Object-Relational vs Object-Oriented DBMSs
Outline

- RDBMS
- OODBMS
- ORDBMS
An Asset Management Scenario

- Dinkey Entertainment Corp.
  - assets: cartoon videos, stills, sounds
  - Herbert films show worldwide
  - Dinkey licenses Herbert videos, stills, sounds for various purposes
    - action figures
    - video games
    - product endorsements
  - database must manage assets and business data
Example App: Asset Management

- Old world: data *models* a business
- New world: data *IS* business
  - 1011010111010100010100111 = $$$$$!
  - software vendors, entertainment industry, direct-mail marketing, etc...
  - this data is typically more complex than administrative data

- Emerging apps mix these two worlds.
The relational model was formally introduced by Dr. E. F. Codd in 1970 and has evolved since then, through a series of writings and later through implementations by IBM and others. The defining standard for relational databases is published by ANSI (the American National Standard Institute) as SQL (ANSI 1986) or SQL1, called SQL-86. A revised standard is called SQL2, also referred to as SQL-92.
A relational database is composed of many relations in the form of two-dimensional tables of rows and columns containing related tuples.

Organizing data into tables, the form in which data is presented to the user and the programmer, is known as the logical view of the database. The stored data on a computer disk system is called the internal view. The rows (tuples) are called records and the columns (fields in the record) are called attributes.
• Each column has a data type (i.e., int, float, date). There are various restrictions on the data that can be stored in a relational database. These are called constraints. The constraints are domain constraints, key constraints, entity integrity constraints, and referential integrity constraints. These constraints ensure that there are no ambiguous tuples in the database.
RDBMSs

• RDBMSs use Structured Query Language (SQL, currently SQL2) as the data definition language (DDL) and the data manipulation language (DML).

• SQL includes statements for data definition, modification, querying and constraint specification. The types of queries vary from simple single-table queries to complicated multi-table queries involving joins, nesting, set union/differences, and others.
Disadvantages of Relational Databases

• All processing is based on values in fields of records. Examples of RDBMSs include Oracle, developed by Oracle Corporation, and Microsoft Access developed by Microsoft.

• The main disadvantages of Relational Databases include their inability to handle application areas like spatial databases (e.g. CAD), applications involving images, special types databases (e.g. complex numbers, arrays, etc.) and other applications that involve complex interrelationships of data.
• The SQL standard enables users to easily migrate their database applications between database systems.

• In addition, users can access data stored in two or more RDBMSs without changing the database sub-language (SQL).
The ORDBMS (like ODBMS or OODBMS) is integrated with an object-oriented programming language.

The characteristic properties of ORDBMS are
- 1) complex data,
- 2) type inheritance,
- 3) object behavior.

Complex data creation in most SQL ORDBMSs is based on preliminary schema definition via the user-defined type (UDT).

Hierarchy within structured complex data offers an additional property, type inheritance.
Object-oriented database management systems (OODBMSs) combine database capabilities with object-oriented programming language capabilities. OODBMSs allow object-oriented programmers to develop the product, store them as objects, and replicate or modify existing objects to make new objects within the OODBMS.

Because the database is integrated with the programming language, the programmer can maintain consistency within one environment, in that both the OODBMS and the programming language will use the same model of representation. Relational DBMS projects, by way of contrast, maintain a clearer division between the database model and the application.

As the usage of web-based technology increases with the implementation of Intranets and extranets, companies have a vested interest in OODBMSs to display their complex data. Using a DBMS that has been specifically designed to store data as objects gives an advantage to those companies that are geared towards multimedia presentation or organizations that utilize Computer-Aided Design (CAD).

Some object-oriented databases are designed to work well with object-oriented programming languages such as Delphi, Ruby, Python, Perl, Java, C#, Visual Basic .NET, C++, Objective-C and Smalltalk; others such as JADE have their own programming languages. OODBMSs use exactly the same model as object-oriented programming languages.
A structured type can have subtypes that reuse all of its attributes and contain additional attributes specific to the subtype.

Another advantage, the object behavior, is related with access to the program objects.

Such program objects must be storable and transportable for database processing, therefore they usually are named as persistent objects.

Inside a database, all the relations with a persistent program object are relations with its object identifier (OID).

All of these points can be addressed in a proper relational system, although the SQL standard and its implementations impose arbitrary restrictions and additional complexity.
• The concept of abstract data types (ADTs) in which the internal data structure is hidden and the external operations can be applied on the object that is specified led to the concept of encapsulation.

• The programming language SMALLTALK, developed by Xerox, was designed to be object-oriented. Other object-oriented programming languages include C++, Java, etc.
The main features of OO programming languages are encapsulation, inheritance and polymorphism.

- **Encapsulation** can be thought as a protective layer that prevents the code and the data from being accessed by other code defined outside the layer.
The process in which one object inherits the properties of a previously defined object is called **inheritance**. Inheritance aids in the reuse of existing definitions for creating new objects.

**Polymorphism** allows the same operator or symbol to have different implementations, depending on the type of objects to which the operator is applied.
• the lack of a defining standard was a drawback for OODBMSs. The Object Data Management Group (ODMG) has proposed a standard known as ODMG-93 or ODMG 1.0 standard, now revised into ODMG 2.0.
The standard consists of the object model, the object defining language (ODL), the object query language (OQL), and the bindings to OO programming languages.

- The ODL and OQL are based on the ODMG data model.
The data model consists of data types, type constructors, etc., and is similar to the SQL report that describes the standard model for relational databases.

The ODL is designed so as to support semantic constructs of ODMG 2.0 object model. It is independent of any programming language. The ODL is used to create object specifications.
• The OQL is designed to work closely with the programming languages for which an ODMG binding is defined such as C++, Java and SMALLTALK.

• The syntax of the OQL queries is similar to the syntax of SQL (a query language for relational databases) with some additional features such as object identity, complex objects, inheritance, polymorphism and relationships.
OODBMSs

- An object-oriented language is the language for both the application and the database. OODBMSs have been integrated with C++, C, Java and LISP.

- The primary interface in an OODBMS for creating and modifying objects is directly via the object language (C++, Java, etc.) using the native language syntax.
Difference between relational databases and OO databases

• The difference between relational databases and OO databases is the way in which relationships are handled.

• In OO databases, the relationships are represented with OIDs, which improves the data access performance.

• In relational databases, relationships among tuples are specified by attributes having the same domain.
The main drawback of OODBMSs has been poor performance. Unlike RDBMSs, query optimization for OODBMs is highly complex.

OODBMSs also suffer from problems of scalability, and are unable to support large-scale systems. Some examples of OODBMSs are O2 (now called Ardent) developed by Ardent Software, and the ObjectStore system produced by Object Design Inc.
The data in object-oriented database management systems (OODBMSs) is managed through two sets of relations, one describing the interrelations of data items and another describing the abstract relationships (inheritance).
Advantages of OODBMSs

• Composite relations and objects
• Hierarchy of class easy to be managed
• Query Language not necessary
• No impedance mismatch (typical of OO programming language interacting with a RDBMS or ORDBMS)
• Dynamic data model
• No primary key management
Disadvantages of OODBMSs

- Problematic change of schema
- Dependence from the programming language
- Lack of ad-hoc queries
object-oriented database management systems (OODBMSs)

- The strong connection between application and database results in less code, more natural data structures, and better maintainability and reusability of code. OO languages, such as C++ or Java, are able to reduce code size by not having to translate code into a database sublanguage such as SQL and ODBC or JDBC.
Introduction to Object-Relational DBMSs

• The basic need of Object-relational database arises from the fact that both Relational and Object database have their individual advantages and drawbacks.

• The isomorphism of the relational database system with a mathematical relation allows it to exploit many useful techniques and theorems from set theory. But these types of databases are not useful when the matter comes to data complexity and mismatch between application and the DBMS.

• An object oriented database model allows containers like sets and lists, arbitrary user-defined datatypes as well as nested objects.
  • This brings commonality between the application type systems and database type systems which removes any issue of impedance mismatch.
  • But Object databases, unlike relational does not provide any mathematical base for their deep analysis
Introduction to Object-Relational DBMSs

- The basic goal for the Object-relational database is to bridge the gap between relational databases and the object-oriented modeling techniques used in programming languages such as Java, C++, Visual Basic .NET or C#.
- However, a more popular alternative for achieving such a bridge is to use a standard relational database systems with some form of object-relational mapping (ORM) software.
- Whereas traditional RDBMS or SQL-DBMS products focused on the efficient management of data drawn from a limited set of data-types (defined by the relevant language standards), an object-relational DBMS allows software developers to integrate their own types and the methods that apply to them into the DBMS.
Several major software companies including IBM, Informix, Microsoft, Oracle, and Sybase have all released object-relational versions of their products. These companies are promoting a new, extended version of relational database technology called object-relational database management systems also known as ORDBMSs.
This article compares and contrasts this new class of database with the relational databases, RDBMS from which they are evolving and also with efficient object-oriented databases, OODBMSs, also known as object databases, ODBMSs.
Does a database supporting complex applications have to be object-oriented?

• A certain group thinks that future applications can only be implemented with pure object-oriented systems.
  – Initially these systems looked promising. However, they have been unable to live up to the expectations.

• A new technology has evolved in which relational and object-oriented concepts have been combined or merged. These systems are called object-relational database systems.
  – The main advantages of ORDBMSs are massive scalability and support for object-oriented features.
The main advantages of extending the relational data model come from *reuse* and *sharing*.

Reuse comes from the ability to extend the DBMS server to perform standard functionality centrally, rather than have it coded in each application.

If we can embed the functionality in the server, it saves having to define it in each application that needs it, and consequently allows the functionality to be shared by all applications.
Disadvantages of ORDBMSs

• The ORDBMSs approach has the obvious disadvantage of complexity and associated increased costs.

• There are proponents of the relational approach that believe the essential simplicity and purity of the relational model are lost with these types of extension.

• There are also those that believe that the RDMSs is being extended for what will be a minority of applications that do not achieve optimal performance with current relational technology.
Disadvantages of ORDBMSs

• Instead of discussing object models, terms like ‘user-defined data types’ are used. The terminology of object-orientation abounds with terms like ‘abstract types’, ‘class hierarchies’, and ‘object models’. However, ORDBMS vendors are attempting to portray object models as extensions to the relational model with some additional complexities.

• This potentially misses the point of object-orientation, highlighting the large semantic gap between these two technologies.
• An ORDBMS supports an extended form of SQL called SQL3 that is still in the development stages. The extensions are needed because ORDBMSs have to support ADT's.

• The ORDBMS has the relational model in it because the data is stored in the form of tables having rows and columns and SQL is used as the query language and the result of a query is also table or tuples (rows).
characteristics of an ORDBMSs

- Base datatype extension,
- Support complex objects,
- Inheritance, and
- Rule Systems.
Users define datatypes

- Object-Relational Database Management Systems (ORDBMSs) allow users to define datatypes, functions and operators. As a result, the functionality of the ORDBMSs increases along with their performance.
An example schema of a student relation which ORDBMS supports

- **STUDENT**(fname, lname, ID, sex, major, address, dname, location, picture)

- Notice: extra attributes "location" and "picture" which are not present in the traditional EMPLOYEE relation of RDBMS. The datatype of "location" is "geographic point" and "picture" is "image".
An RDBMS might commonly involve SQL statements such as these:

```sql
CREATE TABLE Customers (    Id CHAR(12) NOT NULL PRIMARY KEY,
    Surname VARCHAR(32) NOT NULL,
    FirstName VARCHAR(32) NOT NULL,
    DOB DATE NOT NULL
);
SELECT InitCap(Surname) || ', ' || InitCap(FirstName)    FROM Customers
WHERE Month(DOB) = Month(getdate())
    AND Day(DOB) = Day(getdate())
```

Most current SQL databases allow the crafting of custom functions, which would allow the query to appear as:

```sql
SELECT Formal(Id)    FROM Customers
WHERE Birthday(DOB) = Today()
```
Comparison Between RDBMS and ORDBMS (1)

```sql
CREATE TABLE Customers (  
    Id      Cust_Id       NOT NULL PRIMARY KEY,  
    Name    PersonName    NOT NULL,  
    DOB     DATE          NOT NULL
);

SELECT Formal( C.Id )  
    FROM Customers C  
    WHERE BirthDay ( C.DOB ) = TODAY;
```

```sql
SELECT InitCap(C.Surname) || ', ' || InitCap(C.FirstName), A.city  
    FROM Customers C join Addresses A ON A.Cust_Id=C.Id -- the join  
    WHERE A.city="New York"
```

```sql
SELECT Formal( C.Name )  
    FROM Customers C  
    WHERE C.address.city="New York" -- the linkage is 'understood' by the ORDB
```
## The differences between the three approaches

<table>
<thead>
<tr>
<th>Criteria</th>
<th>RDBMS</th>
<th>ODBMS</th>
<th>ORDBMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining standard</td>
<td>SQL2</td>
<td>ODMG-2.0</td>
<td>SQL3 (in process)</td>
</tr>
<tr>
<td>Support for object-oriented features</td>
<td>Does not support; It is difficult to map program object to the database</td>
<td>Supports extensively</td>
<td>Limited support; mostly to new data type</td>
</tr>
<tr>
<td>Usage</td>
<td>Easy to use</td>
<td>OK for programmers; some SQL access for end users</td>
<td>Easy to use except for some extensions</td>
</tr>
<tr>
<td>Support for complex relationships</td>
<td>Does not support abstract datatypes</td>
<td>Supports a wide variety of datatypes and data with complex inter-relationships</td>
<td>Supports Abstract datatypes and complex relationships</td>
</tr>
<tr>
<td>Performance</td>
<td>Very good performance</td>
<td>Relatively less performance</td>
<td>Expected to perform very well</td>
</tr>
</tbody>
</table>
The differences between the three approaches

<table>
<thead>
<tr>
<th>Criteria</th>
<th>RDBMS</th>
<th>ODBMS</th>
<th>ORDBMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product maturity</td>
<td>Relatively old and so very mature</td>
<td>This concept is few years old and so relatively mature feature</td>
<td>Still in development stage so immature</td>
</tr>
<tr>
<td>The use of SQL</td>
<td>Extensive supports SQL</td>
<td>OQL is similar to SQL, but with additional features like Complex objects and object-oriented features</td>
<td>SQL3 is being developed with OO features incorporated in it</td>
</tr>
<tr>
<td>Advantages</td>
<td>Its dependence on SQL, relatively simple query optimization hence good performance</td>
<td>It can handle all types of complex applications, reusability of code, less coding</td>
<td>Ability to query complex applications and ability to handle large and complex applications</td>
</tr>
<tr>
<td>Disadvantage</td>
<td>Inability to handle complex applications</td>
<td>Low performance due to complex query optimization, inability to support large-scale systems</td>
<td>Low performance in web application</td>
</tr>
<tr>
<td>Support from vendors</td>
<td>It is considered to be highly successful so the market size is very large but many vendors are moving towards ORDBMS</td>
<td>Presently lacking vendor support due to vast size of RDBMS market</td>
<td>All major RDBMS vendors are after this so has very good future</td>
</tr>
</tbody>
</table>