Heterogeneous or Multi-database Systems (MDBS)

- This is a layer on top of an existing DB systems that integrates different software environments and machine architectures with different DBMSs which are all heterogeneous.
- The DBMSs employ different data models (or no models at all) - i.e. hierarchical, relational, network - and different DDLs, DMLs - and may have different concurrency control mechanisms and transaction processing environments.
- A multi-database system creates the illusion of logical DB integration without requiring physical DB integration.

Advantages:
- **Extended user capabilities**: Users access and share data without the added burden of learning the intricacies of different DBMSs.
- Pre-existing programs and procedures are still operational in the integrated MDBMS.
- The common data model used to employ this layer is the relational model and in most cases the DML is SQL.
- A complex provision of a common Conceptual Schema: is an integration of all conceptual schemata of the heterogenous DBMSs.

See e-servers and other toys, i.e. environments with MSDOS Windows, Unix, and other OSs with different DBMSs.

Transaction Management

Two types of transactions:
- **Local transactions** - executed by each local DBMS and outside of the MDBMS system control.
- **Global transactions** - executed under the MDBMS system control.

Concurrency and Serializability

- The MDBMS has NO control over the execution of Local Transactions.
- Each Local DBMS must ensure some concurrency control, such as two phase locking or timestamps, and to ensure that its schedule is serializable and avoidance of local deadlocks.
- To ensure Global serializability - local serializability is not enough: we may have a non-serializable situation as below!
**Example**

Suppose two global transactions T1 and T2 each access and update two data items I1 and I2 located at sites S1 and S2 respectively. Suppose that the local schedules are serializable.

A possible situation:

- At site S1, T2 follows T1 and at site S2 T1 follows T2
- This results in a non-serializable global schedule!

To avoid this situation:

1. Prohibit global transactions updating local data items but this limits the scope and the role of the MDBMS and its capabilities.
2. The local DBMS provides the MDBMS with its local scheduling information. But this may require design changes in the local DBMS which results in losing its local autonomy.
3. **SOLUTION:** If each local DBMS uses the strict two-phase locking protocol and global transaction locks are released only after the transaction finishes its execution, then correctness is ensured!
4. If the local DBMS uses a different type of *transaction management*, then the MDBMS must use some form of concurrency control scheme to ensure global serializability, like the tree protocol or the wait-for-graphs.
Database Technology Applied Outside Data Processing

Conventional (OLD) Applications:

- **Uniformity** - data items are mostly large numbers with a specific same size
- **Record Oriented** - fixed length records.
- **Small Data Items** - records are short - 80 bytes, and most a few hundred bytes.
- **Atomic Fields** - fields are short and fixed length
- **Short Transactions** - transactions are programs with execution times measured in fractions of a second - No human interaction with transaction during its execution. User prepares transaction, submits for execution, then awaits response.
- **Static Conceptual Schema** - changed infrequently and when it is changed the changes are relatively simple. In Relational, the modifications allowed are create a Relation, add and remove Attributes from a Relation.

But also, we have:

- **CAD.** Computer Aided Design. A CAD db stores data pertaining to an engineering design, including the components of the items being designed, the interrelationships of components and old versions of designs.

- **CASE:** Computer Aided Software Engineering. A CASE db stores data required to assist software developers. These data include specs, designs, source code and maintenance, - dependencies among software modules, definitions, uses of variables, and the development history of a system. A repository is created and an Object Manager (OM) is used to handle the requests in order to create new software applications. It also includes front end English-like systems instead of embedded programming languages.

- **Multimedia** dbs contain: Spatial data, Audio, Video, Geographical data, Voice Mail systems, Graphics applications, etc.

- **OIS:** Office Information Systems. OIS systems include workstation based tools for document creation and retrieval, tools for maintaining appointments and calendars, and so on. An OIS db must allow queries. Allows for queries pertaining to schedules, documents, and contents of documents.

- **DSS:** Decision Support Systems. It is a special CASE tool - a combination of dbs, OIS and repositories that assist the management in the decision making process.

- **Expert Database Systems (EDBS)** Contains data as well as explicit rules representing integrity constraints, triggers and other knowledge about the enterprise modeled by the DB.
  - Consist of if-then rules
  - The then clause is generally an arbitrary transaction in the DB
  - Continually checks to see if data satisfies the *if clause*
  - One or more rules are then chosen and the correct *then clause* is executed.
  - Similar to AI systems in structure but uses a DB as opposed to virtual (or main) memory.
  - So an expert system is really a standard DB.
Data Warehouses

A data warehouse organizes and stores data needed for informational and analytical processing over a long historical time perspective!

- Collection of data in support of the management's decision making process (DDS).

A data warehouse is:

- Subject oriented
- Interpreted
- Time variant
- Non-volatile

Typically the data entering the data warehouse comes from the operational environment. The Data Warehouse is always a physically separate store of data transformed from the application data found in the operational environment.

A Data Warehouse Case Study

A typical old example of D.W.: 63 grocery stores with more than 19,000 products with roughly 12 million transactions and 35 on going sales promotions.

The data warehouse includes:

- five dimensional tables: Customer, Period, Product, Promotion, Store.
- Two fact tables: Daily forecasts and daily sales.

Loading the data into the data warehouse consists of:

- Data cleaning
- Index building (Relational dbs do not have this)
- Referential Integrity checking
- Aggregation Updating

Some indicative Store: Over 1 Terabyte of data

- 10 days to load
- The largest table is 7.7 billion rows
- Data is loaded at approximately 6.1 GB per hour

The Queries

- Require the system to access and return test results repeatedly
- Applied: 10 queries simultaneously, multiple times to test query performance and user scalability: better when users increased.

Data Warehouse Specifications.

The software:

Red Brick Warehouse 5.1 (Acquired by Informix Jan. 1999 later acquired by IBM.)

Features a high performance loader, self tuning query parallelism and a family of join technologies.

The hardware:

HP 9000 V2200 Enterprise (16200 MHz 64 bit PA-8200 cpus) working with 2 EMC Symmetrix 3700 Enterprise Storage Systems.

- Server configured with 16 Gb memory, total mirrored storage capacity of the EMC Symmetrix systems was 5.8 Tb 1.5 TB, and 1.5 Tb of disk space was used.
Object Oriented Databases
Based on the Object Oriented Paradigm

O-O started in 1967 with the Simula-67 language for simulations.

Now C++, Ada, Java, Smalltalk, etc. encapsulating data and code that operates on the data ==> object.

Complex objects:
A complex Object in OOdbs is an item that is viewed as a single object in the real world, but it contains other objects. Now, these objects may have arbitrary complex internal structures, which

• Are structured hierarchically or relationally (nested) represent the containment relationship.
• Structured objects are grouped into classes: (sub and super) classes.
• Since the value of a data item is an object, then the data item is also an object.
• It is possible to represent an object containment which in turn results in a composite object.

Examples of OODBs include
• HELIX - early 1980's for MAC
• IRIS - for large bibliographic applications (Library)
• Postgress
And others.

Physical Organization
• **Text data** - is usually treated as byte string manipulated by editors and formatters.
• **Audio data** - is typically a digitized, compressed representation of sound and/or speech, used by separate application software.
• **Video and Graphic data** - video data is bitmap or sets of lines, boxes and geometric objects. Some graphical data is manipulated within the DBMS itself or though special application soft.
Extended Relational Systems (ERDB)

1. Logic Based Data Model

- The query language which is non-procedural and based on first order logic.
- The user describes the information desired without giving a specific procedure for obtaining that info.

**Datalog**: A query language Prolog-like. A language that has more expressive power than SQL. A LISP or SNOBOL-like other languages. All these languages allow for recursive queries.

**Example**

A *datalog* db consists of two types of relations:

1. **Base relations** - which are stored in the db and are the same as those used previously. They are also called **Extensional databases** (EDBs).
2. **Derived relations** - not necessarily stored in the db and are usually temporary relations that hold intermediate results computed during the execution of a query. Sometimes referred to as **Intentional dbs** (IDB)

**Notes**

i. The attributes of a relation are not explicitly named. Rather the position of an attribute determines its value, in contrast to the relational model.
ii. Programs are a finite set of rules involving one or more IDBs or EDBs above.

**Example 1.**

Sample Datalog Code: \( ca(y, x) : deposit("Wonderland", x, y, z) \), \( z > 1200 \)

This rule consists of the base relation *deposit* and the one derived relation *ca*. It derives the set of all pairs \{customer-name, account\} of all the customers having an account at the Wonderland branch with a balance greater than $1200.

**Example 2.**

\( ca("Charlie Brown", 999) : - deposit("Wonderland", 999, "Charlie Brown", w), w > 1200 \)

**Note:**

- variable x by constant 999, and variable y by "Charlie Brown" string, and z by w.
2. The Nested Relational Model

- An extension of the relational model in which *domains* maybe either atomic or relation- valued, a requirement for the First Normal Form.
- Note: a domain is *atomic* if the elements are considered to be indivisible units (e.g. - the set of all integers is an atomic domain but the set of all the sets of integers is *non-atomic domain: because integers do not have subparts but the set of integers does*).
- So the value of a tuple on an Attribute may be a relation and relations may be stored within relations.
- Allows for complex objects to be represented by a single tuple of a nested relation. If we view a tuple of a nested relation as a data item we have a one-to-one correspondence between the data items and objects in the user's view of a db.

**Example** - Document Retrieval (OIS)

For each record store the following:
- Document
- Title
- Author list
- Date
- Keyword list

**NOTE:**
- **Authors**: a document may have several authors. But we might want to find all documents written by "Charlie B" as one of the authors => domain authors or subpart of the domain: "set of authors".
- **Keywords**: if we store a set of keywords for a document we may want to retrieve all documents whose keywords include one or more keywords => The keyword list is non-atomic.
- **Date**: does not have a set value domain => consisting of subfields and is non-atomic (day, month, year)

**This is a NOT a 1NF Relation!!!!!**

<table>
<thead>
<tr>
<th>Title</th>
<th>Author_List</th>
<th>Date</th>
<th>Keyword_List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooking</td>
<td>{Bugs B., Emeril L.}</td>
<td>April 1, 2099</td>
<td>{taste, strategy}</td>
</tr>
<tr>
<td>Baking</td>
<td>{Emeril L., Charlie B.}</td>
<td>December 31, 2097</td>
<td>{taste, family}</td>
</tr>
</tbody>
</table>

Definition: In the nested relational model we define a list of nested relational schemes for each level of nesting

**NOTE**: In the relational model we define a database schema by listing the relation schemata for relations in the db.

So the **doc_schema** for the doc_relation is:
- doc_schema = (title, author_list, date, keyword_list)
- author_list = (author)
- date = (month, day, year)
- keyword_list = (keyword)
Nested Relational Query Languages

Besides all relational algebra operations, we have two more operations:

- **nest**: this is an operator which partitions the tuples of a relation into groups. Each member of the group is aggregated into a single tuple.
- **unnest**: unravels a nested sub-relation. A separate tuple is generated for each tuple of the nested sub-relation.

Examples SQL/NF

1. *Find the documents written by Emeril L. pertaining to "taste".*

   ```sql
   select title from doc where "Emeril L." in author_list
   and "taste" in keyword_list
   ```

2. *Find only the title and the year only of publication of docs written by Emeril L. pertaining to taste.*

   ```sql
   select title, (select year from date), from doc
   where "Emeril L." in author_list and "taste" in keyword_list
   ```

   Note: the second select would not be allowed in SQL.

3. *To unnest doc fully:*

   ```sql
   unnest doc on author_list, keyword_list
   ```

   Result in doc1:

<table>
<thead>
<tr>
<th>Title</th>
<th>Author_List</th>
<th>Date</th>
<th>Keyword_List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooking</td>
<td>Bugs B.</td>
<td>April 1, 2099</td>
<td>taste</td>
</tr>
<tr>
<td>Cooking</td>
<td>Emeril L.</td>
<td>April 1, 2099</td>
<td>taste</td>
</tr>
<tr>
<td>Cooking</td>
<td>Bugs B.</td>
<td>April 1, 2099</td>
<td>strategy</td>
</tr>
<tr>
<td>Cooking</td>
<td>Emeril L.</td>
<td>April 1, 2099</td>
<td>strategy</td>
</tr>
<tr>
<td>Baking</td>
<td>Emeril L.</td>
<td>December 31, 2097</td>
<td>taste</td>
</tr>
<tr>
<td>Baking</td>
<td>Charlie B.</td>
<td>December 31, 2097</td>
<td>taste</td>
</tr>
<tr>
<td>Baking</td>
<td>Emeril L</td>
<td>December 31, 2097</td>
<td>family</td>
</tr>
<tr>
<td>Baking</td>
<td>Charlie B.</td>
<td>December 31, 2097</td>
<td>family</td>
</tr>
</tbody>
</table>
4. *nesting doc1 on author_list*

   **nest doc1 on author as author_list**

   Result in doc2:

<table>
<thead>
<tr>
<th>Title</th>
<th>Author_List</th>
<th>Date</th>
<th>Keyword_List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooking</td>
<td>{Bugs B., Emeril L.}</td>
<td>April 1, 2099</td>
<td>strategy</td>
</tr>
<tr>
<td>Cooking</td>
<td>{Bugs B., Emeril L.}</td>
<td>April 1, 2099</td>
<td>taste</td>
</tr>
<tr>
<td>Baking</td>
<td>{Emeril L., Charlie B.}</td>
<td>December 31, 2097</td>
<td>family</td>
</tr>
<tr>
<td>Baking</td>
<td>{Emeril L., Charlie B.}</td>
<td>December 31, 2097</td>
<td>taste</td>
</tr>
</tbody>
</table>

5. *To get the original one from doc1*

   **nest (nest doc1 on author as author_list) on keyword as keyword_list**

6. Aggregate functions (avg, min, max, sum, count) take a asset as an argument and return a value as their result.

   "Find the title number of authors for each document"

   ```
   select title, count (author_list)
   from doc
   ```

   *Note: since the author list is a relation valued attribute containing one tuple for each author, a count of this nested sub-relation provides the number of authors!*
Expert Data Base Systems and Deductive Databases

EDBS consist of if-then rules. The then clause maybe an arbitrary Transaction in the data base. It continually tests for the satisfaction of the if conditions of the rules - When this occurs, one or more rules are chosen and the corresponding then clause is executed.

- Similar to AI Expert systems in the structure - but employs the use of databases as opposed to (virtual) memory. This is because the data are very large to fit in any size main memory.
- So the whole Expert Data Base System, consists, in its simplest form, of a standard database system and a standard expert system.

Data Mining

It consists of finding interesting trends or patterns in large data sets in order to guide decisions about future activities.

Data Mining Tools should identify these patterns in the data with minimal user input.

An example: A subsidiary of a company and or an enterprise that collects complaints. These complaints have to be sorted out according to some pattern and reported to the appropriate management.

This is a sub-area of Statistics and Artificial Intelligence. The stats also called: "Exploratory Data Analysis" and uses statistical measures; while the AI called Knowledge Discovery and Machine Learning.

Spatial Data Management

The term Spatial Data covers multidimensional points, lines, rectangles, cubes, and other geometric objects. A spatial data object occupies, a certain region of space, called its spatial extend, which is characterized by its location and boundary.

Spatial queries are three main types
   i. Spatial range queries
   ii. Nearest neighbour queries
   iii. Spatial join queries.

Applications involving spatial data:

Geographic Information systems (GIS). Dealing with geometric objects and two or three-dimensional regions. A map contains locations of small objects (points), rivers and highways (lines), cities and lakes (regions). A GIS must manage a multi-dimensional data base, also. Commercial GIS, employ an Object-Relational data Model (see below), such as ArclInfo, which is in wide use today.
Internet Data Bases

Used usually for e-commerce applications, through the use of HTML (Hypertext Markup Language), XML (Extensible Markup Language), CGI (Common Gateway Interface Protocol)

In order to execute a program at the web server's site, the server creates a new process and communicates with this process using CGI protocol, which is the set of rules by which programs on the server can be sent into via the web server to the client. This protocol has: HTML (formatting Language to display content on the web), URLs (addresses used to retrieve the HTML content from the web server), and HTTP which is the language spoken by the web server, giving clients the ability to ask the server for documents.

Here, we have:
- Semi-structured Data Model
- Signature Files which contain an index record for each document in the database. There is an index record, called the signature, for each document.
- OLTP (On Line Transaction Processing)
- Multi-dimensional Data Model
- Heterogeneous DBMSs
- Implemented with Inverted Files (see CPS510) and signature files,
- XML-QL. This is XML's query language that has strong similarities to other query languages in the DBMS field.

Example
The following query extracts data from the XML document by specifying a pattern of markups. Looking for data that is nested inside a BOOK element, a NAME element, and a LASTNAME element. For each part of the XML doc that matches the structure specified by the query, the variable I is bound to the contents of the element LASTNAME. To distinguish variable names from normal text, variable names are prefixed with a dollar sign $.

```
WHERE <BOOK.>
   <NAME><LASTNAME> $I </LASTNAME></NAME>
 </BOOK> IN www.wonderlandbookstore.com/books.xml
CONSTRUCT <RESULTNAME> $I <RESULTNAME>
```

A typical result would be the following XML document:

```
<RESULTNAME>CharlieBrown</RESULTNAME>
<RESULTNAME>BugsBunny</RESULTNAME>
```
Midrange Database Engines

Some midrange engines:

- **mysql** [http://www.mysql.com/](http://www.mysql.com/)
- **postgreSQL**: [http://www.postgresql.org/](http://www.postgresql.org/) From Object Oriented Postgres
- **GNU SQL Server**: The Institute for System Programming at the Russian Academy of Science. [www.ispras.ru/~kim/gss/index.html](http://www.ispras.ru/~kim/gss/index.html) (Note: as of June 26, 2001, this program is no longer being maintained or developed. [http://directory.fsf.org/gnusql.html](http://directory.fsf.org/gnusql.html))
- **beagle**: [www.beaglesql.org](http://www.beaglesql.org) (Note: This url is no longer valid. This 2001 link [http://freshmeat.net/articles/view/305/](http://freshmeat.net/articles/view/305/) suggests beagle is dead.)