HETEROGENEOUS DISTRIBUTED DATABASES:
ISSUES IN INTEGRATION

Amit P. Sheth, Ph.D.

Corporate Systems Development Division
Honeywell Inc.
1000 Boone Ave. N.
Golden Valley, MN 55427
(612) 541-6899

© Copyright by Amit P. Sheth, 1987
OUTLINE

1. INTRODUCTION *
2. ALTERNATIVES AND ARCHITECTURES **
3. SCHEMA TRANSLATION
4. SCHEMA INTEGRATION
5. OPERATION TRANSFORMATION
6. QUERY PROCESSING AND OPTIMIZATION
7. TRANSACTION MANAGEMENT AND CONCURRENCY CONTROL
8. MISCELLANEOUS ISSUES
9. CASE STUDIES *
10. SUMMARY
1. INTRODUCTION

1.1 PROBLEM DEFINITION
   - Why do we have heterogeneous Systems and Databases?
   - What types of Heterogenieties Exist?
   - Why do we need to Integrate?

1.2 REQUIREMENTS OF INTEGRATION

1.3 DEFINITIONS AND CONCEPTS

1.4 EXAMPLES
1. INTRODUCTION
1.1 Problem Definition
Why do we have Heterogeneous Systems and Databases

1. Multiple Preexisting Databases:
   - Benefits of Homogeneity and/or Integration not well understood
   - Decentralized Control
   - Historical Reasons

2. Induced Distribution:
   - Functional and Application Distribution
   - Benefits of Distribution
   - Decentralized Control, Security, etc.
1. INTRODUCTION
1.1 Problem Definition
Types of Heterogeneity

<table>
<thead>
<tr>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBMSs</td>
</tr>
<tr>
<td>Data Access Methods</td>
</tr>
<tr>
<td>Data Models</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files Systems</td>
</tr>
<tr>
<td>Naming, File Types, Operations</td>
</tr>
<tr>
<td>Transaction Support</td>
</tr>
<tr>
<td>Interprocess Communication</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Communication</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Hardware/ System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction Set</td>
</tr>
<tr>
<td>Data Formats &amp; Representation</td>
</tr>
<tr>
<td>Configuration</td>
</tr>
</tbody>
</table>
1. INTRODUCTION
1.1 Problem Definition
Why do we need to Integrate
(Benefits of Integration)

1. Data Sharing
   - among applications
   - among users
   - among functional components
     of organization

2. Easier Application Development

3. Evolutionary System/Database Expansion
1. INTRODUCTION

1.2 Requirements of (ideal) Integration

** Uniform User Interface

** Transparent Access to Multiple Databases

* Location Transparency

* Distribution Transparency

* Replication Transparency

* User Transparency (Concurrency Control)

* System Transparency
1. INTRODUCTION

1.3 Definition and Concepts

Glossary

Schema: A description of data in a database in terms of its structure (representation) and constraints on it. Constraints can be explicitly stated or can be inherent in the data model. Schema is also called an intentional component of a database.

Data Model: A method of describing and manipulating the values and relationships of database objects. It may permit specification of underlying data representations, restrictions on use of the data, and so on.
Data Model = Data Definition Model + Data Manipulation Model + Semantic Integrity Constraints

DBMS: Database Management System, software that manages access to a database.

DDBMS: Distributed DBMS, software that manages several databases distributed over multiple sites.

HDDBMS: Heterogeneous DDBMS, software that manages several DBMSs which are distributed over multiple sites.

Federated Databases: A collection of cooperating component (autonomous) databases and unified schemas.

CDM: Common or cannonical data model, data model into which schemas of different databases are translated for the purpose of integration or common access.
1. INTRODUCTION
1.3 Definition and Concepts

Glossary

Representation Data Model: A data model normally used for modelling the databases (e.g., Relational, Hierarchical, and Network).

Conceptual Data Model: allows higher level of description of data (e.g., Entity-Relationship Model) as compared to representation data models. It is often used as a CDM.

Mapping: A correspondence between the data objects of two different schemas. A mapping is nonconstructive if only a source schema is mapped into a target schema. A mapping is constructive if a database (extension) according to a source schema is mapped into a database according to the target schema.

Unified Schema: Provides a view on portions of the underlying databases (component databases). Also called Import Schema.

Export Schema: Describes a portion of component databases which a user of Federated Database is allowed to access. Several export schemas can be integrated to form a unified schema.
1. INTRODUCTION

1.3 Definition and Concepts
Relational Data Model

- It consists of relations (also called tables) consisting of rows and columns

- A column corresponds to an attribute. Repeating values within a column is not allowed

- A row corresponds to a tuple. No ordering is implied among rows.

- An attribute, or a collection of attributes form a key. Values for key attribute(s) must be distinct.

- Relations are manipulated using relational algebra (e.g., project, join, select) or relational calculus expressions.

- Most relational data manipulation languages are nonprocedural. User specifies what results are desired without saying how to access the database.

[Codd 70]
1. INTRODUCTION

1.3 Definition and Concepts
Network Data Model

- Consists of records and sets.
- Records consist of data items or fields. Homogeneous records are grouped as record types.
- Associations among records are called sets. Homogeneous sets are called set types.
- Each set has an owner and a few or more members. Each set type has an owner record type and a member record type.
- Data manipulation language permits the user to navigate through the database by accessing a single record for each command. It is called a navigational or procedural language.
1. INTRODUCTION

1.3 Definition and Concepts
ECR- Model

Entity: an object in a real world

Entity type: collection of similar entities

Category: entity type related to other
  entity types in a certain role
  (subset, generalization)

Relationship: relationship (association) of one
  entity type with another

ECR or other conceptual models are useful
for specifying conceptual schema of a database.
They allow specification in high level concepts,
independent from storage and efficiency consi-
derations. However, it is desirable to map a
schema in a conceptual model into lower level
models.

[Chen 76][Elmasri et al 85]
1. INTRODUCTION

1.4 An Example

Database Integration in a Factory

Islands of Automation
1.4 An Example (continued) Data Sharing in Factory Information System
1. INTRODUCTION
1.4 An Example (continued)
Data Sharing in a part of Information System
1. INTRODUCTION

1.4 Another Example

Consider schemas of three databases and their integration.

**DB1: INQUIRE DATABASE**

```
WELLID  DEPTH  PERM  POROS  SATUR
```

**DB2: IMS DATABASE**

```
WELLID  WELLNAME  WELLDEPTH  WELLOWNER
```

```
ROCKCOMPR
```

**DB3: RIM DATABASE**

```
WID   DEPTH  PERM  POROS  GRDEN
```

**INTEGRATED DATABASE**

**DATA ITEMS**

WELLID, DEPTH, PERM, POROS, SATUR, RCOMP, WOWNER, GRDEN, WELLNAME

**RELATIONS**

R1(WELLID, DEPTH, PERM, POROS, SATUR)
R2(WELLID, WELLNAME, WELLDEPTH, WELLOWNER)
R3(WID, DEPTH, PERM, POROS, GRDEN)

**INTEGRATED-DB = OJOIN * (**

```
R1 OJOIN * (R2, R3)
where R2.WELLID = R3.WELLID
and R2.DEPTH = R3.DEPTH
```

```
where LR1.WELLID = LR2.WELLID and
and LR1.DEPTH = LR2.DEPTH
```

[Breitbart et al 86]
2. ALTERNATIVES AND ARCHITECTURE

2.1 Concepts

2.2 Remote Database Interface

2.3 Multiple Homogeneous Databases

2.4 Multiple Heterogeneous Databases
2. ALTERNATIVES AND ARCHITECTURE

2.1 Concepts

Solution Space

Data Distribution

Multisite

Two Sites

Centralized

System Autonomy

Types of Heterogeneity

Hardware

Operating System

DBMS

Systems

DBMSs

Solutions become more difficult as we go away from origin.
2. ALTERNATIVES AND ARCHITECTURE

2.1 Concepts

DBMS Mappings and Architectures for Three Environments:

Case 1: Remote Database Interface
- One database accessed by multiple applications on different systems (section 2.2)

Case 2: Multiple Homogeneous Databases
- Multiple Databases/identical DBMS distributed over multiple sites (section 2.3)

Case 3: Multiple Heterogeneous Databases
- Different Databases/DBMSs distributed over multiple sites (section 2.4)
2. ALTERNATIVES AND ARCHITECTURE
  2.1 Concepts
  DBMS MAPPINGS

* DBMS may be based on different data models
* Mapping may be constructive or nonconstructive
* Mapping may or may not include operations

<table>
<thead>
<tr>
<th>Same data models</th>
<th>constructive</th>
<th>Operations included</th>
<th>Mapping Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Schema Restructuring (case 1)</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Schema Translation (case 3)</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Database Reorganization (case 3)</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Database Translation (case 3)</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>View (all cases)</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Operation Transform (case 3)</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Homogeneous Distr. System: (Homogeneous DDBMS) (case 2)</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Database Cooperation (Heterogeneous DDBMS) (case 3)</td>
</tr>
</tbody>
</table>

Adapted from: [Tsichritzis and Lachovsky 82]
2. ALTERNATIVES AND ARCHITECTURE
   2.2 Case 1:
   Remote Database Interface

Background: Started with need for
Micro-Mainframe Link:

* Terminal Emulation:
  - Provides basic communication
  - PC acts as a dumb terminal
  - Screen can be dumped into PC file
  - File Transfer can be performed
  (Example: IRMA)

* Virtual Systems:
  - Allows virtual drives on Mainframe disc
  - Data Sharing among PCs which are part of
    Virtual System
  - It runs on PC and Mainframe
    (under control of applications such as
    CICS, TSO, CMA)
  (Example: Free-Link, Tempus Link)

* Remote Database Interface
  .. next page..
2. ALTERNATIVES AND ARCHITECTURE

2.2 Case 1:
Remote Database Interface

Suitable for Environments containing PCs with main Database on Mainframe

Loading application 'database' with data from main Database
- File Transfer from Host to PC
- User Interface to control data transfer
- File Reformatting to meet application needs

Example: INGRES/PC-Link [RTI 86]
2. ALTERNATIVES AND ARCHITECTURE

2.3 Case 2:
Homogeneous Distributed Databases

* Same DBMS at every node but
  Heterogeneous Systems

* It provides Online Access to
  Distributed Databases.
  Main Issues:
  - Distributed Query Processing
  - Transaction Management
  - Concurrency Control
  - Fault Tolerance

* However, if there are preexisting
  Databases and Applications,
  Database Translation and
  Modification of Existing Applications
  may be needed.
2. ALTERNATIVES AND ARCHITECTURE

2.3 Case 2: Homogeneous Distributed Databases

Identical DBMSs may run on

- Identical Computers
  (e.g., PCs networked together)

- System Consisting of PCs, Minis and Mainframes
  (E.g., INGRES/STAR w/o Gateways)

- Multivendor Heterogeneous Systems.

Main issues is that of masking System Boundaries and Heterogeneities.

* Examples: R*, Distributed INGRES,
  SDD-1, Distributed ORACLE

[ Rothnie et al 80] [Haas et al 82]
2. ALTERNATIVES AND ARCHITECTURES

2.4 Case 3: Multiple Heterogeneous Databases

ALTERNATIVES:

a Batch Mode: Periodic Data Exchange,
   No adhoc query.

![Diagram: IMS to DB2]

b Heterogeneous Interface: Online access to single database.
Access in one data model while the DBMS is using other data model.
Extension of Remote Database Interface.

![Diagram: Relational Interface to MDBS-III]

Example: Relation Query Interface in MDBS-III [MDBS-III]
2. ALTERNATIVES AND ARCHITECTURES

2.4 Case 3:
Multiple Heterogeneous Databases

No integration

---

c. Interoperability:
Access to multiple databases with similar or different DBMSs. No location transparency. Nonintegrated schemas.

---

[Litwin and Abdellatif 86]

-25-
2. ALTERNATIVES AND ARCHITECTURES

2.4 Case 3:
Multiple Heterogeneous Databases

d. Gateway: Partial Integration.
Primarily Homogeneous DDBMS.
Use of Gateways to heterogeneous databases.
Usually no Distributed Query with
heterogeneous databases.

e. Heterogeneous DDBMS/Federated Databases:
Complete solution. User deals with single
global database made up of multiple databases,
and DBMSs. In HDDBMS, there is single
global schema which all users may access
(views may be allowed). In Federated Database,
there are multiple global schemas.
Common Data Model, Integrated Schema,
Distributed Query.
2. ALTERNATIVES AND ARCHITECTURES
   2.4 Case 3.e:
       Heterogeneous DDBMS

ARCHITECTURES FOR HETEROGENEOUS DDBMS:

System Architecture

Schema Architecture
   - Three Schema Architecture
   - Five Schema Architecture
   - Federated Databases

(We will see examples in Section 9).

-27-
2. ALTERNATIVES AND ARCHITECTURE
2.4 Case 3.e:
Heterogeneous DDBMS

System Architecture of a HDBMS

Application Processor

Data Processor

Database

Application Processor

Communication

Data Processor

Application Processor

Database

Database

Application Processor performs Transaction Management.
Data Processor performs Data Management.

[Devor et al. 82b]
2. ALTERNATIVES AND ARCHITECTURE
2.4 Case 3.e:
Heterogeneous DDBMS

System Architecture of a HDDBMS (continued)

External Interface (EI) translates a set of commands (transaction) on the conceptual schema into a transaction on the canonical (representation) schema. It may also enforce semantic integrity constraints.

Global Data Manager (GDM) translates a transaction into subtransactions for different sites at which required data reside. This function is also called global access planning. It may perform global optimization. GDM is also called Materialization and Access Planner.

Distributed Execution Monitor (DEM) coordinates executions of subtransactions at different sites.

Local Execution Monitors (LEM) manages subtransaction evaluation at the local site and interacts with DEM.

Local Data Interface (LDI) translates the subtransaction in canonical model into a set of commands in data model of local DBMS.
2. ALTERNATIVES AND ARCHITECTURE

2.4 Case 3.e:
Heterogeneous DDBMS

(Extended) Three Schema Architecture

The ANSI/SPARC information architecture for databases uses three schemas: internal, which describes the data storage structure and access paths; conceptual, which includes restrictions on values or relationships; and external, which describes the different user views. Mapping operations relate schemas at different levels.

[ANSI/SPARC 77]
2. ALTERNATIVES AND ARCHITECTURE
Case 3.e
Heterogeneous DDBMS
Five Schema Architecture

Adding local and global representation schemas allow better representation of the data distribution and DBMS heterogeneity, as well as provide data independence and multiple views. Many systems use same model at representation and conceptual level, thus degenerating to a four level schema architecture.

[Devor et al 82]
Federated Databases provides a model of most flexible logical HDDBMS organization. Two important points are autonomy of component databases and lack of single organization schema.

[Elmasri et al 86] [Heimbigner and McLeod 85]
2. ALTERNATIVES AND ARCHITECTURE
2.4 Heterogeneous DDBMS/Fed. DB

Remarks

A. Federated vs Non Federated

1. Global Enterprise View
   Fed  Non-Fed
   No   Yes

2. Autonomy of Components
   Yes  No

B. Three Level vs. Five Level Schema Architecture

Efficiency: Number of translations and Level of Indirection

Efficiency: Number of translations and Level of Indirection

Expressiveness and Ease:
Ease in Logical Integration and External Interfaces

Three  Five
More  Less
Less  More

C. Selecting Common Data Model
Conceptual Model: Semantically rich, Expressive
used as: CDM Representation

<table>
<thead>
<tr>
<th></th>
<th>E-R/E-C-R/IDEF1</th>
<th>Daplex</th>
<th>Relational</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
3. SCHEMA TRANSLATION

3.1 Concepts

3.2 Network to Relational Translation

3.3 Remarks
3. SCHEMA TRANSLATION

3.1 Concepts

Problem definition: Given a source schema in one model, map it into an equivalent target schema in another model. If the source schema and target schema are according to the same data model, we have a schema restructuring mapping, otherwise we have a schema translation mapping.

Two schemas are equivalent if
(1) they describe the same database,
(2) any command expressed in terms of
   the object of one schema can be translated
   into commands expressed in terms of the
   objects of the other schema, and
(3) the same changes are made to the database
   if either source command or the target
   commands are executed.

Recall: Schema is description of the database. It has two components: Structure and Constraints. Constraints can be explicitly stated or inherent in the data model.

[Larson 83] [Tsichritzis and Lochovsky 82]
3. SCHEMA TRANSLATION
3.1 Concepts

Two ways to perform translation:
Specify mapping between every
pair of data models OR use a common
(canonical/mapping) data model

Examples of common data models:
Relational [Klug 80] [Breitbart et al 84]
Binary [Pelagatti 78] DIAM [Senko 76]
Extended Set Theory [Sibley and Hardgrave 77]
E-R type [Katz 80] [Elmasri et al 85]

S-DMi = Schema in Data Model "i"
C-DM  = Schema in Common Data Model
3. SCHEMA TRANSLATION
3.2 Network to Relational Translation

Structural Similarities:

Network Model :: Relational Model
Record type : Relation
Data item of the record: Column of the relation
Record key : Relation Key
Set : Foreign Key or Relation
3. SCHEMA TRANSLATION
3.2 Network to Relational Translation

Some aspects of Network Model deserve careful examination:

Set Participation: Foreign Key or New Relation

Set Ordering: New Column

Repeating Groups: Restructuring network schema or repeating values in a column

System-owned set: New column

Areas: New column

[Larson 83]

-38-
3. SCHEMA TRANSLATION

3.2 Network to Relational Translation

(1) For each record type $N_i$ define a relation $R_i$ such that:
(a) $R_i$ contains one attribute for each data item of $N_i$.
(b) If $N_i$ has a key, the key of $R_i$ is equal to the key of $N_j$;
    otherwise, the key of $R_i$ is equal to the database key
    of $N_i$, which appears as an explicit attribute of $R_i$.

(2) For each link $L_{ij}$, with owner record type $N_i$ and member
    record type $N_j$, define relational constraints and change
    the existing relations such that:
    (a) The key of $N_i$ appears as a foreign key of $R_j$.
    (b) One of the following sets of constraints applies,
        depending on the type of set membership ($R_i$ and $R_j$
        are the relations corresponding to $N_i$ and $N_j$,
        respectively).
        - Fixed Automatic         - Fixed Manual
        - Mandatory Automatic    - Mandatory Manual
        - Optional Automatic      - Optional Manual
    (c) Instead of explicit constraints of (b), it is possible
        to derive relational schema that contains only
        relations and multivalued dependencies if network
        schema is loop-free and the links are fixed automatic.

[Tsichritzis and Lochovsky 82]
[Vassiliou and Lochovsky 80] [Lien 80]
3. SCHEMA TRANSLATION
3.2 Network to Relational Translation
An Example

Source schema: A network database and schema.

Target schema: A relational database and schema.

[Larson 83]
3. SCHEMA TRANSLATION
3.2 Network to Relational Translation
An Example: Alternate Translation

Source schema: A network database and schema.

Target schema: A relational database and schema.

[Larson 83]
3. SCHEMA TRANSLATION
3.2 Network to Relational Translation
An Example: Possible Mapping

1. R-COMPANY.Company Name <- N-COMPANY.Company-Name
   R-COMPANY.City <- N-COMPANY.City
   R-PRODUCT.Product-Name <- N-PRODUCT.Product-Name
   R-PRODUCT.Cost <- N-PRODUCT.Cost

   Key(R-COMPANY) = Company-Name
   <- Key(N-COMPANY) = Company-Name

   Key(R-PRODUCT) = Product-Name
   <- Key(N-PRODUCT) = Product-Name

2. Foreign-Key(PRODUCT.Company-Name)
   <- Key(N-COMPANY.Company-Name)

   Explicit Constraints = Optional Automatic
   R-PRODUCT.Product-Name sequenced
   by R-PRODUCT.Company-Name
3. SCHEMA TRANSLATION
   3.3 Remarks

* Easier Translations:

  Procedural to Non-procedural
  Conceptual to Representation

  E.g. Network to Relational
       ECR to Relational

* More Difficult Translations:

  Non-procedural to Procedural
  Representation to Conceptual

  E.g. Relational to Network
       Relational to ECR

[Tsichritzis and Lochovsky 82]
### 3. SCHEMA TRANSLATION

#### 3.3 Remarks

Comparison Summary of Mapping:
Representation Structures

<table>
<thead>
<tr>
<th>ECR</th>
<th>Relational</th>
<th>Network (DBTG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity-type</td>
<td>Relation</td>
<td>Record type</td>
</tr>
<tr>
<td>Category (ISA)</td>
<td>Relation</td>
<td>Record type and single owner single member set</td>
</tr>
<tr>
<td>Category (generalization)</td>
<td>Relation</td>
<td>Record type and single owner multiple member set</td>
</tr>
<tr>
<td>Functional relationship (1:1,1:N)</td>
<td>Foreign key</td>
<td>Single owner single member set</td>
</tr>
<tr>
<td>Non-functional relationship (M:N)</td>
<td>Relation</td>
<td>Record type and owner-member sets</td>
</tr>
<tr>
<td>Single valued attribute</td>
<td>Attribute</td>
<td>Field</td>
</tr>
<tr>
<td>Multi-valued attribute</td>
<td>Attribute 'Nest' relation</td>
<td>Repeating group, vector</td>
</tr>
<tr>
<td>Entity identifier</td>
<td>System assigned attribute</td>
<td>Database key</td>
</tr>
</tbody>
</table>

Elamsri et al 85]
4. SCHEMA INTEGRATION

4.1 Application of Schema Integration
   - Integrating Preexisting Databases
   - Logical Database Design

4.2 Basic Concepts

4.3 Resolving Incompatibilities

4.4 Process of Integration
4. SCHEMA INTEGRATION
4.1 Applications of Schema Integration

INTEGRATING PREEXISTING DATABASES
4. Schema Integration

4.1 Applications of Schema Integration

Use of Integrated Schemas

User → Unified Schema 1 → Request Mapping → Export Schema I → Database S

→ ...

→ Export Schema k → Database r

→ ...

→ Export Schema j → Database q

→ Subrequests
4. SCHEMA INTEGRATION
4.1 Applications of Schema Integration

LOGICAL DATABASE DESIGN

User View 1 (Schema) → Integration → Global Conceptual Schema
User View 2 (Schema)

User View n (Schema)
4. SCHEMA INTEGRATION

4.1 Applications of Schema Integration

USE OF INTEGRATED VIEWS

User → User View 1 (Schema) → Request Mapping

User → User View 2 (Schema) → Request Mapping

... → Global Conceptual Schema → Data Base

User → User View n (Schema) → Request Mapping
4. SCHEMA INTEGRATION
   4.2 Concepts

Represent all component/export schemas in
same data model.

Use of conceptual model is preferred since
(a) it is easy to understand, and
(b) it has semantics needed for integration.

Process:
(1) Recognize similarity between every
    pair of objects.
(2) Specify Assertions.
(3) Integrate.
4. SCHEMA INTEGRATION
4.2 Concepts

Integration of Similar Objects

Case 1: Identical Domains

```
Name   SC1.Department  +  Name   Department
```

Case 2: Contained Domains

```
ID      Student
GPA     +
ID      Grad_Student
GPA     Support type
```

[Navathe et al 86]
4. SCHEMA INTEGRATION
4.2 Concepts
Integration of Similar Objects

Case 3: Overlapping Domains

Case 4: Disjoint Domains

Case 5: Non-integrable Domains

-52-
4. SCHEMA INTEGRATION
4.3 Resolving Incompatibilities

Types of Incompatibilities:
1. Differences in Naming Conventions
2. Differences in Underlying Data Structures
3. Differences in Representations
4. Differences in Scales/Units
5. Missing Data
6. Conflicting Data Values

Solutions
A. View Mechanism allows
   Renaming
   Logical Restructuring
   Scale Conversion
   These enable us to solve 1 thru 4.
B. Auxiliary Database can be stored
   with auxiliary schema attached to the
   unified schema to solve 5 and 6.

[Landers and Rosenberg 82] [Litwin Abdellatif 86]
4. SCHEMA INTEGRATION
4.4 Process of Integration

Non-cannonical Component Schemas → Schema Translation → Cannonical Component Schemas

Pre-integration
- Schema Analysis, Incompatibilities Resolution and Modification
- Assertion Specification, Analysis, and Modification

Integrating Data Dictionary

Mappings

Mapping Generation

Object Integration
- Lattice Merging
- Schema Integration

Integratated Schema

[Navathe et al 86]
5. OPERATION TRANSFORMATION

5.1 Concepts

5.2 View Mapping

5.3 Query Translation
5. OPERATION TRANSFORMATION

5.1 Concepts

Problem definition: Given a source schema, a database according to the source schema and an equivalent target schema, map operations on target schema to equivalent operations on source schema. If the source and target schemas are according to the same data model then we have view mapping, otherwise operation transformation mapping. Operation transformation mapping is used to translate a query operation on a target schema into equivalent operations on the source schema.
5. OPERATION TRANSFORMATION

5.1 Concepts

Global Data Manager: Global Access Planning and Optimization

Local Data Interface: Operation Transformation

Local DBMS

Local Database

Operations on Canonical Schema → Operation Transformation → Operations on Internal Schema

GDM: Global Data Manager  LDI: Local Data Interface
5. OPERATION TRANSFORMATION
5.2 View Mapping
(through Query Modification)

DATABASE RELATIONS:

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th>DEPARTMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>ADDRESS</td>
</tr>
<tr>
<td>Smith</td>
<td>15 Bloor</td>
</tr>
<tr>
<td>Jones</td>
<td>25 King</td>
</tr>
<tr>
<td>Carter</td>
<td>171 Yonge</td>
</tr>
</tbody>
</table>

VIEW DEFINITION:

EMPVIEW <-
SELECT EMPLOYEE.NAME,ADDRESS,MANAGER
FROM EMPLOYEE,DEPARTMENT
WHERE EMPLOYEE.DEPT = DEPARTMENT.DEPT

VIEW:

EMPVIEW

<table>
<thead>
<tr>
<th>NAME</th>
<th>ADDRESS</th>
<th>MANAGER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>15 Bloor</td>
<td>Miles</td>
</tr>
<tr>
<td>Jones</td>
<td>25 King</td>
<td>Quin</td>
</tr>
<tr>
<td>Carter</td>
<td>171 Yonge</td>
<td>Miles</td>
</tr>
</tbody>
</table>

Now a query on EMPVIEW:

ANSWER <-
SELECT MANAGER
FROM EMPLOYEE
WHERE NAME = 'Smith'
can be rewritten into the following query by a technique called Query Modification:

ANSWER <-
SELECT MANAGER
FROM DEPARTMENT
WHERE DEPARTMENT.DEPT = SELECT EMPLOYEE.DEPT
FROM EMPLOYEE
WHERE NAME = 'Smith'

[Stonebraker 75] [Tsichritzis and Lochovsky 82]
5. OPERATION TRANSFORMATION

5.3 Query Translation
Relational to Network

Transforming nonprocedural relational data manipulation command into a sequence of procedural network commands:

Select =>
A looping statement in network data manipulation language in which each record is retrieved and examined. Records satisfying the selection condition are moved to the target relation.

\[
T = \text{Select} \ (R1, \text{condition}) \Rightarrow r = \text{get\_first} \ (N1);
\]
\[
\quad \text{Loop}: \text{if} \ \text{condition}(r); \text{insert} \ r \ \text{into} \ T
\]
\[
\quad r = \text{get\_next}(N1); \ \text{go to Loop}
\]

Project =>
A looping statement in which each record is retrieved and the specified items are copied to the target table.

Join =>
Two nested loops in which the outer loop reads successively records from one record type and the inner loop successively reads records from second record type in the set owned by each of the record of the first record type.
5. OPERATION TRANSFORMATION

5.3 Query Translation

Remarks

Operation Mapping can be established by direct mappings or by using intermediate mapping data model as in schema translation.

Following complexities exist in query translation:
1. Translating procedural operations into nonprocedural operations is very difficult. (This may be an important reason for not choosing a procedural model as a representation model.)
2. Dynamic creation of tables is required.
3. Translating conjunctive (OR) terms in a relational query is difficult.

[Larson 83] [Vassiliou and Lochovsky 80]
6. QUERY PROCESSING AND OPTIMIZATION

6.1 Query Processing

6.2 Global Optimization

6.3 Local Optimization
6. QUERY PROCESSING AND OPTIMIZATION

6.1 Query Processing

Diagram:

- Global Query flows to Global Data Manager, which then sends a Result.
- Single-site Query flows from Global Execution Monitor to Local Execution Monitor.
- Data flows between Local DBMS and Local Data Manager.
- Local Query flows to Local DBMS, which sends Data to Local Data Manager.
6. QUERY PROCESSING AND OPTIMIZATION

6.1 Query Processing

Goal: Utilizing the potential for parallel processing in a distributed system and minimizing processing cost by reducing the amount of data moved between sites.

Issues: Additional complexities introduced by the heterogeneous environment.

1) Cost of performing a local query may differ greatly from site to site.
2) The ability to receive and reference "moved" data may vary from site to site. Many DBMS do not support creation and loading of temporary databases.
3) Local querying capabilities may be limited.
4) Local DBMSs vary in their capabilities to optimize.

[Landers and Rosenberg 82]
6. QUERY PROCESSING AND OPTIMIZATION

6.2 Global Optimization

There is a trade-off between the amount of work done by the GDM and the software complexity of LDI. There is also a trade-off between amount of communication and processing done by different sites. Depending on how a global transaction is divided, a range of alternatives is possible.

Examples of alternatives are:

(1) More work by GDM, more communication but simple: Translate the global query in smallest possible single site subqueries. Each site may be sent several subqueries. Collect partial result at GDB and merge.

(2) Medium work by GDM, medium communication, medium complexity of GDM and LDI's: Translate the global transaction into largest possible single site subtransactions (one transaction for each site where data is located). This will reduce communication and needs less work when merging partial results. However, the LDI's have to be sophisticated.
6. QUERY PROCESSING AND OPTIMIZATION

6.2 Global Optimization

(3) Less work by GDM, less communication, complex GDM and LDI's:
Generate efficient programs such that LDI's participate in global optimization. Partial results can be sent to GDM or other LDI's. LDI's have to be able to support some additional functionalities like sorting, removing duplicates, handling and merging temporary files.

E.g., consider multisite join:

![Diagram of multisite join]
6. QUERY PROCESSING
AND OPTIMIZATION

6.3 Local Optimization

Goal: Optimize processing of the subtransaction received from GDM against the local DBMS.

Opportunities for Optimization:
- Local Query Language
- Physical Database Organization
- Access Path Selection

[Onuegbe et al 83] [Dayal and Goodman 82]
Sequences of joins, selects, and projects can be optimized.
E.g., a select followed by a project can be combined into a single loop that selects records satisfying the select condition and copies the projected items.
7. TRANSACTION MANAGEMENT AND CONCURRENCY CONTROL

7.1 Concepts

7.2 Effect of Heterogeniety

7.3 Alternative Solutions
7. TRANSACTION MANAGEMENT AND CONCURRENCY CONTROL

7.1 Concepts

A transaction is a set of operations on database. It has following ACID properties:

1. **ATOMICITY**: Either all or none of the operations should "happen". It commits or aborts.

2. **CONSISTENCY**: Each transaction should see a correct picture of the database state, even if concurrent transactions are executing.

3. **INTEGRITY**: The transaction should be a correct state transformation.

4. **DURABILITY**: Once a transaction commits, all its effects must be preserved, even if there is a failure.

[Haeder and Reuter 83]

-69-
7. TRANSACTION MANAGEMENT
AND CONCURRENCY CONTROL

7.1 Concepts

Two concurrently executing transactions may leave the database inconsistent. The issue of Concurrency Control is to coordinate or synchronize concurrent accesses to a database.

Transaction 1:
- read BALANCE
- Add 50 to BALANCE
- Write BALANCE

Transaction 2:
- read BALANCE
- Add 100 to BALANCE
- Write BALANCE

But, BALANCE should be 1150.
7. TRANSACTION MANAGEMENT AND CONCURRENCY CONTROL

7.1 Concepts

There are three methods of Concurrency Control:

- Locking
- Timestamping
- Optimistic

Locking: If a transaction accesses a data item, it sets a lock on it. Another transaction with a conflicting lock request has to wait until the lock is released.

Timestamp: Each transaction get a timestamp at the beginning and the conflicting transactions are processed according to the timestamp.

Optimistic: After a transaction is executed, it checks if it could have conflicted with other transaction. If so it "aborts" otherwise it "commits".

Two transactions conflict if one of them write a data item while other reads or writes the same data item.

The above definitions are incomplete and oversimplified.

[Bernstein and Goodman 81]
7. TRANSACTION MANAGEMENT
AND CONCURRENCY CONTROL

7.1 Concepts

Locking is the most popular method. But, whenever a locking is used, deadlock can occur.

Arrows show wait-for graph. A cycle in a wait-for graph implies a deadlock.
7. TRANSACTION MANAGEMENT
AND CONCURRENCY CONTROL

7.1 Concepts

In order to preserve atomicity property of the transaction, TWO-PHASE COMMIT protocol is used.

[Bernstein and Goodman 81]
7. TRANSACTION MANAGEMENT AND CONCURRENCY CONTROL

7.2 Effect of Heterogeneity

* Different DBMSs use different Concurrency Control methods.

* Even if the Concurrency Control methods are the same, implementations may vary significantly. E.g., locking methods may vary in deadlock handling methods, number of items locked by one lock may vary, etc.

* The DBMSs may vary in commit protocols. E.g., Some may not provide two phase commit, while some may.

[Logar and Sheth 86]
7. TRANSACTION MANAGEMENT AND CONCURRENCY CONTROL

7.3 Effect of Heterogeneity

Global Deadlock Detection Problem

A global deadlock involves transactions at more than one site. Problem of global deadlock detection is very difficult because (1) a local process (part of local DBMS or Operating System) does not know about non-local transactions, and (2) a global process does not know about the local transactions.

Most practical solution seems to be time-out.

Example of global deadlock:
7. TRANSACTION MANAGEMENT
AND CONCURRENCY CONTROL

7.3 Alternatives

1. Allow only read (retrieval) operation. Do not allow write (update) operation. Concurrency control problem does not arise if all transactions are read only.

2. Assume all DBMSs provide locking and support two-phase commit. Detect deadlocks by time out. This assumption is made in INGRES/STAR.

3. Allow single site updates. Local Concurrency Control mechanism is sufficient in this case.

4. Allow multi-site updates. Local DBMSs and Operating Systems will have to be changed if there is heterogeniety. DDTS used this approach.

[Gligor and P-Zeletin 85] [Logar and Sheth 86]
8. MISCELLANEOUS

8.1 Hardware Heterogeniety
8.2 Communication Heterogeniety
8.3 Operating System Heterogeniety
8.4 Data Dictionary
8.5 Integrating CAD/CAM databases with Business Databases
8.6 Standards: MAP/TOP, IGES/PDES, IRDS
8. MISCELLANEOUS

8.1 Types of Heterogeneity in Hardware/System

Difference in Domain Representations:
e.g., Boolean, integer

Difference in Data Formats
e.g., IBM integer, PRIME integer

Difference in Data Representation
e.g., ASCII, EBCDIC

Translation: one-to-one OR cannonical
9. CASE STUDIES

Some of the efforts to build HDDBMS are:

Distributed Database Testbed System (DDTS);
Honeywell: [Devor et al 82, Dwyer et al 86]

Integrated Information Support System (IISS);
Arizona State University: [IISS 86]

Multibase; Computer Corp. of America: [Landers.. 82]

Multics Relational Data Store Multidatabase (MRDSM);
INRIA, France: [Litwin 85]

INGRES/STAR; Relational Tech., Inst. [RTI 86]

Mermaid; Systems Dev. Corp.: [Templeton et al 83]

Automated Manufacturing Research Facility (AMRF);
National Standards Foundation:

Amoco Distributed Database System (ADDS);
Amoco: [Breitbart et al 86]

ARCHEDDA; U.K.: [Stephenson and Main 86]

SCOOP; France: [Spaccapietra et al 82]
9. CASE STUDIES

9.1 DDTS
Important Features

Started in 1979 at Honeywell.

Schema Architecture:

Five Level Schema Architecture
(see section 2.4, page 31)

System Architecture:

Application Processor
Data Processor
(see pages 28 and 29)

Communication: DSA
(MAP in a second similar system)

Common Data Model: Relational
Conceptual Data Model: E-C-R
9. CASE STUDIES

9.1 DDTS
Important Features

Geographically Distributed (Minneapolis, Phoenix)

Heterogeneous Hardware, Operating System, DBMSs.
Multiple user interfaces-
GORDAS Language, SQL, and Graphical.

Current System Configuration:
Four Nodes:
Three Honeywell DPS-6 with IDS-II DBMS
One DPS-8 with RAM DBMS
Also ties to another HDDBMS called
Shop Floor Two Level Query System
(Three Node, DDTS like system with
different Hardware, OS, DBMSs)

Compiled Queries can be stored.
Allowed updates in past. Changed Operating System for Concurrency Control.

Lacks good performance because of prototype nature.
9. CASE STUDIES

9.1 DDTS

System Schematic

[Dwyer et al 86]

-82-
9. CASE STUDIES

9.1 DDTS

System and Schema Architecture
9. CASE STUDIES

9.2 IISS
Important Features

Funded by Government. Developed at Arizona State University.

Schema Architecture:
Extended Three Level.

System Architecture:
User Interface (Virtual Terminal Interface and Forms Processor)
Common Data Model Processor
Network Transaction Manager
Communication

Nice forms based interface, Neutral
Data Manipulation Language (like SQL)
Common Data Model: IDEF1x (Ex. E-R)

Current Status: Prototype runs on Two Systems (IBM and VAX). Workshop held.
Plan to make a system for regular use.
Current performance (response time) not very good.
9. CASE STUDIES
9.2 IISS
Three Schema Architecture

External Schema

Conceptual Schema

IDEF1x

Internal Schema
9. CASE STUDIES
9.2 IISS

Common Data Model Processor - Precompiler

- CDM
- Users Application (Neutral Language)
- Conceptual to External Transformer
- Request Processor 1
- Request Processor 2
- Modified Users Application
9. CASE STUDIES
9.2 IISS

Common Data Model Processor - Program Execution

Modified Users Application

Distributed Request Supervisor DRS

Request Processor 1

Result R1

DataBase DB1

Request Processor 2

R2

DB2

Conceptual To External Transformer

Aggregator

R4

R3
9. CASE STUDIES
9.2 IISS
Software System Architecture
9. CASE STUDIES

9.3 Multibase
Important Features

Developed in early 80's at Computer Corporation of America.

Three level schema architecture. Four types of schemas:

- Local host schema
- DAPLEX local schema
- DAPLEX global schema
- DAPLEX auxiliary schema

System architecture:

Global Data Manager (GDM)
Local Data Interface (LDI)

Common Data Model: Functional
Global DDDL and DML: DAPLEX

Current Status:
One or more prototypes in regular use.

[Landers and Reosenberg 82]
9. CASE STUDIES
9.3 Multibase

Multibase System Schematic

Multibase System Architecture
9. CASE STUDIES
9.3 Multibase

Global Data Manager Architecture

Local Database Interface Architecture
9. CASE STUDIES

9.4 MRDSM
Important Features

Datamodel homogeneous, but semantically heterogeneous

No global schema but (a) system provides functions for manipulating data belonging to distinct/nonintegrated schemas, and (b) users face a variety of views of reality

Functionality:

A. DBA can define dependency schema which describe interdatabase relationships in terms of privacy, integrity, data meaning, etc.
9. CASE STUDIES

9.4 MRDSM
Important Features

B. Data sublanguage to perform:

- joining of data in different database schemas
- user defined value types for dynamic transformation of attributes
- interdatabase queries
- dynamic aggregation of data, etc.

Interesting features:

- multiple identifiers
- semantic variables
- dynamic attributes
- interdatabase queries

[Litwin and Abdellatif 86]
9. CASE STUDIES

9.5 INGRES/STAR
Important Features

Homogeneous Relational DDBMS will be available in Spring 1987. Gateways are being built to interface with limited databases (IMS and DB2).

Advantages of Homogeneous DDBMS with Limited Access to Heterogeneous Databases

Distributed Query and Update in Distributed Ingres databases. Remote query to gateway connected DBMSs.

CDM: Relational. Gateways will convert SQL to local DML.

Distributed update. Locking supported by all Ingres nodes. Deadlock detection by timeout.

Ingres DBMS running on heterogeneous systems will be fully connected.

[Ref: RTI 86]
9. CASE STUDIES

9.5 INGRES/STAR
9. CASE STUDIES

9.6 Comparison

Common Data Model

Relational:  
DDTS
INGRES/STAR
ADDS
ARCHEDDA
(MERMAID)
(MRDSM)
(AMRF)

E-R:  
IISS (IDEF1x)

Functional:  
Multibase

Binary:  
SCOOP
9. CASE STUDIES

9.6 Comparison

Important points for comparison:

- Data models supported/ DBMSs supported
- Types of systems supported
- Frontend to multiple DBMSs or complete DBMS
- Limitations on manipulation on component databases (single site query, distributed query, single site update, multisite update)
- Conceptual elegance vs practicality and performance
10. SUMMARY

Some of the questions we have attempted to answer, directly or indirectly, are:

What types of heterogeneities exist?
Why do we have to deal with heterogeneities?
How can we deal with heterogeneities?
What are the range of choices?
What is the state of art?
How to evaluate a heterogeneous DDBMS?
Bibliography

Schema Translation: Elmasri et al 85, Kelinichenko 78, Katz 80, Klug 81, Larson 83, Lien 81, Navathe 80b, Pelagatti et al 78, Senko 76, Sibley and Hardgrave 77, Vassiliou and Lachovsky 80, Zaniolo 79a, Zaniolo 79b

Query Translation and Optimization: Dayal and Goodman 82, Katz 80, Larson 83, Onuegbe et al 83, Vassiliou and Lachovsky 80, Zaniolo 79b

Schema Integration: de Souza 86, Effelsberg and Mannino 84, Elmasri et al 86, Larson et al 86

Concurrency Control: Bernstein and Goodman 81, Gilgor and P-Zeletin 85, Logar and Sheth 86

Heterogeneous Distributed Databases and Federated Databases: Breitbar et al 86, Dwyer et al 86, Furlani et al 83, Hammer and McLeod 79, Heimbigner and McLeod 85, IISS 86, Landers and Rosenberg 82, Litwin 85, Spaccapietra et al 82, Stephenson and Main 86, Templeton et al 83


This is perhaps the most referenced paper on the topic of concurrency control (in homogeneous distributed database systems).


Describes Amoco Distributed Database System (ADDS). It integrates multiple heterogeneous DBMSs. Extended relational model is used as the CDM. It can integrate relational, hierarchical and network databases, and files. Status: System is designed and is being implemented.


A very good reference on E-C-R model. Also discusses schema translation among E-C-R, relational and network schemas.


Our discussion on Schema Integration is based on this reference and [Navathe et al 86]. A Schema Integration tool for integrating E-C-R schema has been built at Honeywell and U. of Florida, Gainesville on Apollo.

[Free-Link] Free-Link - Product Literature, On-Line Software Int'l, Inc, Fort Lee Executive Park, 2 Executive Dr., Fort Lee, NJ 07024


We reference this article since it may be easy to access. There are many other publications on IISS in form of project reports to the US Government, but they may be difficult to obtain.


Multibase is one of the best known HDDBMS. This well known and well written reference may be read to study Multibase as well as general concepts.


A good reference on Schema Translation. Some concepts related to Operation Transformation are also discussed.


published, 1986.


We have used this reference for discussing interoperability and MRDSM. Several interesting concepts even though MRDSM only supports relational databases.


Chapter 2 and 6 discuss how LOCUS handles heterogeneity in hardware.


Uses ANSI/SQL capabilities as the internal (common) data model. Addresses the issue of integrating existing systems.


Mermaid is a front end to heterogeneous DDBMS. This is a well known system and has several interesting concepts.


Chapter 14 is a very well presented account of DBMS Mapping (including Schema Translation and Database Translation) issues.


