Object-Oriented Databases

Contents

In this session, we will cover:

- Complex Applications
- RDBMS Weaknesses
- Next Generation Data Models
- Object-Oriented Databases

Relational DBMS Suitability

- Relational DBMS are suitable for certain types of applications
  - simple data types, e.g. dates, strings
  - large number of instances, e.g. students, employees
  - well defined relationships between data, e.g. student, course relationships and use of joins
  - short transactions, e.g. simple queries
- It is currently the most successful for business applications (Online transaction processing (OLPT))

Complex Applications

- RDBMS are inadequate for applications including:
  - CAD, CAM
  - CASE
  - Office Information Systems
  - Multimedia systems
  - GIS
  - Science and medicine

Complex Applications

CAD (Computer Aided Design)

CAD database stores data relating to mechanical and electrical design covering, for example, building, aircraft characterized by:

- a large number of types but few instances of each type
- Designs are very large often with many interdependent subsystems
- Design not static but evolve through time.
- Updates are far-reaching i.e. One change is likely to affect a large number of design objects
- Cooperate engineering - Large number of staff involved with the design and may work parallel on multiple versions of a large design. The end-product must be consistent and coordinated.

Complex Applications

CAM (Computer Aided Manufacturing)

- CAM database stores same data as CAD system, in addition to data relating to discrete production(such as cars on assembly line) and continuous production(such as chemical synthesis were say there is an application that monitors information about state of the system).

Complex Applications

CASE (Computer Aided Software Engineering)

- CASE database Stores data relating to the software development lifecycle
- It is a design system where the whole of the software development lifecycle is being stored in the database.
- This needs to be cooperative in that normally a group of people are working on the design, therefore parts of the design need shared.
- Kinds of things which need stored are code, documentation, questionnaires, etc.

Complex Applications

Office Information and Multimedia Systems

- Stores data relating to the computer control of information in a business which includes:
  - e-mail support
  - documentation
  - SGML documents(Standardized Generalized Markup Language) - an early form of HTML like documents

Complex Applications

Geographic Information Systems

- GIS database stores various types of spatial and temporal information much of which is derived from survey, satellite photographs and tends to be very large.
- Searches may involve identifying features based, for example, on shape, colour, or texture using advanced pattern recognition techniques
- They store images, maps, satellite photos and queries work by pattern recognition and inspecting shapes rather than more traditional business queries.

RDBMS Weaknesses

1. **Poor separation of ‘real world’ entities**
   - In a relational system we force everything into tables and normalise.
   - This looks nothing like the real world and also the join is a very costly operation.
   - normalisation leads to entities that don’t closely match real world
   - This also leads to many joins during query processing hence costly

RDBMS Weaknesses

2. **Semantic overloading**
   - Relational model has only one construct for representing data and relationships between data namely the relation.
     - data and relationships are all forced into tables, so there is no way to differentiate which one is being stored.
   - In representing a many-to-many relationship between two entities we end up creating an extra relation to represent the relationship.
   - no mechanism for differentiation between entities and relationships

RDBMS Weaknesses

3. **Poor support for integrity and enterprise constraints**
   - relational systems good for supporting basic constraints i.e. referential, entity and simple business constraints such as primary key, foreign key
   - However, many commercial systems do not fully support all these constraints and so it is necessary to build them into the application.
   - It is not good for more complex enterprise constraints
4. **Homogeneous data structure**
   - everything is stored in a 2 dimensional table - the real world does not always follow this structure.
   - Vertical homogeneity –data must come from same domain and Horizontal homogeneity- each tuple of a relation must be composed of the same attributes.

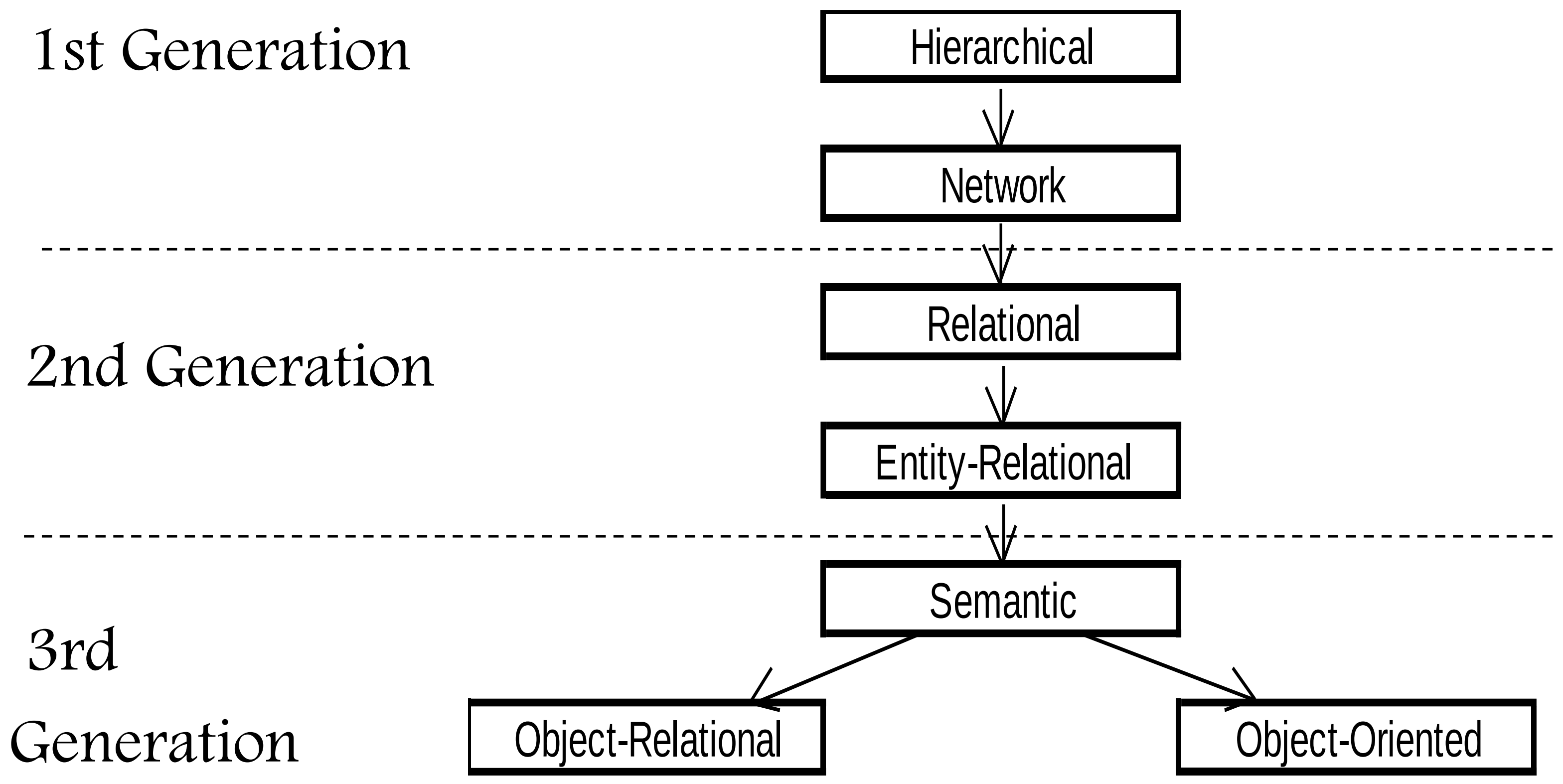
RDBMS Weaknesses

5. **Limited operations**
   - SQL is not computationally complete, e.g. we can not do calculations of the age of a person given date of birth using SQL.
6. **Difficulty handling recursive queries**
   - Recursive queries are impossible with SQL.
   - In the example we may have a table which relates people to their ancestors. To find the parents we join the table with itself once, to find grandparents we need a second join, and so on...
7. **Impedance mismatch**
   - Because SQL is not computationally complete it is sometimes necessary to revert to another programming language, e.g. Java.
   - But the data types in SQL and the programming language are not the same, therefore it is problematic storing the results of queries.

RDBMS Weaknesses

8. **Concurrency, schema changes and poor navigational access**
   - transactions in a relational system are short (seconds) rather than for example in a CAD system where the transaction may be updating a drawing and last hours rather than days.
   - This means the concurrency mechanism will not work.
   - Also, it is difficult to change the schema, e.g. adding new types and changing types by for example adding new columns to a table.
   - Also, some queries need to be navigational rather than based on the contents of tables.

Data Models


1st Generation
Hierarchical
Network
2nd Generation
Relational
Entity-Relational
3rd Generation
Semantic
Object-Relational
Object-Oriented

Data Models

- First generation systems were the hierarchical and network models.
  - Best known hierarchical systems was **IMS** from IBM, developed for the Apollo space program's database storage).
  - Problems with 1st generation systems were complex programs, minimal data independence and no widely accepted theory.
- Relational model was devised in 1970 by Codd to address data independence problem but has limited modelling capabilities.
- In 1976 Chen devised the E-R model which is now a widely used tool for designing relational databases.

Data Models

- In 1980/1981 there was a realisation that models needed to be devised which more closely represented the real world.
  - These can be classed loosely as semantic models, the most well known being **Daplex** devised in 1981 by Shipman.
- Then in the late 80s development work began on object-oriented models, and two models were devised.

Object-Oriented Databases

- Overview and Origins
- OODB Strategies
- OO Database System Manifesto
- Advantages and Disadvantages
- Object Database Standard
  - OQL
  - JDO

OO Databases Overview

- object-oriented databases are becoming much more commonplace nowadays in an attempt to resolve some of the weaknesses of relational databases
- Object-Oriented Database
  - e.g. ObjectStore, Objectivity, Jasmine, POET
  - based on object-oriented programming techniques
  - Information is represented in the form of objects
  - Objects: " A uniquely identifiable entity that contains both attributes that describe the state of a 'real-world' object and the actions associated with it" (Connelly & Begg, 2005)

OO Databases Overview

- include concepts such as
  - user extensible type system,
  - complex objects,
  - encapsulation,
  - inheritance,
  - polymorphism,
  - dynamic binding,
  - object identity

OO Databases Overview

- ODMG (Object Data Management group) standard being devised to define data model and query language standard
- There is no theoretical model as in relational model, but there is a standard developed by the ODMG which defines a data model, query language and an interface standard for data exchange between compliant systems
  - There are not many object-oriented databases around, primarily because of their disadvantages which we will see soon, and because companies such as Oracle have invested so much effort in relational technology.

Origins of OO Databases

The concepts of OO databases are drawn from a number of areas which include:

- Traditional Database Systems
  - Persistence, sharing, transactions, concurrency control, recovery control, security, integrity, querying
- Semantic Data Models
  - generalisation, aggregation, navigational querying
- Object-Oriented Programming
  - object identity, encapsulation, inheritance, types and classes, methods, complex objects, polymorphism, extensibility
- Special Requirements
  - Versioning allowing changes to the properties of objects to be managed in such a way that object references always point to the correct version of an object, schema evolution

OODBMS Development Strategies

- Various approaches to developing an OODBMS:
  - Extend an existing OO-PL with database capabilities e.g. add traditional database capabilities to Smalltalk, Java, C++, etc. Gemstone does this
  - Provide extensible OO DBMS libraries rather than extend the language, add class libraries for persistence, aggregation etc. E.g. ObjectStore, Ontos, Versant
  - Embed OO database language constructs in a conventional host language
    - Equivalent to embedding SQL in a language, O_2 provided embedded extensions to C (and BASIC!)

OODBMS Development Strategies

- Extend an existing database language with OO capabilities
  - Extend SQL with OO concepts – popular, e.g. SQL3, ObjectSQL, Oracle8
- Develop a novel data model/data language
  - Radical, develop an entirely new DBMS + language e.g. SIM (Semantic Information Manager) based on semantic data model

Object Oriented DB System Manifesto

- Developed by Atkinson et. Al. 1989
- Devised 13 mandatory features for an OODBMS
  - based on two criteria:
    - should be an OO system
    - should be a DBMS

Object-Oriented DB System Manifesto

- OO Criteria:
  1. Complex objects must be supported
  2. Object identity must be supported
  3. Encapsulation must be supported
  4. Types or classes must be supported
  5. Types or classes must be able to inherit from their ancestors
  6. Dynamic binding must be supported
  7. The DML must be computationally complete
  8. The set of data types must be extensible

Object-Oriented DB System Manifesto

- DBMS Criteria:

  9. Data persistence must be provided
  10. The DBMS must be capable of managing very large databases
  11. The DBMS must support concurrent users
  12. The DBMS must be capable of recovery from hardware and software
  13. The DBMS must provide a simple way of querying data

OODB ‘Advantages’

- Enriched modelling capabilities
- Extensibility
- Removal of impedance mismatch
- More expressive query language
- Support for schema evolution
- Support for long duration transactions
- Applicability to advanced database applications
- Improved performance

OODB ‘Disadvantages’

- Lack of universal data model
- Lack of experience
- Lack of standards
- Query optimisation compromises encapsulation
- Locking at object level may impact performance
- Complexity
- Lack of support for views
- Lack of support for security

And most importantly…

- What about **integrity**?

Object Database Standard

- Standard for Object-Oriented Data Model proposed by Object Data Management Group
- ODMG Object model is a superset of OMG object model
- Consists of
  - Object model (OM)
  - Object Definition Language (ODL)
  - Object Interchange Format (OIF)
  - Object Query Language (OQL)
  - Language bindings: C++, Smalltalk, Java

Object Model

- Basic Constructs
  - object
  - literals
- both characterised by types
- objects
  - attributes
  - relationships
  - operations

Types

- Interface Definition
  - defines abstract behaviour of object type
  - e.g. interface Employee{..};
- Class
  - defines abstract behaviour and abstract state
  - extended interface with information for ODMS schema definition
  - objects are class instances
  - e.g. class Person{..};
- Literal
  - abstract state of a literal type
  - e.g. struct Complex {float ie, float im;};

Types (cont)

- Inheritance
  - applied to both interfaces and classes
  - inheritance of behaviour between object types
- Extend
  - applied to object types only
  - inheritance of state and behaviour
- Extent
  - set of all instances of a class
    - extension
    - must have a unique key

Objects

- Instances of a class
- Have an unique object identifier
  - remains for lifetime of object
- Names
  - equivalent to global variables
- Lifetime can be
  - transient
  - persistent
  - type and lifetime are independent

Objects

- Collections
  - Set, Bag, List, Array
  - Dictionary - sequenced key-value pairs
- Structured objects
  - Date, Interval, Time, Timestamp
- Literals
  - Atomic, Collection, Structured

Example ODL Schema

```
class Student
(extent students)
{
   attribute short id;
   attribute string name;
   attribute string address;
   attribute date dob;
   relationship set<Module> takes
        inverse Module takenby;
   short age();
};
```

Example ODL Schema

```
class Module
   (extent modules)
{
   attribute string title;
   attribute short semester;
   relationship set<Student> takenby
        inverse Student takes;
};

class Postgrad extends Student
   (extent postgrads)
{
   attribute string thesis_title;
};
```

Object Interchange Format

- Used to dump/load current state of ODBMS to/from a set of files

e.g.
```
Sarah Person{Name "Sarah",
          PersonAddress{Street "Willow Lane",
                   City "Durham",
                   Phone {CountryCode 44,
                   AreaCode 191,
                   PersonCode 1234}}}
```

Object Query Language

- Similar to SQL92
  - extensions: complex objects, object identity, path expressions, polymorphism, operation invocation, late binding
- e.g.

```
select distinct x.age
from Persons x
where x.name = "Pat";
```

- Return literal of type set<struct>

```
select distinct struct(a : x.age, s : x.sex)
from Persons x
where x.name = "Pat";
```

OQL Examples

- Path Expressions

```
select c.address
from Persons p, p.children c
where p.address.street = "Main Street"
and count(p.children) >= 2
and c.address.city != p.address.city;
```

- Methods

```
select max(select c.age from p.children c)
from Persons p
where p.name = "Paul";
```

OQL Polymorphism Examples

- Late Binding

```
select p.activities
from Persons p;
```

- Class Indication

```
select ((Student)p).grade
from Persons p
where “course of study” in p.activities;
```

Java Data Objects

- ODMG disbanded in 2001
- ODMG Java Data Binding superceded by JDO:
  - Provides transparent persistence
  - Scales from embedded to enterprise
  - Integrates with EJB and J2EE
  - Is being widely adopted in the database industry
- More information at www.odmg.org

Assignment

1. Using appropriate examples, demonstrate the use of COMMIT, ROLLBACK and SAVEPOINTS in transaction processing. (10 Marks)
2. With appropriate examples, discuss how Virtual Data Warehousing differs from the common Data Warehousing? (10 Marks)
3. Briefly discuss the implication of using the XML and web as a database platform. (10 Marks)

Further Reading

- Connolly & Begg, chapters 25, 26 and 27
- Date 7th ed., chapter on Object-Oriented databases
- Atkinson et al, Object-Oriented Database System Manifesto, Proc. 1st Intl Conference on DOOD, Japan, 1989.
- Cattell, et. al., The Object Data Standard: ODMG3.0