

# Heterogeneous / Federated / Multi-Database Systems

Dr. Denise Ecklund  
9 October 2002

©2002 Vera Goebel & Denise Ecklund

HDBMS-1

## Contents: Heterogeneous DBSs

- Motivation
  - Applications for Heterogenous Database Systems (HDBS)
- What is a HDBMS?
- Architectures for HDBS
- Main Problems:
  - Defining a Global Data Model
  - Query Processing & Optimization
  - Transaction Management
- Summary and Conclusion

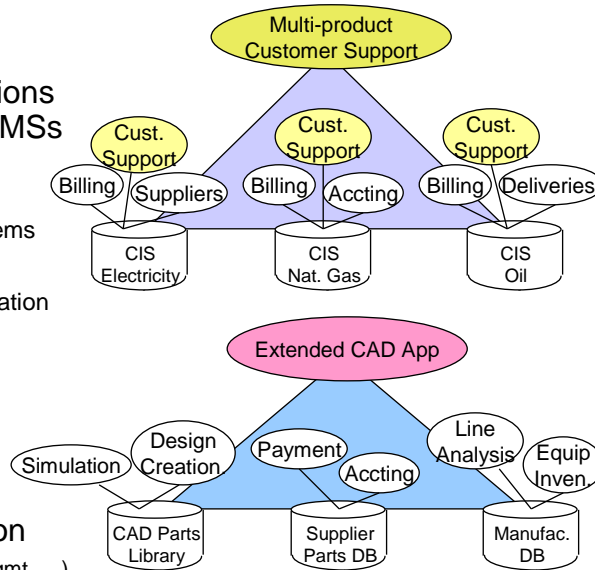
Pensum: Garcia-Molina/Ullman/Widom: Section 20.1, 20.2, 20.3  
and this set of presentation slides.

©2002 Vera Goebel & Denise Ecklund

HDBMS-2

## Applications

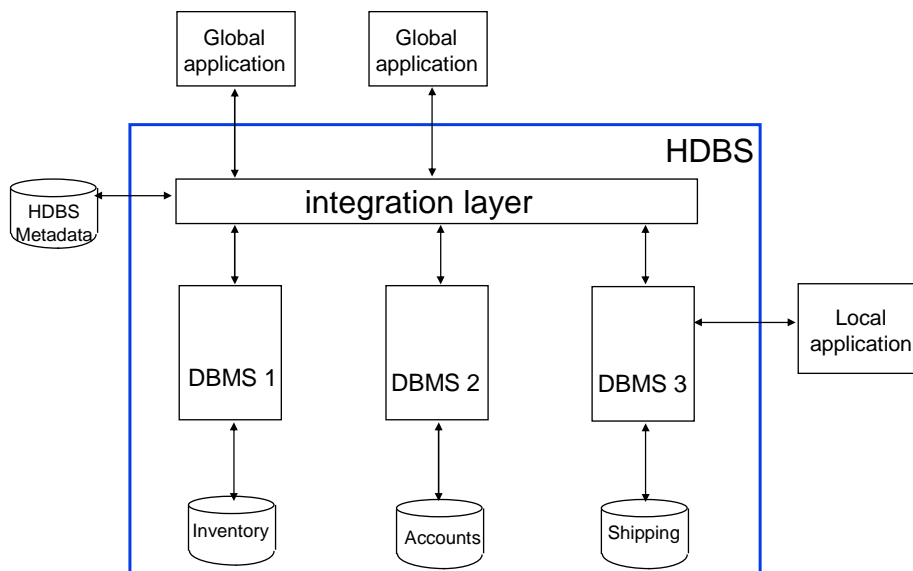
- Multitude of extensive, isolated data agglomerations managed by different DBMSs or file systems
  - Similar data
    - Ex: 3 Customer Info Systems
  - Dissimilar data
    - Ex: Extended CAD Application
- Extension of data and management software because of new and/or extended applications
- Heterogeneous application domains (e.g., CIM, CAD, Biz-mgmt, ...)



©2002 Vera Goebel & Denise Ecklund

HDBMS-3

## Heterogeneous Database Systems (HDBS)



©2002 Vera Goebel & Denise Ecklund

HDBMS-4

## Requirements for HDBS

- **Properties known from homogeneous DBS:**
  - global data model, transactions, recovery, dist transparency, ...
- **Integration of Heterogeneous Data Stores**
  - > queries across HDBs (combine heterogeneous data)
  - > heterogeneous information structures
  - > avoid redundancy (data and data model)
  - > access (query) language transparency
- **“Open” system**
  - support for integration of existing data models and DBSs, as well as their schemas and DBs
- **Constraints**
  - > retain autonomy of DBS to be integrated
  - > avoid modifications of existing local applications
  - > define a viable global data model for global applications

©2002 Vera Goebel & Denise Ecklund

HDBMS-5

## Definition - Heterogeneous DBS (HDBS)

A HDBS comprises a software layer (integration layer) and multiple DBSs and/or file systems to be integrated.

Users can transparently access the integrated DBSs and/or file systems via the interface provided by the integration layer.

Defines a global data model

Supports a Data Definition Language (DDL)

Supports a Data Manipulation Language (DML)

Distributed Transaction Management

Transparent integration of the underlying, disparate DBSs

**Complete  
Distributed Database  
Services**

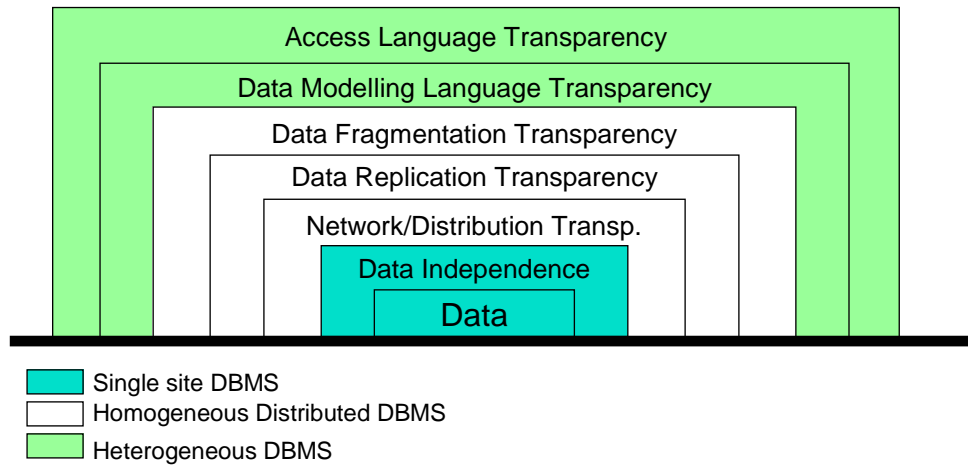
The integrated, local DBSs are autonomous and can also be used as stand-alone systems.

Local applications are unchanged and unknown to the HDBS.

©2002 Vera Goebel & Denise Ecklund

HDBMS-6

## Layers of Transparency



©2002 Vera Goebel & Denise Ecklund

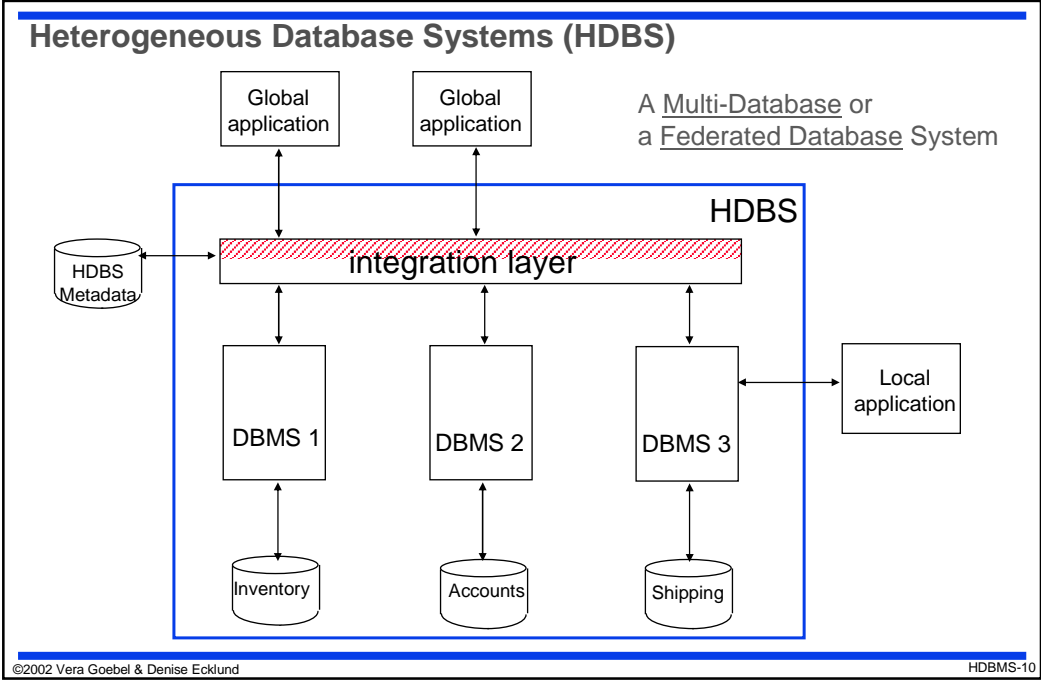
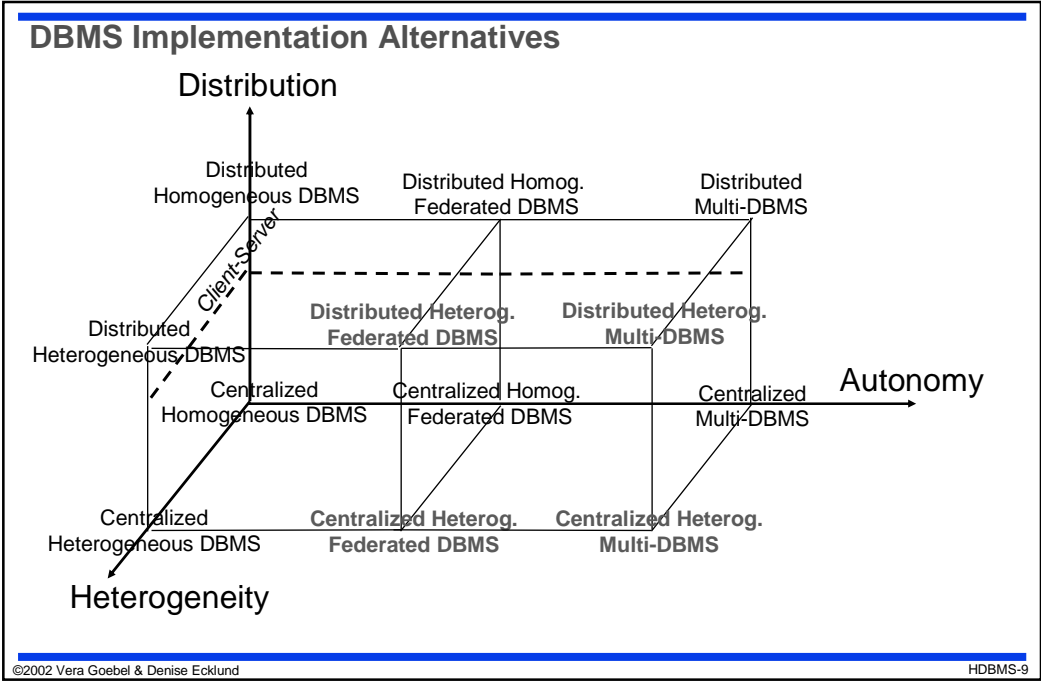
HDBMS-7

## Abstraction Levels [Christmann et al. 87]

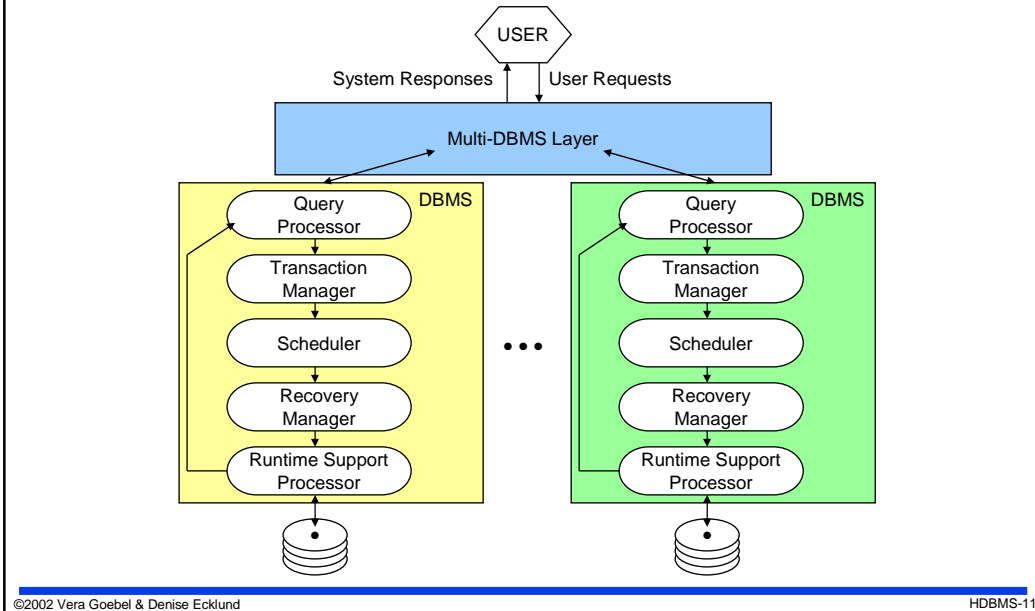
Abstraction Level	Supported By	Objects	
access & data model lang	global conceptual schema	relations or objects	Global Abstractions
fragmentation transparency	fragmentation schema	fragments of rels/objs	
replication transparency	replication schema	multiple copies of fragments of rels/objs	
network transparency	remote communication services	remotely located multiple copies of fragments	
logical data independence	local conceptual schema	local relations/objects	Local Abstractions
physical data independence	physical schema	records, access paths	
file system	file definitions and buffer management	physical records, pages	
storage and I/O system	disk storage definitions	tracks, physical blocks	

©2002 Vera Goebel & Denise Ecklund

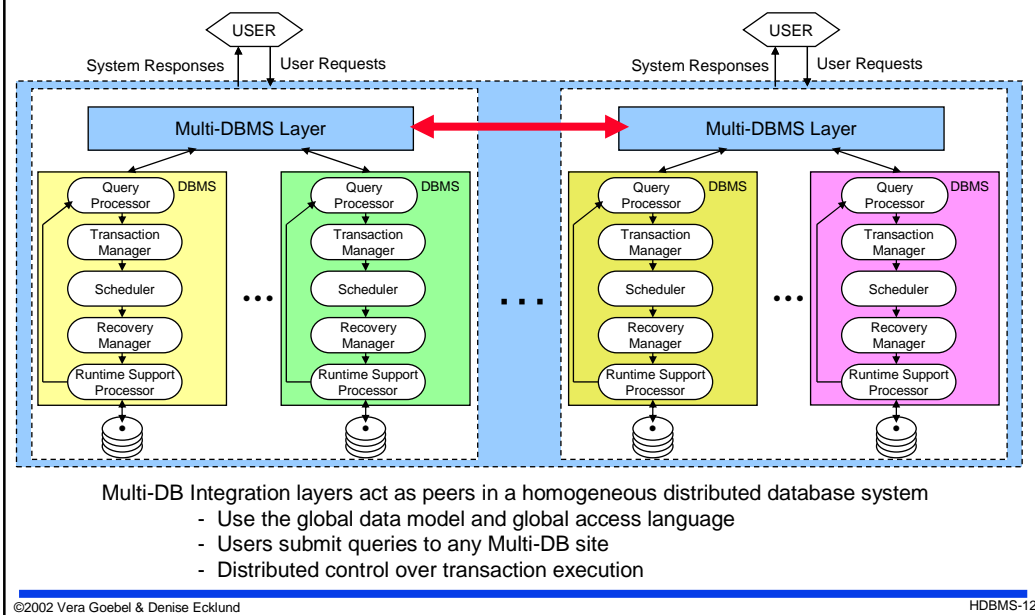
HDBMS-8



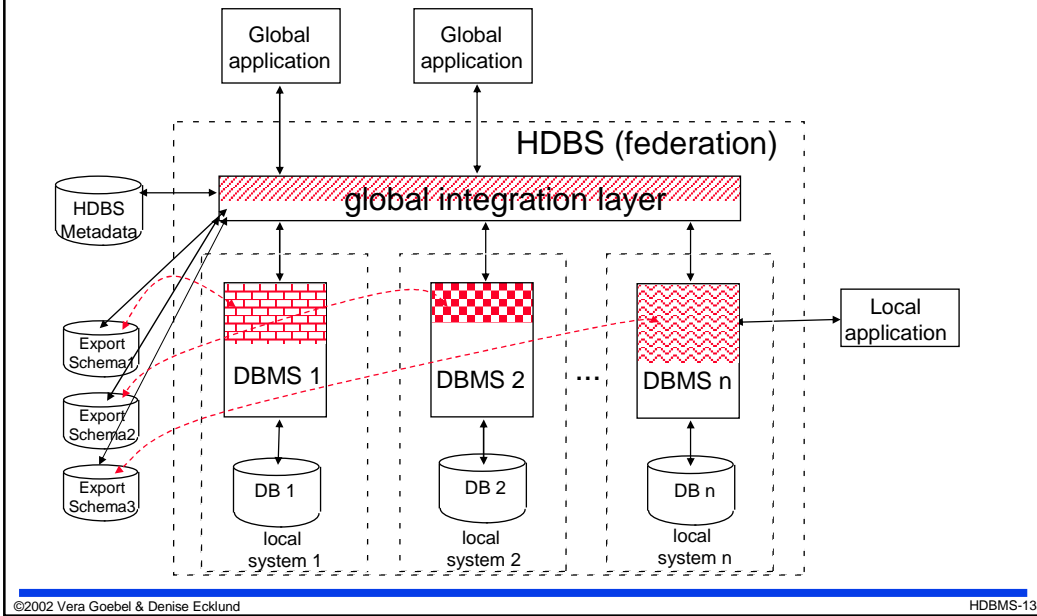
## Components of a Multi-DBMS



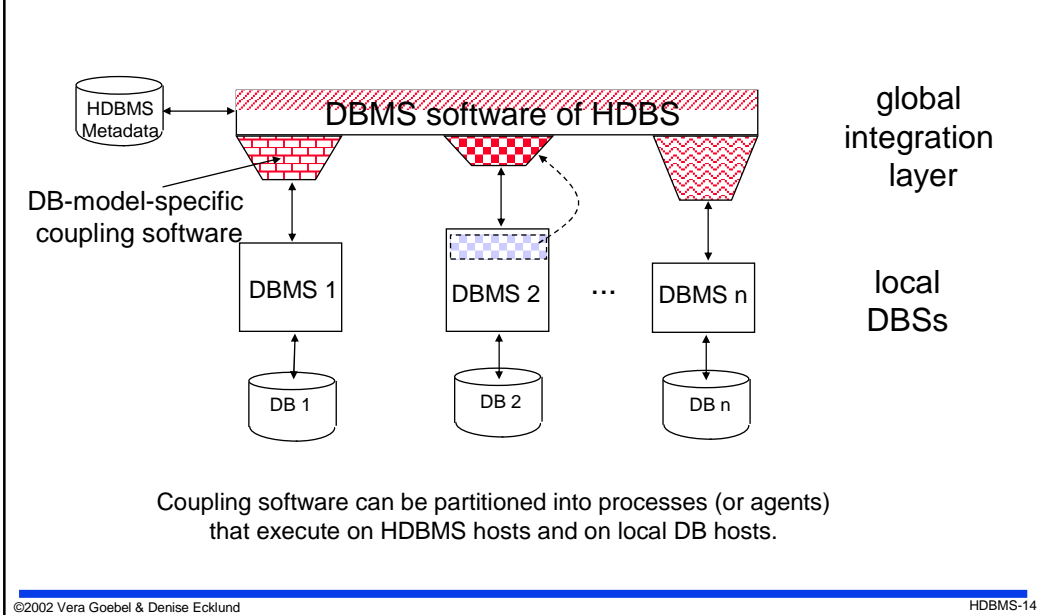
## Components of a Distributed Multi-DBMS



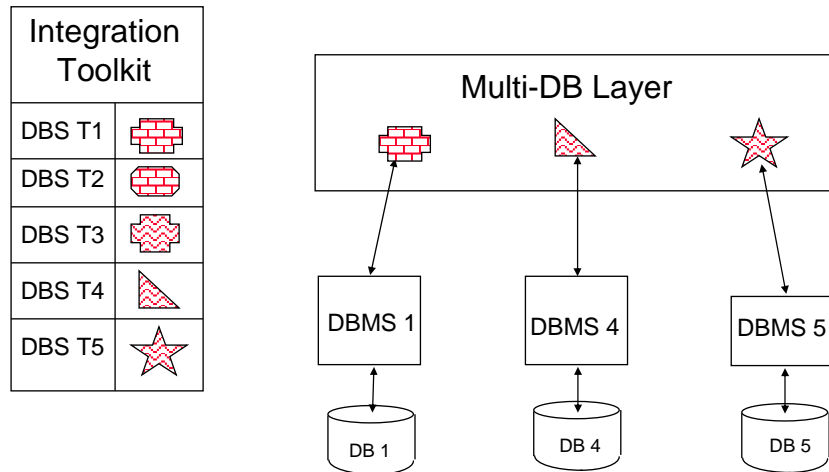
## HDBS Architecture



## Abstract Component Architecture of HDBS



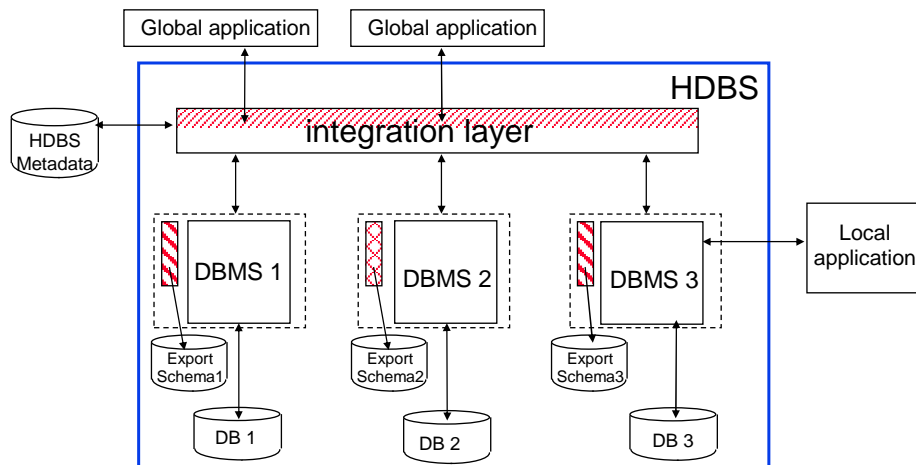
## Toolkits for HDBMS – an implementation approach




©2002 Vera Goebel & Denise Ecklund

HDBMS-15

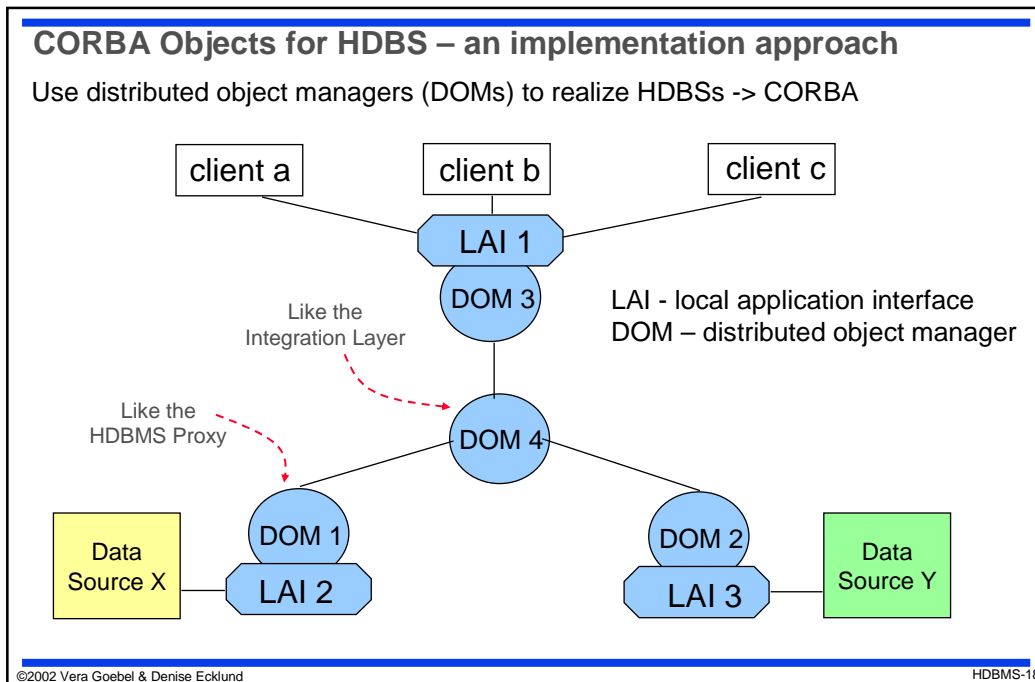
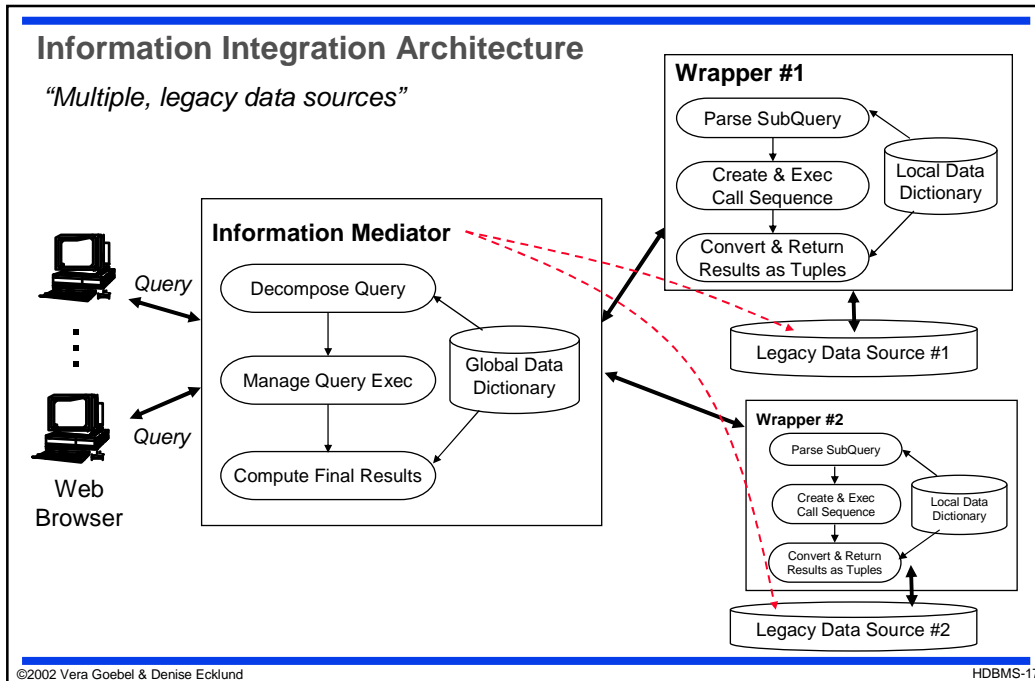
## Heterogeneous Database Systems (Fully-autonomous HDBS)



-  HDBS Server or HDBS Proxy
- Runs on the local DB site
  - Typically includes some code that is specific to the local DB type

©2002 Vera Goebel & Denise Ecklund

HDBMS-16



## Concepts in the Integration Layer

- Global data model
- Global schema and meta data management
- Distributed query processing and optimization
- Distributed transaction management
- Extensible software construction  
(to allow the “easy” integration of additional system components)

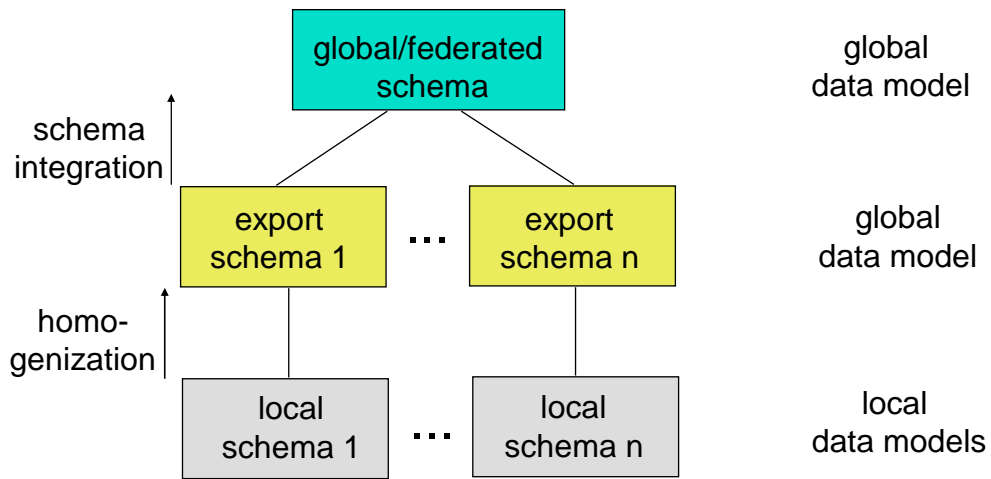
## Data Model

- Local data models: any kind of data model possible, e.g., object-oriented, relational, entity-relationship, hierarchical, network-oriented, flat files, ...
- Global data model: must comprise modeling concepts and mechanisms to express the features of the local data models
  - When integrating N local data models, use the “richest” model of the N models you are integrating
  - Object-oriented data models
    - Provide user-defined data types and methods
    - Are often used as the global (integration) data model

Goals - To define a data model that:

- 1) Is a complete, minimal, and understandable data model for the union of the data stored in the set of local data bases (*application development time*)
- 2) Support application queries that can be satisfied by retrieving data from the set of local data bases (*application runtime*)

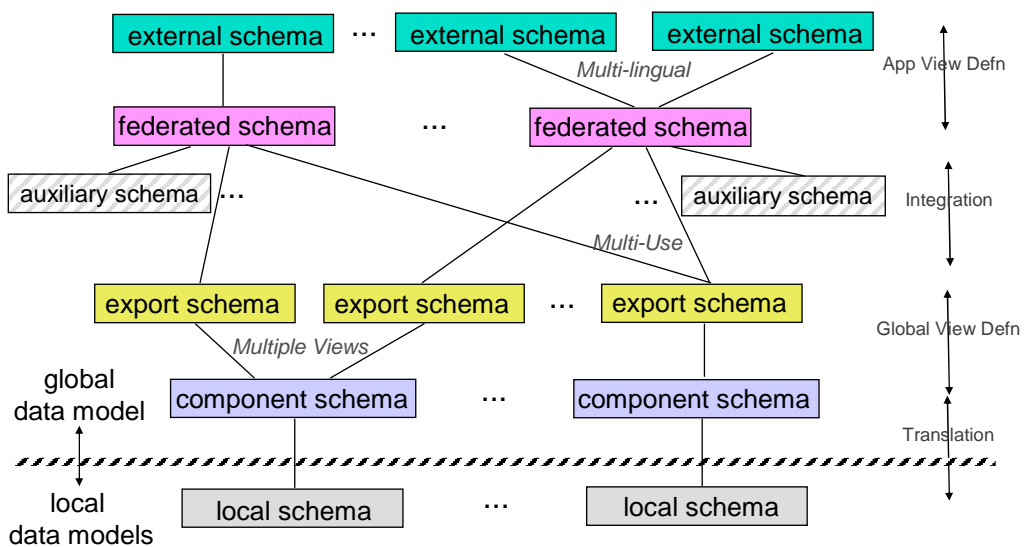
## Schema Architecture of HDBS



©2002 Vera Goebel & Denise Ecklund

HDBMS-21

## Schema Architecture of HDBS - 2 5-layer schema architecture



©2002 Vera Goebel & Denise Ecklund

HDBMS-22

## Schema Homogenization

- Schema Translation

- Map each local schema to the language of the global data model
  - Ex: a Relational schema to an Object-oriented schema

- Schema Integration

- For  $N$  translated, local schemas
  - Pairwise integration, X-at-a-time integration, One-step integration
- Determine "common semantics" of the schemas

Adequate design tools are not available



- Make the "same things" be "one thing" in the integrated schema
- Resolve conflicts
  - structural and semantic

## Schema Conflicts

- Name

- Different names for equivalent entities, attributes, relationships, etc.
- Same name for different entities, attributes, ...

- Structure

- Missing attributes
- Missing but implicit attributes

- Relationship

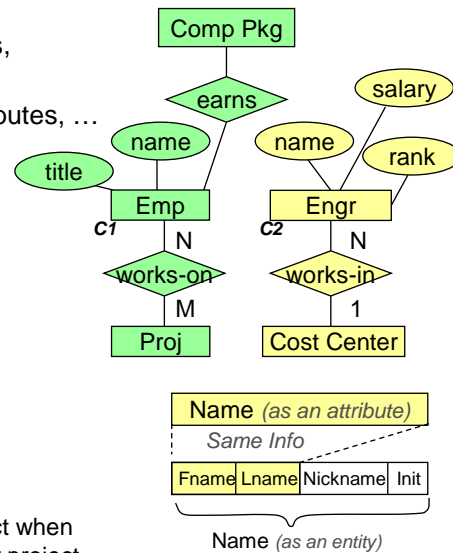
- One-to-many, many-to-many

- Entity versus Attribute (inclusion)

- One attribute or several attributes

- Behavior

- Different integrity constraints
  - Ex: automatic update, delete a project when the last engineer is moved to another project



## Data Representation Conflicts

- Different representation for equivalent data
  - Different units
    - Celsius ↔ Fahrenheit; Kilograms ↔ Pounds; Liters ↔ Gallons;
  - Different levels of precision
    - 4 decimal digits versus 2 decimal digits
    - Floating point versus integer
  - Different expression denoting same information
    - Enumerated value sets that are not one-to-one
      - {good, ok, bad} versus {one, two, three, four, five}

How to Resolve Schema Conflicts?  
Can Object-Oriented Models Help?

©2002 Vera Goebel & Denise Ecklund

HDBMS-25

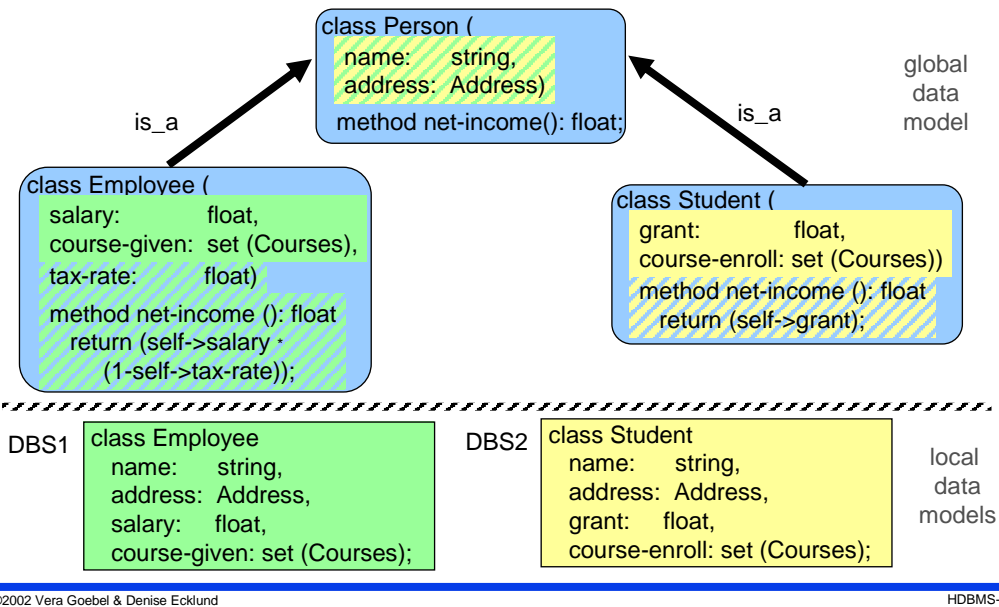
## Suitability of Object-Oriented Data Models as Global Data Models

- **Rich set of type constructors**
  - > easy representation of other data models
- **Extensibility** (user-defined types + type specific operators) & **Encapsulation**
  - > representation of “foreign” types/systems
  - > hiding heterogeneity (concrete storage) in a natural way
- **Inheritance** (generalization) & **computational completeness**
  - > schema integration
    - factor out common properties of similar types
    - thereby “arbitrary” computations possible

©2002 Vera Goebel & Denise Ecklund

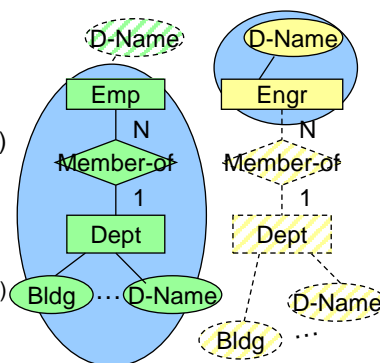
HDBMS-26

## Use of Generalization & Computational Completeness (Example)



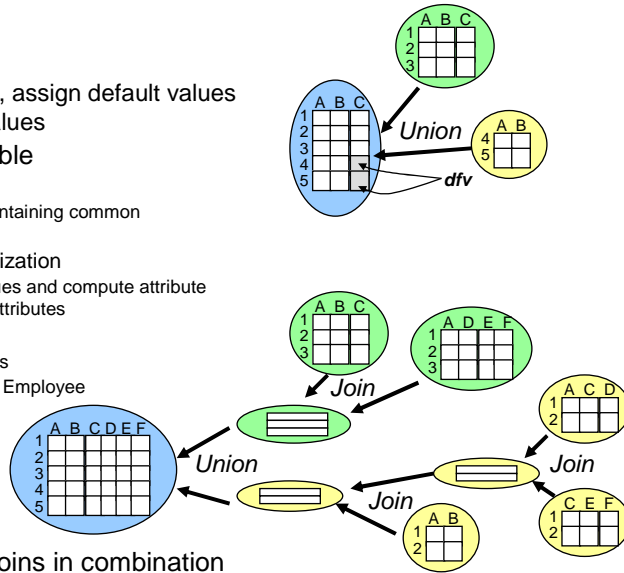
## Conflict Resolution

- Renaming entities and attributes
  - Pick one name for the same things
  - Use unique prefixes for different things
- Homogenizing representations
  - Use conversions and mappings
    - stored programs in relational systems
    - methods in OO systems
    - auxiliary schemas to store conversion rules/code
- Homogenizing attributes
  - Use type coercion (e.g., integer to float)
  - Attribute concatenation (e.g., first name || last name)
  - For missing attributes, assign default values
- Homogenizing an attribute and an entity
  - Extract an attribute from the entity
    - Ex: Project department name from the Dept entity to create a virtual attribute (e.g., Emp->Dept.D-Name)
  - Create an entity from the attribute
    - Ex: Define default values and behavior for all other attributes of the Dept entity



## Conflict Resolution

- Horizontal joins
  - Union compatible
    - For missing attributes, assign default values or compute implicit values
  - Extended union compatible
    - Use generalization
      - Define a virtual class containing common attributes
    - Subclasses of the generalization
      - Provide specialized values and compute attribute values for generalized attributes
    - See earlier example
      - class Person generalizes class Student and class Employee
- Vertical joins
  - Many and many to one
- Mixed Joins
  - Vertical and horizontal joins in combination

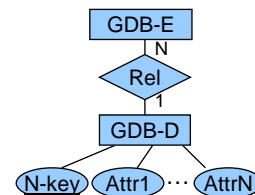
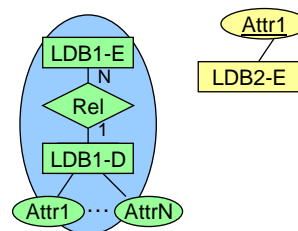


©2002 Vera Goebel & Denise Ecklund

HDBMS-29

## Conflict Resolution involving a Database Key

- Entity-Attribute Conflicts where the Attribute is a DB key in one local schema
- Example:
  - The global schema defines Attr1 as an entity
  - Attr1 is a DB key for instances of LDB2-E
- If Attr1 is a complete DB key in LDB2, then in the global schema
  - Define entities E and D and relationship Rel
  - Define a new DB key attribute that will be used to uniquely identify instances of LDB2-E when they are accessed through GDB-E and GDB-D



©2002 Vera Goebel & Denise Ecklund

HDBMS-30

## Conflict Resolution involving a Partial Database Key

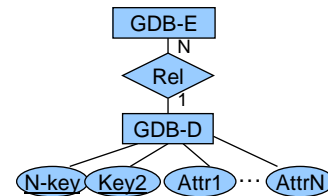
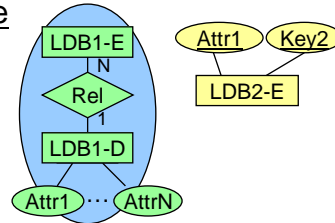
- Entity-Attribute Conflicts where the Attribute is a partial DB key in one local schema

- Example:

- The global schema defines Attr1 as an entity
- Attr1 is a partial DB key for instances of LDB2-E

- If Attr1 is a partial DB key in LDB2

- Define the entities E and D, and relationship Rel
- Define a new attribute as a partial DB key
- Add the other partial key attributes from LDB2 as partial keys
- Add partial DB key LDB2-Attr1 as an attribute only



©2002 Vera Goebel & Denise Ecklund

HDBMS-31

## Global Schema Management

- HDBS manages the global schema =  $\sum$  (all local exported schema)
- Global schema definition facilities provide mechanisms for handling the full spectrum of schematic differences that may exist among the heterogeneous local schemata.
  - Can use an Auxiliary Schema to store mappers, translators, and converters
- Data is stored in the local component systems.
- Global dictionary information is used to query and manipulate the data. The global language statements are translated into equivalent statements of the local languages supported by the local systems

©2002 Vera Goebel & Denise Ecklund

HDBMS-32

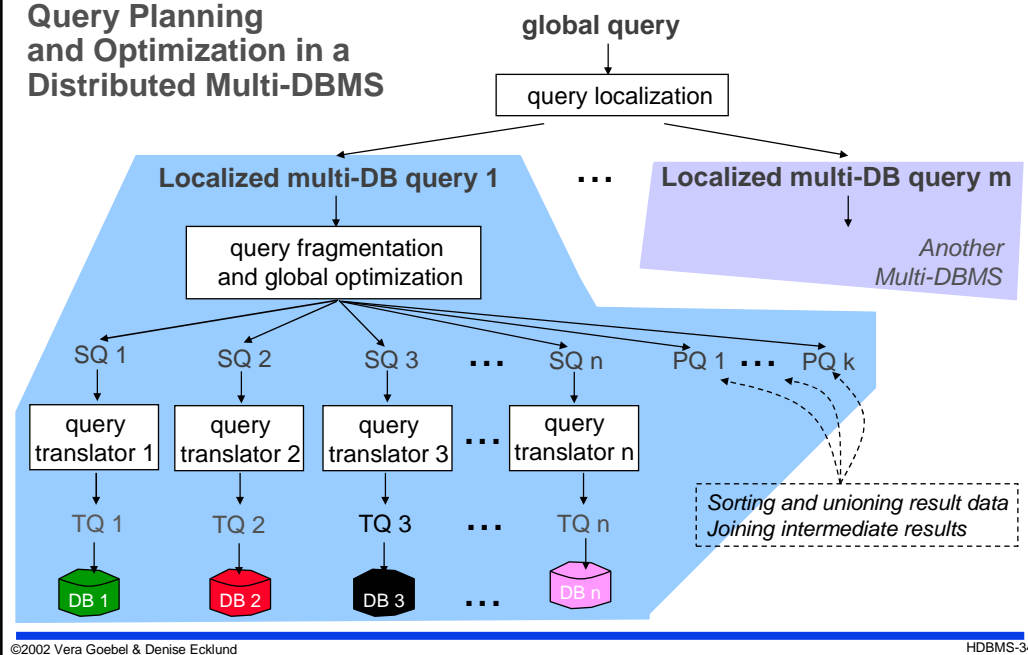
## Query Processing and Optimization

- The HDBMS has
  - A global Data Definition Language (DDL)
  - A global Data Manipulation Language (DML)
  - A set of local DMLs
- The HDBMS Query Processing Goal:
  - Given a query stated in the global query language (DML), execute that query, in an optimal manner, using the local database management systems

©2002 Vera Goebel & Denise Ecklund

HDBMS-33

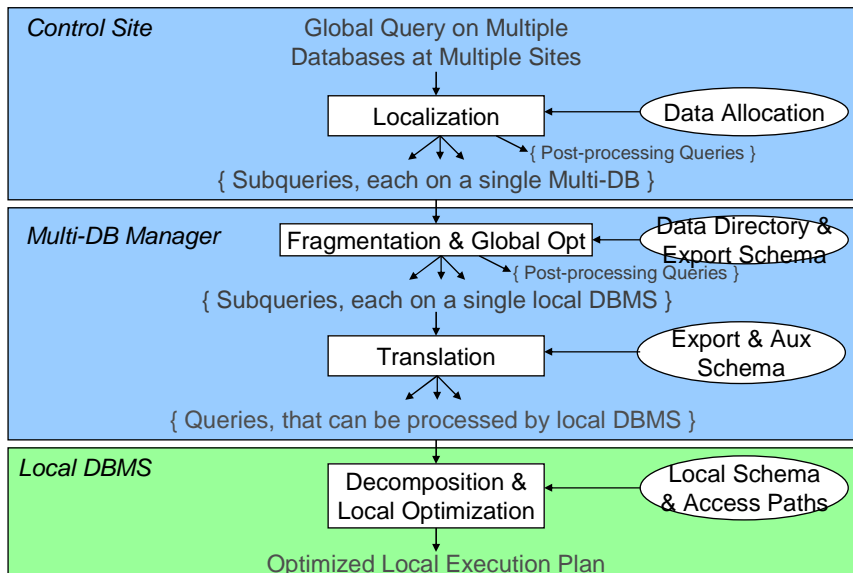
## Query Planning and Optimization in a Distributed Multi-DBMS



©2002 Vera Goebel & Denise Ecklund

HDBMS-34

## Information Supporting Query Planning & Optimization



©2002 Vera Goebel & Denise Ecklund

HDBMS-35

## Query Fragmentation

- Similar to query fragmentation problem for homogeneous distributed DBSs
- But ...Complicating factors:
  - Autonomy
    - Little information about “how” the subquery will be executed by the Local DBS
  - Heterogeneous Data Definition Languages
    - Weaker modeling languages do not support the same manipulation “features”
    - Must use multiple techniques in order to define a consistent global data model
    - Query fragmentation must produce a set of subqueries that reverse the operations used to create/define the global schema
- Processing Steps:
  - (1) Replace names from the global schema with “fullnames” from the export schemas
  - (2) If a subquery involves multiple export schemas, then break the query into queries that operate on one export schema and insert data communication operators to exchange intermediate results between local database systems

©2002 Vera Goebel & Denise Ecklund

HDBMS-36

## Global Query Optimization

- Similar to global query optimization for homogeneous distributed DBSs (many algorithms can be used directly)
- But only possible under the following assumptions:
  - No data inconsistency (the global schema correctly represents the semantics of disjoint, overlapping, and conflicting data)
  - Know the characteristics of local DBSs
    - e.g., statistical info on data cardinalities and selectivities are available
  - Can transfer partial data results between different local DBSs
    - Major impact on post-processing plans
- Primary Considerations:
  - Post-processing Strategy
  - Parallel Execution Possibilities
  - Global Cost Function/Estimation

©2002 Vera Goebel & Denise Ecklund

HDBMS-37

## Post-Processing Strategies

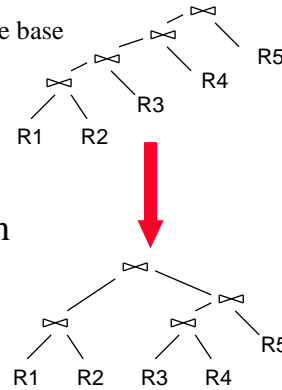
- Three Strategies:
  - 1) Control site performs all intermediate and post-processing operations (I&PP-ops)
    - Heavy work load; minimal parallelism
  - 2) Control site performs I&PP-ops for multi-DB results; Multi-DB managers, and HDBMS agents on the local database sites perform I&PP-ops for DBSs within one multi-DB environment
    - Better work load balance; more parallelism
  - 3) Use strategy #2 and use “pushdown” to get the local database systems to perform I&PP-ops
    - Possible if local DBMS can read intermediate results from external sources, and sort, join, etc. can be directly invoked

©2002 Vera Goebel & Denise Ecklund

HDBMS-38

## Parallel Execution Strategies

- Join operations are slow → speedup with parallel execution?
- Traditional query plans use left linear join trees
  - One of the operands is always a base relation
    - Have good info on cardinality and selectivity for the base
  - Used even in homogeneous distributed DBSs because cooperative nodes can pipeline the sequence of joins
- Bushy join trees provide parallel execution in heterogenous multi-DB environments
  - Convert a left linear join tree into a (balanced?) bushy join tree



©2002 Vera Goebel & Denise Ecklund

HDBMS-39

## Global Cost Estimation

- Differs from cost estimation in homogeneous distributed DBSs
  - Little (or no) info on QP algorithms in local DBSs and data statistics
- Cost Estimation Function
  - Cost to execute each subquery on the local DBSs
  - Cost to execute all I&PP-ops
    - via pushdown or by any HDBMS agent or proxy
- Use a simplified cost function

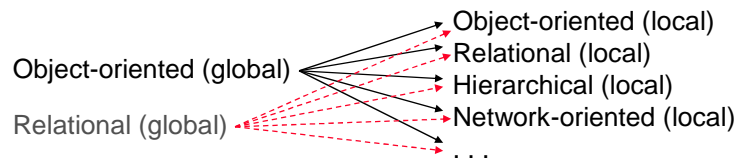
Cost = Initialization cost  
+ cost to retrieve a set of objects  
+ cost to process a set of objects
- Run test queries on the local DBSs to get time estimates for ops
  - Selection, with and without an index
  - Join (testing for different algorithms: sort, hash, or indexed based algs)

©2002 Vera Goebel & Denise Ecklund

HDBMS-40

## Query Translation

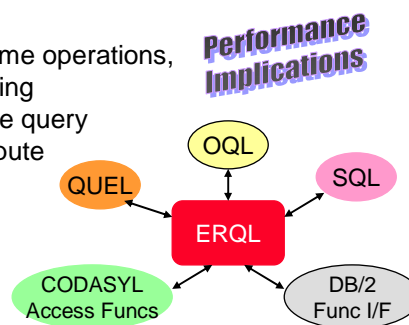
When a query language of a local DBS is different from the global query language, each export schema subquery for the local DB needs to be translated from the global language to the target language.



Weaker target languages do not support the same operations, so emulate required operations in post-processing

Ex: retrieve more data than requested by the query and then post-process that data to compute the correct response to the query

Reduce the number of language mappings using the Entity-Relationship Query Language as an intermediary language



©2002 Vera Goebel & Denise Ecklund

HDBMS-41

## Local Data Sources with Heterogeneous Capabilities

- "GenCompact" defines a simple language to describe query capabilities of the local DBSs
- Generates the "best" query plan for each local DBS
- The "best" query plan is
  - Feasible (i.e., can be executed by the local DBS)
  - Provides nearly optimal performance
- The approach:
  - Generate a large set of possible query plans
  - Efficiently select best plan, quickly pruning bad plans

©2002 Vera Goebel & Denise Ecklund

HDBMS-42

## Restrictions on Query Processing Capabilities

- Condition-Attribute restrictions
  - Can not select on attribute A1
  - Must specify a selection value for attribute A2
- Condition-Expression-Size restriction
  - Can not specify more than  $k$  selection conditions
- Condition-Expression-Structure restrictions
  - Allow only atomic conditionals (i.e., no  $\wedge$  or  $\vee$ )
  - Allow only "and-ed" conditions (i.e., no "or" operators)
  - Combination of restrictions on using  $\wedge$  and  $\vee$

©2002 Vera Goebel & Denise Ecklund

HDBMS-43

## Simple Source Description Language - SSDL

- Example database of cars for sale
  - Attributes: {make, model, year, color, price}
- SSDL definition of query processing capabilities for one local DBS

$\_s \rightarrow \_s1 \mid \_s2$

$\_s1 \rightarrow \text{make} = \$m \wedge \text{price} < \$p$  } *Defines 2 forms of*

$\_s2 \rightarrow \text{make} = \$m \wedge \text{color} = \$c$  } *supported queries*

attributes ::  $\_s1$  : {make, model, year, color} } *Defines valid output*

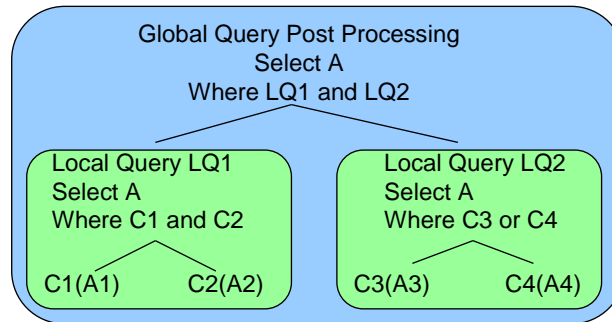
attributes ::  $\_s2$  : {make, model, year} } *attributes for each supported query form*

" "	indicates a variable/non-terminal	\$m	indicates a string constant
\$p	indicates an integer constant	\$c	indicates a string constant from an enumerated set

©2002 Vera Goebel & Denise Ecklund

HDBMS-44

## The Global Query as a Condition Tree



- Obvious Query Plan:
  - DB1 evaluates LQ1
  - DB2 evaluates LQ2
  - PP joins the two results

*What if ?*

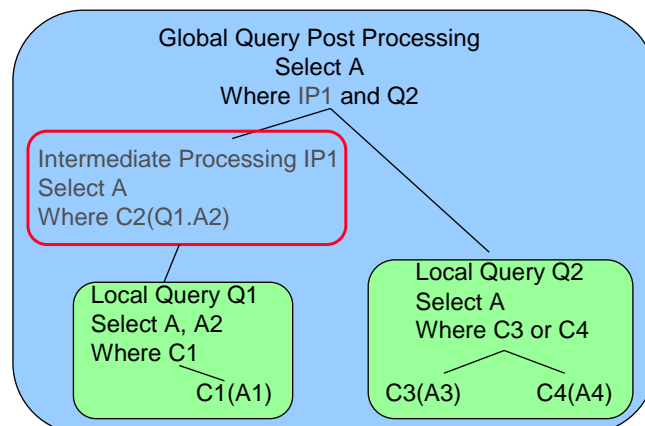
*DB1 cannot evaluate condition C2?*

*DB2 cannot evaluate condition (C3 or C4)?*

## Generating Alternate Plans

DB1 cannot evaluate condition C2

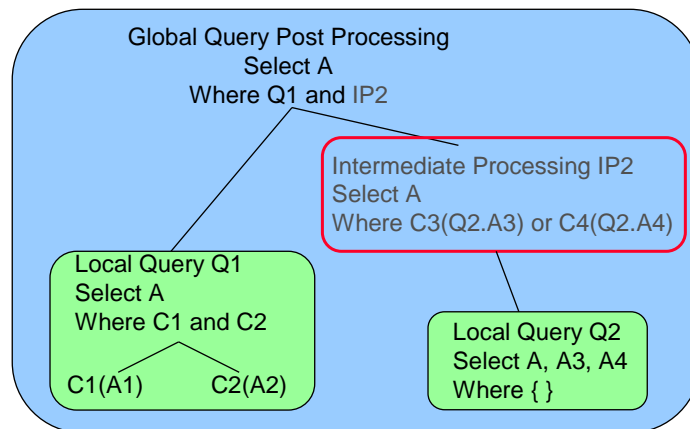
An alternate query plan is:



## Generating Alternate Plans

DB2 cannot evaluate the condition (C3 or C4)

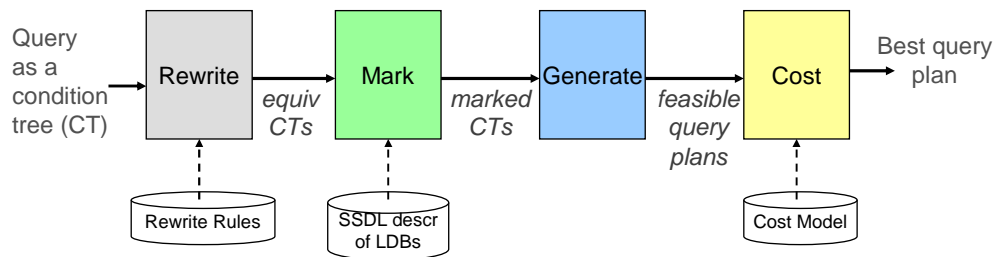
An alternate query plan is:



©2002 Vera Goebel & Denise Ecklund

HDBMS-47

## GenCompact Architecture



- Rewrite - Commutativity, associativity, distribution, and copy rules
- Mark - Tests executability on the applicable LDBs
- Generate - Creates alternate query plans by inserting intermediate processing steps
- Cost - Applies a cost function to quickly select the "best plan"

©2002 Vera Goebel & Denise Ecklund

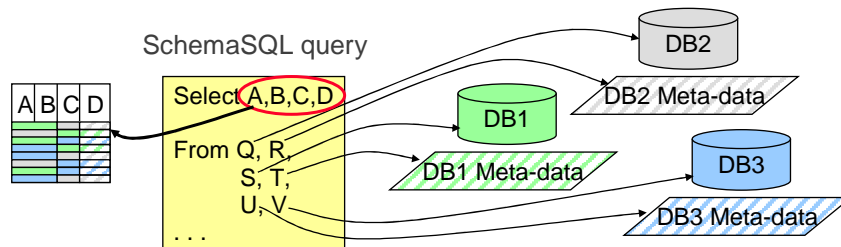
HDBMS-48

**What if . . .**  
all your local systems are relational,  
have similar processing capabilities,  
but use different schemas?

Maybe you can use SchemaSQL to write  
and execute your global queries!

## Schema SQL – A Natural Extension to SQL

- Manipulate data and schema using one language



- SchemaSQL supports
  - One query accessing multiple databases
  - Queries over the schema definitions themselves
  - Runtime restructuring of views and output schemas
  - Aggregation operations over rows, columns and selected blocks

**Next  
lecture**

## SchemaSQL Syntax Definitions

- Extends the variables and ranges defined by SQL
- Valid SchemaSQL ranges are:

Symbol	Meaning
→	Set of database names in the federation
db →	Set of relation names in the database db
db::rel →	Set of attribute names in relation rel in the database db
db::rel	Set of tuples in the relation rel in the database db
db::rel.attr	Set of values in the columns named attr in the relation rel in the database db

- A valid SchemaSQL variable is of the form

<range> <var>

©2002 Vera Goebel & Denise Ecklund

HDBMS-51

## SchemaSQL – Examples: 3 CIS Databases

### Electricity Database

CustInfo	Name	Address1	Address2	CustType	RatePerKwH
----------	------	----------	----------	----------	------------

### Natural Gas Database

Home	Name	Address	RateCategory	
Biz	Name	ServiceAddr	BillingAddr	RateCategory
Industry	Name	ServiceAddr	BillingAddr	RateCategory
Rates	RateCategory	RatePerCuM	Fees	

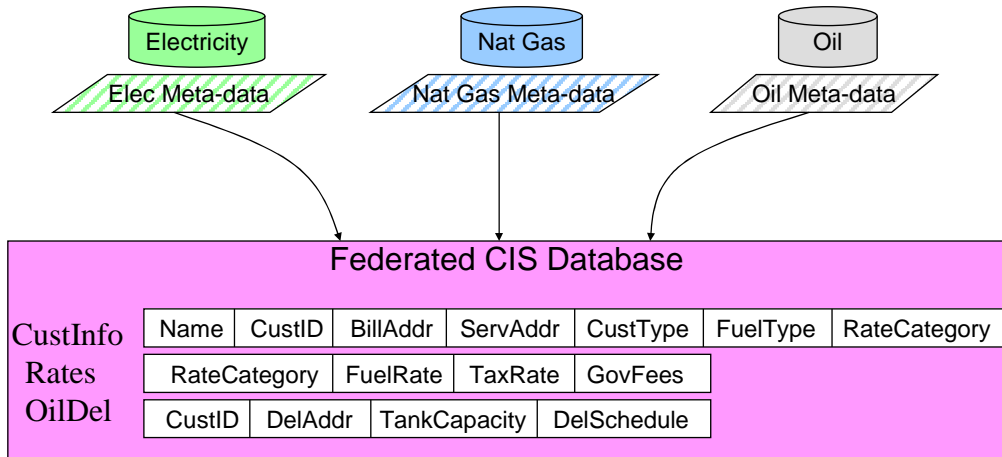
### Oil Database

DeIN	Name	DeliveryAddr	MailingAddr	TankCapacity	DeliveryFreq	PriceRate
DeIS	Name	DeliveryAddr	MailingAddr	TankCapacity	DeliveryFreq	PriceRate
DeIE	Name	DeliveryAddr	MailingAddr	TankCapacity	DeliveryFreq	PriceRate
DeIW	Name	DeliveryAddr	MailingAddr	TankCapacity	DeliveryFreq	PriceRate

©2002 Vera Goebel & Denise Ecklund

HDBMS-52

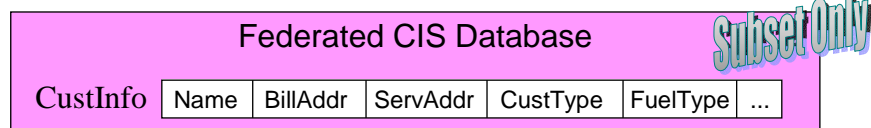
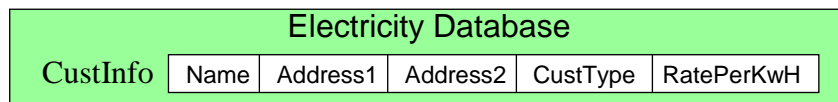
## Global Schema for the Federated CIS Database



©2002 Vera Goebel & Denise Ecklund

HDBMS-53

## SchemaSQL – Example #1



```

create view ElectoFed::CustInfo(Name, BillAddr, ServAddr, CustType, FuelType)
select  Rel.Name, Rel.Address2, Rel.Address1, Rel.CustType, DBname
from    → DBname,
        DBname::CustInfo Rel
    
```

©2002 Vera Goebel & Denise Ecklund

HDBMS-54

## SchemaSQL – Example #2

Natural Gas Database				
Home	Name	Address	RateCategory	
Biz	Name	ServiceAddr	BillingAddr	RateCategory
Industry	Name	ServiceAddr	BillingAddr	RateCategory
Rates	RateCategory		RatePerCuM	Fees

Federated CIS Database					
CustInfo	Name	CustType	FuelType	RateCategory	...

Subset Only

```
create view NGtoFed::CustInfo(Name, CustType, FuelType, RateCategory)
select  Rel.Name, Rel, DBname, Rel.RateCategory
from    → DBname,
        DBname→ Rel,
where   Rel <> "Rates"
```

©2002 Vera Goebel & Denise Ecklund

HDBMS-55

## SchemaSQL – Example #3

**Problem:** Create a view containing the "Low Rate Payers" from all the federated CIS databases.

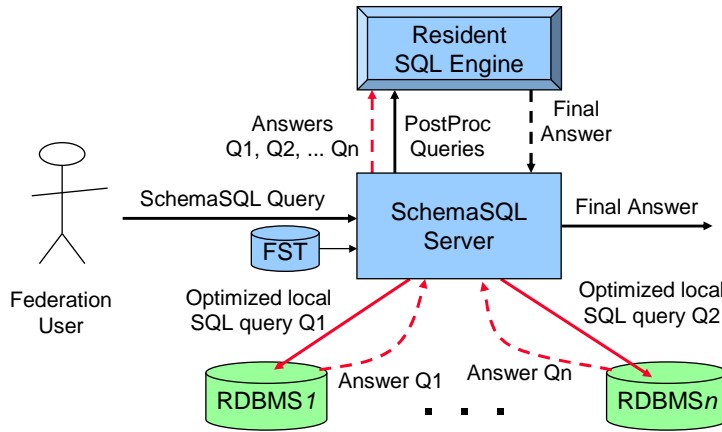
*Low Rate means you pay less than 0.035 for each unit of fuel*

```
create view LowRatePayers::CInfo(Name, FuelType, Rate)
select  CustNameRels.Name, DBnames, RateValue
from    → DBnames,
        DBnames→ CustNameRels,
        DBnames→ FuelRateRels,
        DBnames::FuelRateRels→ RateAttrs,
        DBnames::FuelRateRels.RateAttrs RateValue
where   CustNameRels <> "Rates" and
        (((RateAttrs = "RatePerKwH")
         or (RateAttrs = "PriceRate")
         or (RateAttrs = "RatePerCuM")
         and RateValue < 0.035)
```

©2002 Vera Goebel & Denise Ecklund

HDBMS-56

## SchemaSQL Service Architecture



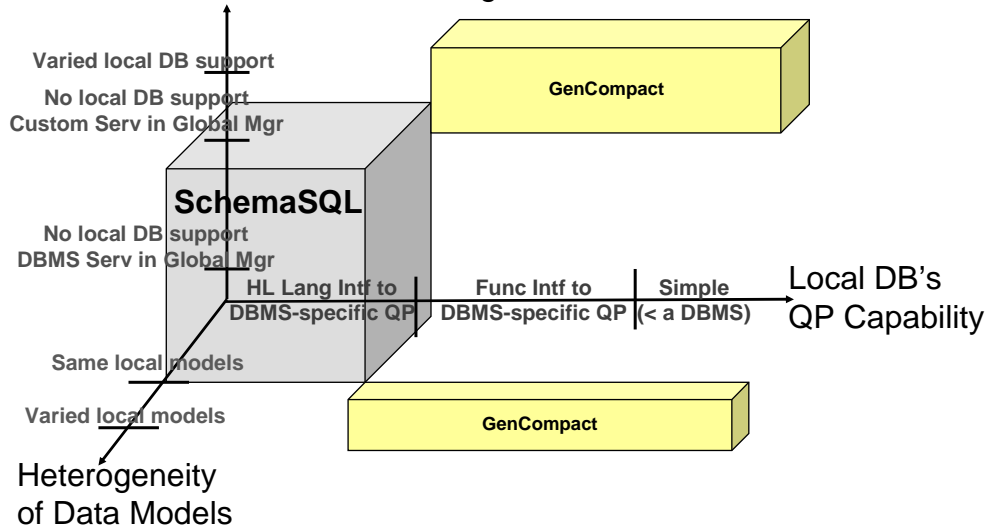
Federation System Table (FST) – stores database names, relation names, attribute names, and statistical information on the local RDBMSs

©2002 Vera Goebel & Denise Ecklund

HDBMS-57

## Summary – Some Query Processing Tools for in HDBMSs

### Intermediate and Post Processing



©2002 Vera Goebel & Denise Ecklund

HDBMS-58

**To Be Continued ... Next Week ...**

Transaction Management in HDBMSs