CO527: Advanced Database Systems

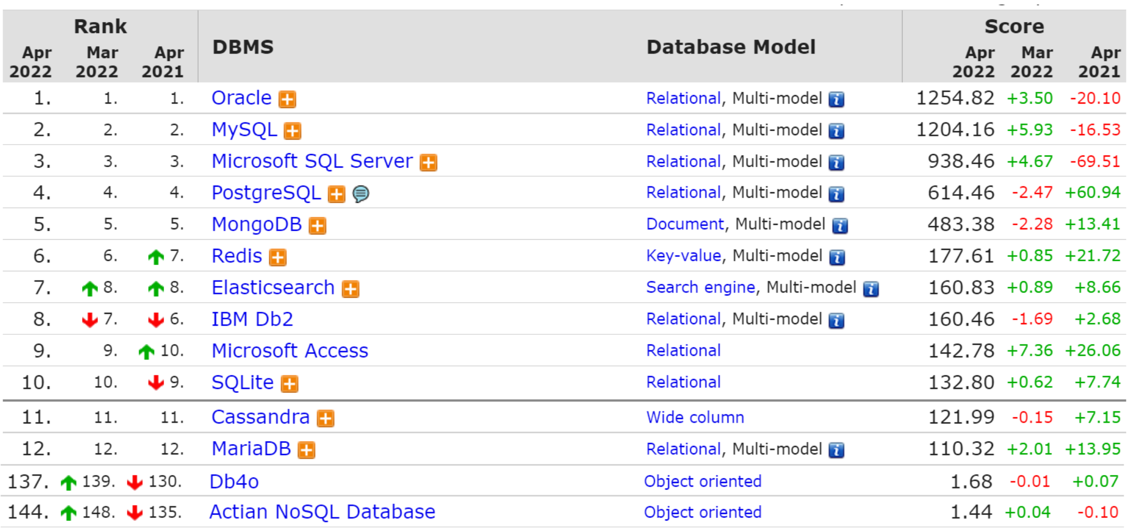
Sampath Deegalla
dsdeegalla@pdn.ac.lk, sampath@eng.pdn.ac.lk

1 Object Oriented Databases

1.1 Overview

- Discuss limitations of the relational data model.
- Introduce object databases, databases that handle complex data types.
- Understand the difference between object oriented-databases and object-relational databases.

DBMS Rankings[1]



Rank			DBMS	Database Model	Score		
Apr 2022	Mar 2022	Apr 2021			Apr 2022	Mar 2022	Apr 2021
1.	1.	1.	Oracle	Relational, Multi-model	1254.82	+3.50	-20.10
2.	2.	2.	MySQL	Relational, Multi-model	1204.16	+5.93	-16.53
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	938.46	+4.67	-69.51
4.	4.	4.	PostgreSQL	Relational, Multi-model	614.46	-2.47	+60.94
5.	5.	5.	MongoDB	Document, Multi-model	483.38	-2.28	+13.41
6.	6.	↑ 7.	Redis	Key-value, Multi-model	177.61	+0.85	+21.72
7.	↑ 8.	↑ 8.	Elasticsearch	Search engine, Multi-model	160.83	+0.89	+8.66
8.	↓ 7.	↓ 6.	IBM Db2	Relational, Multi-model	160.46	-1.69	+2.68
9.	9.	↑ 10.	Microsoft Access	Relational	142.78	+7.36	+26.06
10.	10.	↓ 9.	SQLite	Relational	132.80	+0.62	+7.74
11.	11.	11.	Cassandra	Wide column	121.99	-0.15	+7.15
12.	12.	12.	MariaDB	Relational, Multi-model	110.32	+2.01	+13.95
137.	↑ 139.	↓ 130.	Db4o	Object oriented	1.68	-0.01	+0.07
144.	↑ 148.	↓ 135.	Actian NoSQL Database	Object oriented	1.44	+0.04	-0.10

1.2 Shortcomings of Relational Databases

- On one hand we have a tremendous increase in the amount of data applications have to handle, on the other hand we want a reduced application development time.
  - Object-Oriented programming
  - DBMS features: query capability with optimization, concurrency control, recovery, indexing, etc.
- Can we merge these two to get an object database management system since data is getting more complex?
- A television channel needs to store video sequences, radio interviews, multimedia documents, geographical information, etc., and retrieve them efficiently.

[1] https://db-engines.com/en/ranking/relational+dbms

- A movie producing company needs to store movies, frame sequences, data about actors and theaters, etc. (textbook example)
- A biological lab needs to store complex data about molecules, chromosomes, etc, and retrieve parts of data as well as complete data.

What are the Needs?

- Engineering design and manufacturing (CAD/CAM)
- Biological and other sciences
- Telecommunications
- Geographical Information Systems (GIS)
- Multimedia and image databases
- Document/hypertext databases
- Virtual Worlds
- Games
- List of lists
- User defined data types

Shortcomings with RDBMS

- Supports only a small fixed collection of relatively simple data types (integers, floating point numbers, date, strings)
- No set-valued attributes (sets, lists,...)
- No inheritance in the Is-a relationship
- No complex objects, apart from BLOB (binary large object) and CLOB (character large object)
- Impedance mismatch between data access language (declarative SQL) and host language (procedural C or Java): programmer must explicitly tell how things to be done.

Object/Relational Modeling

- How to map relational data into objects?
- Simply mapping object attributes to fields in a table does not create the perfect mapping.
  - Objects may store collections or relationships with other objects.
  - Relational databases such as MySQL have no way of modeling inheritance.
- Rules of Thumb for Object/Relational Modeling
  - Each persistent class has a corresponding database table.
  - Object fields with primitive datatypes (integers, characters, strings, etc.) map to columns in the associated database table.
  - Each row from a database table corresponds to an instance of its associated persistent class.
  - Each many-to-many object relationship requires a join table just as database entities with many-to-many relationships require join tables.
  - Inheritance is modeled through a one-to-one relationship between the two tables corresponding to the class and subclass.

Existing Object Databases

- Object database is a persistent storage manager for objects:
  - Persistent storage for object-oriented programming languages (C++ or Java)
  - Object-Database Systems:
    - ∗ Object-Oriented Database Systems: alternative to relational systems
    - ∗ Object-Relational Database Systems: Extension to relational systems
- Orion, OpenOODB, Iris, Ode, ENCORE/ObServer Commercial Products: GemStone, ONTOS DB, Objectivity/DB, Versant, FastObjects, ObjectStore, Ardent

DBMS Classification Matrix

Query	Relational DBMS	Object-Relational DBMS
No Query	File System	Object-Oriented DBMS
	Simple Data	Complex Data

1.3 The Concept of Object Data Model

Object Data Model

- The object data model is the basis of object oriented databases, like the relational data model is the basis for the relational databases.
- The database contains a collection of Objects (similar to the concept of entities in E-R model)
- An object has a unique ID (OID) and a collection of objects with similar properties is called a class.
- Properties of an object are specified using Object Definition Language (ODL) and objects are manipulated using Object Manipulation Language (OML).

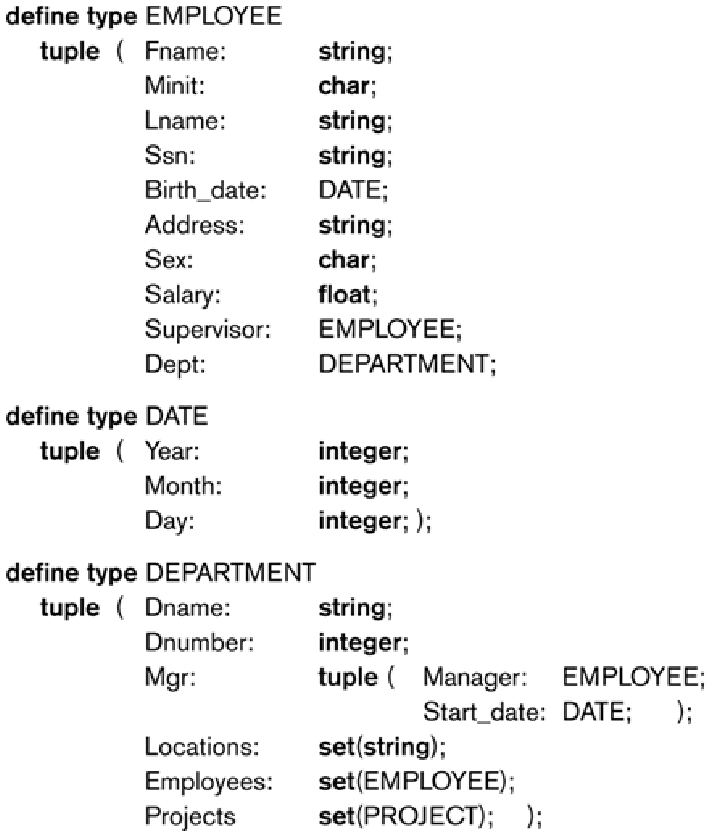
Properties of an Object

- Attributes: atomic or structured type (set, bag, list, array)
- Relationships: reference to an object or set of such objects.
- Methods: functions that can be applied to objects of a class.

Abstract Data Type

- One key feature of object database systems is the possibility for the user to define arbitrary new data types.
- A new data type should come with its associated methods to manipulate it. The new data type and its associated methods is called abstract data type (ADT).
- DBMS has built-in types.
- How does the DBMS deal with new data types that were never seen before.

Type Constructors



```
define type EMPLOYEE
   tuple (  Fname:        string;
            Minit:        char;
            Lname:        string;
            Ssn:          string;
            Birth_date:   DATE;
            Address:      string;
            Sex:          char;
            Salary:       float;
            Supervisor:   EMPLOYEE;
            Dept:         DEPARTMENT;
define type DATE
   tuple (  Year:         integer;
            Month:        integer;
            Day:          integer; );
define type DEPARTMENT
   tuple (  Dname:        string;
            Dnumber:      integer;
            Mgr:          tuple (  Manager:     EMPLOYEE;
                                   Start_date:  DATE;     );
            Locations:    set(string);
            Employees:    set(EMPLOYEE);
            Projects      set(PROJECT);    );
```

Figure 20.2
Specifying the object types EMPLOYEE, DATE, and DEPARTMENT using type constructors.

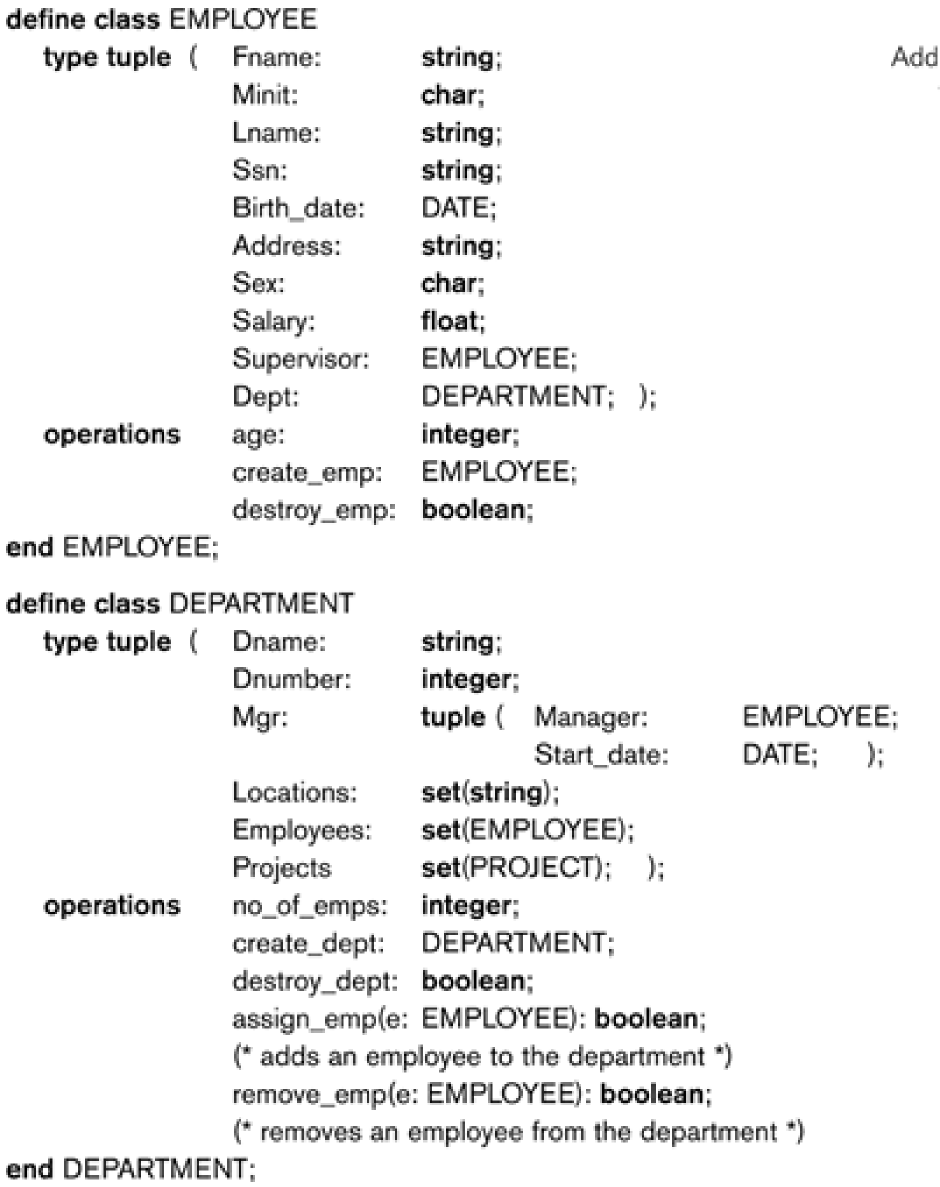
Encapsulation

- Encapsulation = data structure + operations
- It is the main characteristic of object-oriented languages.
- The encapsulation hides the abstract data type internals. ADT= opaque type.
- The DBMS does not need to know how the ADT's data is stored nor how the ADT's methods work. DBMS only needs to know the available methods and how to call them (input/output types of the methods)

Encapsulation

Inheritance

- Type (class) hierarchy
  - System permits the definition of new types based on other existing types
  - A subtype inherits all properties of its supertype
- Class hierarchy
  - A sub-class C' of a class C is a collection of objects such that each object in C' is also an object in C.
  - An object in C' inherits all properties of C
- may change the behaviour of some methods (overloading/overriding of methods)
- typically adds additional attributes and methods



```
define class EMPLOYEE
   type tuple (  Fname:          string;
                 Minit:          char;
                 Lname:          string;
                 Ssn:            string;
                 Birth_date:     DATE;
                 Address:        string;
                 Sex:            char;
                 Salary:         float;
                 Supervisor:     EMPLOYEE;
                 Dept:           DEPARTMENT;   );
   operations    age:            integer;
                 create_emp:     EMPLOYEE;
                 destroy_emp:    boolean;
end EMPLOYEE;

define class DEPARTMENT
   type tuple (  Dname:          string;
                 Dnumber:        integer;
                 Mgr:            tuple (   Manager:        EMPLOYEE;
                                           Start_date:     DATE;     );
                 Locations:      set(string);
                 Employees:      set(EMPLOYEE);
                 Projects        set(PROJECT);     );
   operations    no_of_emps:     integer;
                 create_dept:    DEPARTMENT;
                 destroy_dept:   boolean;
                 assign_emp(e:   EMPLOYEE): boolean;
                 (* adds an employee to the department *)
                 remove_emp(e: EMPLOYEE): boolean;
                 (* removes an employee from the department *)
end DEPARTMENT;
```

Figure 20.3
Adding operations to the definitions of EMPLOYEE and DEPARTMENT.

- Multiple inheritance (inherits from more than just one superclass) e.g. ENGINEERING_MANAGER that is a subtype of both MANAGER and ENGINEER.
- Selective inheritance (inherits only some of the properties of a superclass)

Common Structured Types

- Type constructors are used to combine atomic types and user defined types to create more complex structures:
- ROW(n_1, t_1, $\cdots$ n_n, t_n) : tuple of n fields
- listof(base): list of base-type items
- ARRAY(base): array of base-type items
- setof(base): set of base-type items without duplicates
- bagof(base): multiset of base-type items Not all collection types supported by all systems

Objects, OIDs, and Reference Types

- An object has an identity and the system can generate an object identifier (OID) for objects which is unique in the database across time
- Reference types - REF(basetype) - have object ids as values, i.e., an object of type REF(basetype) is basically a “pointer” to an object of type basetype.

1.4 Object-Oriented Databases

Object-Oriented Databases

- OODBMS aims to achieve seamless integration with an object-oriented programming language such as C++, Java or Smalltalk.
- OODBMS is aimed at applications when an object-centric view point is appropriate. (occasional fetch from object repository)
- No efficient implementations for DML. There are no good optimizations for a query language such as OQL in OODBMSs today.

Current Status

- OODB market growing very slowly these days.
- O-O ideas are being used in a large number of applications, without explicitly using the OODB platform to store data.
- *Growth*: O-O tools for modeling and analysis, O-O Programming Languages like Java and C++
- *Compromise Solution Proposed*: Object Relational DB Management (Informix Universal Server, Oracle 10i, IBM’s UDB, DB2/II . . .)