{0,1}, \ldots, x_{0,n0}, y)$,
$V^b_1\ (x_{1,1}, \ldots, x_{1,n1}, y_1)$, … ,
$V^b_k\ (x_{k,1}, \ldots, x_{1,nk}, y_k)$.

$R^u\ (x_1, \ldots, x_n, \beta(y))$:- $B_{v0}, B_{v1}, \ldots, B_{vk}$.

Each of α, β can be one of max, min, sum, count, identity.
Thus, three types of central rewritings: CQ/CQA, CQA/CQ, CQA/CQA.

Central Rewriting CQA/CQA In the *Sales* Example

Q_1 (X, sum(Y)) :- Sales (X, Z, T_1, Y, T_2, T_3), Time (Z, W_1, 2004),
Customer (X, U_1, U_2, U_3, U_4, U_5), $W_1 \geq 10$, $W_2 \leq 12$.

V_1 (X, Z, sum(Y), max(Y)) :- Sales (X, Z, T_1, Y, T_2, T_3).

R_1(X, sum(P)) :- V_1 (X, Z, P, S), Time (Z, W_1, 2004),
Customer (X, U_1, U_2, U_3, U_4, U_5), $W_1 \geq 10$, $W_2 \leq 12$.

Central Rewriting CQ/CQA In the *Sales* Example

Q_1 (X, sum(Y)) :- Sales (X, Z, T_1, Y, T_2, T_3), Time (Z, W_1, 2004),
Customer (X, U_1, U_2, U_3, U_4, U_5), $W_1 \geq 10$, $W_2 \leq 12$.

V_2 (X, Z, Y) :- Sales (X, Z, T_1, Y, T_2, T_3).

R_2(X, sum(Y)) :- V_2 (X, Z, Y), Time (Z, W_1, 2004),
Customer (X, U_1, U_2, U_3, U_4, U_5), $W_1 \geq 10$, $W_2 \leq 12$.

Central Rewriting CQA/CQ In the *Sales* Example

Q_1 (X, sum(Y)) :- Sales (X, Z, T_1, Y, T_2, T_3), Time (Z, W_1, 2004),
Customer (X, U_1, U_2, U_3, U_4, U_5), $W_1 \geq 10$, $W_2 \leq 12$.

V_3 (X, sum(Y), max(Y)) :- Sales (X, Z, T_1, Y, T_2, T_3).

R_3(X, P) :- V_3 (X, P, S), Time (Z, W_1, 2004),
Customer (X, U_1, U_2, U_3, U_4, U_5), $W_1 \geq 10$, $W_2 \leq 12$.

Equivalence of Rewritings and Unfoldings CQA/CQA: Positive Result

Theorem. *Let R be a CQA/CQA rewriting. Suppose that all noncentral views of R are without aggregation and are bag-valued. Then R is equivalent to its unfolding* R^u.

This result holds for aggregate functions max, min, sum, count.

Central Rewriting CQA/CQA: Set- or Bag-Valued Views?

R (X, sum(Y)) :- V (X, Y), W (X).

V (X, count(*)) :- P (X, U).

W (X) :- S (X, Z).

R^u (X, count(*)) :- P (X, U), S (X, Z).

P: A B

1	2

S: A C

1	3
1	4

If W(D) is computed as a set: R(D) = { (1, 1) }, R^u(D) = { (1, 2) }

If W(D) is computed as a bag: R(D) = { (1, 2) }, R^u(D) = { (1, 2) }

Equivalence of Rewritings and Unfoldings CQA/CQA: Negative Result

Proposition. *Let R be a CQA/CQA rewriting with central aggregation* sum *or* count. *Suppose that*

- *R has a noncentral view with aggregation, or*
- *R has a set-valued noncentral view with nondistinguished arguments.*

Then R is not set-*equivalent to its unfolding* R^u.

Terminology for *Find-R*

- A central view V is *grouping complete* w.r.t. a rewriting R if the set of grouping attributes of V is a subset of the grouping attributes of R; otherwise V is *grouping incomplete.*
- Given an aggregate view V, its *reduced-core view* V^r is a view without aggregation whose body is the body of V and whose head attributes are all grouping attributes of V.
- Given a rewriting R, its *reduced-core rewriting* R^r is a rewriting without aggregation whose body uses only reduced-core views of R and whose head attributes are all grouping attributes of R.

Result underlying *Find-R*

Proposition. *Let R be an equivalent central rewriting of a query Q using views* $\mathbf{V}$*, and let the central view of R be CQA. Let* R^r *be a reduced-core rewriting of R, and* $\mathbf{V}^r$ *the reduced-core views of* $\mathbf{V}$*. Then* R^r *is an equivalent rewriting of the reduced-core query of Q using the views* $\mathbf{V}^r$*.*

Algorithm for Constructing Equivalent Rewritings of a Query using Views

Procedure *Find-R*. Input: query Q, set of views **V**

{ Consider Q^r, $\mathbf{V}^r$.
  Find all rewritings of Q^r using $\mathbf{V}^r$.
    For each rewriting R^r do:
    { Consider the expansion $R^{r\text{-}exp}$.
      For each containment mapping from Q^r to $R^{r\text{-}exp}$ do:
      { If there is a view *V* in the rewriting such that its aggregated
        attribute is the image of the aggregated attribute of the query, do:
        { Call V the central view.
          If the central view is grouping-incomplete then construct
            a CQA/CQA rewriting.
          If the central view is grouping-complete then construct
            a CQA/CQ rewriting.
        }
      }
    }
}

Correctness Result for *Find-R*

Theorem. *If there is a central rewriting of a query Q using views **V**, then procedure Find-R will find a rewriting. Moreover, it will find a CQ rewriting if there exists one.*

Problem: View Selection

- Input:
  - a workload ***Q*** of queries (some may be without aggregation)
  - an oracle ***O*** that gives view sizes
  - a storage-limit constraint ***L*** on the views to be materialized
- Output: a set of views, with or without aggregation, that
  - satisfies the storage limit ***L,***
  - produces equivalent central rewritings of the queries in ***Q***, and
  - minimizes the total evaluation costs of ***Q***

Cost Model for Query Evaluation

- No cost on computing the given views.
- Size of a database relation is the number of tuples in it.
- Cost of a join: sum of the sizes of the input relations and of the output relation (*cf.* hash joins).
- Cost of evaluating conjunctive queries: sum of the costs of all the joins, assuming left-linear join trees.
- Sum cost model for aggregate queries: sum of the costs of
  - computing the conjunctive core of the query,
  - performing the grouping step, and
  - performing the aggregation step.

View Selection: Decidability

Theorem. *The decision version of the view-selection problem under the storage limit is decidable for finite query workloads and for conjunctive views and rewritings, with or without aggregation, for the three central rewriting types.*

View Selection for Sum and Count: The Problem Is NP-Complete

Theorem. *The decision version of the view-selection problem under the storage limit is NP-Complete for finite workloads of queries with* sum *or* count *aggregation and for conjunctive views and rewritings, with or without aggregation, for the three central rewriting types.*

Returning to an Earlier Example

Q (X, count(*)) :- P (X, Y), S (X, Z).

R_1 (X, sum(U)) :- V (X, U), W^b (X).

V (X, count(*)) :- P (X, Y).
W^b (X) :- S (X, Z).

R^u_1 (X, count(*)) :- P (X, Y), S (X, Z). Q is equivalent to R^u_1

R_2 (X, sum(U)) :- V (X, U), W^b (X), W^b (X).

R^u_2 (X, count(*)) :- P (X, Y), S (X, Z), S (X, T).

Q is *not* equivalent to R^u_2

View Selection for Max and Min: Exponential-Time Lower Bound

Theorem. *The view-selection problem under the storage limit has an exponential-time lower bound for finite workloads of queries with* max *or* min *aggregation and for conjunctive views and rewritings, with or without aggregation, for the three central rewriting types.*

Back to the Example

Q (X, max(Y)) :- P (X, Y), S (X, Z).

R_1 (X, max(U)) :- V (X, U), W^b (X).

V (X, max(Y)) :- P (X, Y).
W^b (X) :- S (X, Z).

R^u_1 (X, max(Y)) :- P (X, Y), S (X, Z). Q is equivalent to R^u_1

R_2 (X, max(U)) :- V (X, U), W^b (X), W^b (X).

R^u_2 (X, max(Y)) :- P (X, Y), S (X, Z), S (X, T).

Q *is* equivalent to R^u_2

Terminology for *Design-Views* (I)

- A view V is a *central minimal view* for a query Q if the body of V is the subset of subgoals of Q that is in the body of the definition of any central view that is used in an equivalent central rewriting of Q.
- A view W is a *collective noncentral view* for Q if the body of W has all the subgoals of Q that provide the grouping arguments of Q that do not come from the central minimal view of Q.
- A pair (V,W) as above is the *characteristic views* for Q.

Terminology for *Design-Views* (II)

- Two pairs of characteristic views, (V_1, W_1) for query Q_1 and (V_2, W_2) for query Q_2, are *compatible* if
  - The two central views V_1 and V_2 can be combined into a single multiaggregate view V_m, and
  - V_m can be used to rewrite both Q_1 and Q_2 .

Results underlying *Design-Views* (1 of 2)

Proposition. *Let R be a central (CQA or CQ) rewriting of a query Q. Then*

1. *All the subgoals of Q that contain the aggregated attribute of Q are also subgoals of the central view of R, and*
2. *Each grouping attribute of Q is a grouping attribute of the central view of R or is in the head of one of the noncentral views in R.*

Results underlying *Design-Views* (2 of 2)

Proposition.

1. *Each query Q has a finite number of characteristic views.*
2. *In any central rewriting R of Q, the views used in R can also be used to produce central rewritings of characteristic views of Q.*
3. *It is decidable to tell whether two pairs of characteristic views are compatible.*

Algorithm for Designing Central Views

Procedure *Design-Views*.

Input: query workload **Q.**

{ For each query Q in **Q** do:

Construct characteristic views for Q.

For each combination of characteristic views do:

Find multiaggregate views for **Q** by finding compatible pairs of characteristic views.

Return a maximal set of characteristic central views.

}

Correctness of *Design-Views*

Theorem. *Given a query workload **Q**, the procedure Design-Views finds all maximal multiaggregate views for **Q**.*

What this Talk Was About

- Topic: using materialized views to improve query-evaluation performance
- Context:
  - Workloads of queries with or without aggregation
  - Designing optimal views under constraints
- Results:
  - Formats of equivalent rewritings of queries using views: sufficient and necessary conditions
  - Designing and using views: complexity and algorithms

Some Open Problems and Possible Extensions

- Using rewritings of aggregate queries to solve practical problems
  - Range-aggregate star-schema queries [ACGL05]
- Conjecture:
  - The rewriting templates of [CNS99] and of [AC05] are the "only" templates for *equivalent* rewritings of conjunctive queries with sum, count aggregation

(Assuming conjunctive views and rewritings with or without aggregation)

Thanks!

Related Work

- [NSS98] Cohen, S., Nutt, W., and Serebrenik, A.: "Rewriting aggregate queries using views," in *Proceedings of PODS,* 1999.
- [CNS99] Nutt, W., Sagiv, Y., and Shurin, S. W.: "Deciding equivalences among aggregate queries," in *Proceedings of PODS,* 1998.
- [CNS00] Cohen, S., Nutt, W., and Serebrenik, A.: "Algorithms for rewriting aggregate queries using views," in *Proceedings of ADBIS-DASFAA,* 2000.
- [GHQ95] Gupta, A., Harinarayan, V., and Quass, D.: "Aggregate-query processing in data warehousing environments." In *Proceedings of VLDB,* 1995.
- [SDJL96] Srivastava, D., Dar, S., Jagadish, H.V., and Levy, A.Y. "Answering queries with aggregation using views," In *Proceedings of VLDB,* 1996.
- [GCB+97] Gray, J., Chaudhuri, S., Bosworth, A. et al.: "Data cube: A relational aggregation operator generalizing Group-by, Cross-tab, ans Sub totals." *Data Mining and Knowledge Discovery,* 1(1):29-53, 1997.
- [RSSS98] Ross, K.A., Srivastava, D., Jagadish, H.V., and Sudarshan, S.: "Foundations of aggregation constraints." *Theoretical Computer Science,* 193(1-2): 149-179, 1998.
- [ALU01] Afrati, F., Li, C., and Ullman, J.D.: "Generating efficient plans for queries using views." In *Proceedings of SIGMOD,* 2001.
- [ACGL05] Afrati, F., Chirkova, R., Gupta, S., and Loftis, C.: "Designing and using views to improve performance of aggregate queries," in *Proceedings of DASFAA,* 2005.