Advanced Database System

Mod 1 : Object based Database

* Overview :
  - limited type system support in relational model
  - complex apps requires complex data type
  - ORr model extend relational model by providing richer type system including complex types & object orientation
  - Difficulty in accessing db from programs written in JAVA, PL

* Complex Data types →
  - Traditional db has simple types [illegible] that we cannot add [illegible] types for info like an Address [illegible] postal code & writing queries for these will be complicated.
  - A better alternative is structured data types that allow type Address with [illegible] postal code.
  - With complex data types we can represent [illegible] model concepts [illegible] attributes [illegible] represented directly

* Structured types and [illegible]
  - SQL : 1999

** Structured Types :
  - Allows composite attributes [illegible] to be represented directly

```
create type Name as (          create type Address as (
    fname varchar(20),             [illegible]
    lname varchar(20))             [illegible]
final;                             [illegible]
                               not final;
```

↑ we cannot change subtypes   ↑ we can change subtypes

Advanced Database Systems

Mod 1 : Object based Databases

* overview :-
  - limited type system support in relational model —①
  - complex apps requires complex data types
  - ORD model extends relational model by providing richer type system including ~~richer~~ complex type & object orientation.
  - Difficulty in accessing db from programs written in C++, JAVA, PL.

* complex Data types :→
  - Traditional db has simple types in that we cannot add or see specific info like in Address → city, state, street, postal code & writing queries for this would be more complicated.
  - A better alternative is structured data type that allows type Address with subparts street_add, city, postal code
  - with complex data types we can represent E-R model concepts such as composite attributes, generalization, specialization directly.

* Structured Types and Inheritance in SQL :-
  - SQL : 1999

** Structured Types :-
  - Allows composite attributes of E-R designs to be represented directly.

```
create type Name as (
    fname varchar(20),
    lname varchar(20))
final;
```
↑ we cannot change subtypes

```
creat type Address as (
    street varchar(20),
    city varchar(20),
    zip varchar(20))
not final;
```
↑ we can change subtypes.

** Table creation →

```
create table person (
    name Name,
    Address Address );
```

```
create type PersonType as (
    name Name,
    address Address)
    not final
create table person of PersonType;
```

Composite attribute can be accessed using 'dot' notation.

```
Select name.lname, address.city from person;
```

*** constructor functions →

- used to create values of structured types
  ↳ same name as structure

```
* create function Name (fname varchar(20), lname varchar(20))
  returns Name
  begin
      set self.fname = fname;
      set self.lname = lname;
  end
* new Name ('John', 'Mali')

insert into person values (
    new Name ('Vijay', 'Mali'),
    new Address ('20 st', 'Karad', '415104
```

** Type Inheritance :-

We may want to store extra information in db about people who has common properties so we can Inherit type and create new type & add extra info accordingly.

```
create type person (
    name varchar(20),
    Address varchar(20));

create type Student
    under person (
    dept varchar(20));

create type Teacher
    under person (
    salary integer);
```

↑ supertype

Subtypes

- final type → subtype may not be created from given type.
- not final → subtypes may be created.

* If type system supports multiple inheritance we can do it.
  - SQL standard does not support multiple inheritance.

```
create type Combine (TeachingAssistant)
    under type1, type2;
```

Conflict of same attribute from diff types.
To avoid conflict use as keyword.

```
create type combine (TeachingAssistant)
    under type1 with (dept as st_dept),
          type2 with (dept as t_dept);
```

* Table Inheritance :-

1. create table people of Person;
2. create table students of Student under people;
3. create table teachers of Teacher under people;

Multiple inheritance possible with table just as type.

```
create table teaching_assistants of TeachingAssistant
    under students, teachers;
```

** consistency required in table →
SQL does not support multiple inheritance.

* Arrays & Multiset Types in SQL :-

- SQL supports 2 collection types → ① Arrays ② Multisets.
  (Multisets: unordered collection, duplicate elem-)

```
create type publisher as (
    name varchar (20),
    branch varchar (20));
```

```
create type Book as (
    title varchar(20)
    auth_array varchar(20) array[10],
    publisher Publisher,
    key_set varchar(20) multiset);

create table books of Book;

insert into books values(
    'compilers', array['smith','Jones'],
        new publisher('McGraw', 'New-York'),
        multiset['parsing', 'analysis']);
```

we can access & update element by indexing.

```
select author_array[1], auth_array[2]
from books
where title = 'Database System Concepts';
```

* nesting & unnesting →

* Object-Identity & Reference Types in SQL :-

- oops provides the ability to refer to objects
- An attribute of type can be reference to an object of specified type.

```
create type Department (
    name varchar(20),
    head ref(Person) scope people);
Create table dept of Department;
```

* self referencing →

```
create table people of person
    ref is person_id system generated
```

- insert into dept values ('cs', null);
- update dept
```
    set head = ( select people.person_id
                 where name = 'John')
where name = 'cs';
```

Alternative to system generated identifiers is user generated ref.

```
create type Person (
    name varchar(20),
    address varchar(20))
    ref using varchar(20);

create table people of Person
    ref is person_id user generated.

insert into dept values ('cs', '01234567');
```

* Implementing O-R Features :-

- Object relational database systems are basically extensions of existing relational db systems.

- Implementation may choose to represent array and multiset types directly or may choose to use normalized representation internally.

** Persistent programming Languages :-

* Object - Relational Mapping :-

- Object relational mapping systems provide third approach to integration of object-oriented programming languages & databases.
- Built on top of traditional relational systems and allows programmer to define mapping betn tuples in db relations & objects.
  - primary Goal is to ease the job of programmers who build applications by providing them an object model

* Implementing O-R features :-

- object relational database systems are basically extensions of existing relational db systems.

- Implementation may choose to represent array and multiset types directly or may choose to use normalized representation internaly.

* Persistent programming Languages :-

* Object - Relational Mapping :-

- object relational mapping systems provide third approach to integration of object-oriented programming languages & databases.
- Built on top of traditional relational systems and allows programmer to define mapping betn tuples in db relations & objects.
- primary Goal is to ease the job of programmers who build applications, by providing them an object model

Mod 2 : Application development & Administration

* Application programs and UI :

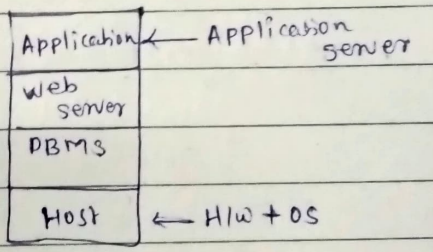
** Application → the way in which user can utilize the technology is called Application.

- End users can use the application
- SQL is native application.
- Application program contains front-end component, which deals with UI, back-end component which communicates with db

** UI →

- first UI is CLI / CUI
- cmd is one of type of UI
- workbench & oracle app is example of GUI.

- client / server architecture allowed the creation of powerful GUI, which terminal based app did not support.
- primary approach for application development
  - technological stack
  - layered architecture.



* Web Fundamentals →

- Http & Https protocols
- process must be stateful in nature.

* Application layer →

UI mostly used in web technology

Model view controller

* Application Architecture :-

- To handle their complexity, large applications are often broken into several layers.

- presentation / UI layer deals with user interaction.
- It broken into layers based on MVC architecture.

  Model → corresponds to business logic layer

  view → defines presentation of data.

  controller → recieves events, execute actions.

- Business logic layer → provides high-level view of data and actions on data.

- Data access layer → provides interface betn business logic & db.
  ↳ provides mapping from object oriented data model used by business logic to the relational model supported by database

* RAD → (Rapid Application Development)

- several approaches have been developed to reduce the effort required to build applications:

  ① provide library functions to generate UI with minimal programming.

  ② provide drag & drop features in an integrated dev environment.

  ③ Automatically generate code for the UI from declarative specification.

* Application performance :-

- Efficiency → For given input what is output
- Throughput → Response Time
- BenchMark → Suite of tasks that are used to quantify the performance of s/w systems

↳ ① Caching → In this connection pooling method is used to reduce this overhead.
- Costs can be further reduced by a caching final web page that is sent in response to request. IP new request came with same parameter then ~~se~~ send response that is cached.

② parallel processing → commonaly used approach to handling such a very heavy loads is to use a large number of application server running in parallel, each handling fraction of request.
- web server or network router can be used to route each client request to one of the application servers.

* Application security :-

#class

security

Application layer
(public access the db)

DB layer
Based on Roles → Authorization → to perform op has permission

Authentication → valid user

↑ To avoid this app should ensure orderAttribute variable value is allowed or not.

** SQL Injection :-

Attacker manages to get application to execute an SQL query created by attacker.

A malicious user can send an arbitrary string in place of meaningful orderAttribute value,

** Cross site scripting and Request forgery :-

- A website that allows user to enter text, such as comments or name and then stores it and later displays it to other users is called (XSS) attack.

- XSS can be done in other ways, such as luring a user into visiting a web site that has malicious scripts embedded in its pages.

* protect → ① Disable HTML tags whatsever in text input by users.

② Instead of using cookie to identify session, session could also be restricted to the IP address from which it was originally authenticated.

** password leakage →

** Application Authentication →

** Audit trails → An log of all changes (inserts, deletes, & updates) along with time and user who has performed.

** privacy → Dont provide all info to researchers.

* Performance Tunning →

- It is process of adjusting parameter to design choices to improve the performance for specific application.

① stored procedures → at server which are precompiled.

② TO support bulk load operations db systems provides bulk import & bulk export utility. (CSV) (TSV) (XML)

③ performance of most systems is usually limited primiarily by the performance of one or few components called bottelnecks

when tuning system we must first try to discover what the bottelnecks and eliminate them by improving performance of system components.

** Tunable parameters :→

Levels → ① Hardware ② DB system ③ schemas & transactions

① Hardware → adding disk, using RAID systems.
add more memory, use faster processor.

② DB system parameters →
- buffer size and checkpointing intervals.

③ schemd transation →
- Administrator can chang schema for tunning.

all tranasactions must follow ACID properties

checkpoints → It is process where the db do recovery
↳ Based on last commit.

Indexing → To retrive data efficienHy we need indexing catlog.
- DB systems supports diff kinds of indexing such as Hash & B-Tree

Suites → combines db tools to a bundle.

* Performance Benchmarks

- suites of tasks that are used to quantify the performance of s/w systems.
- performance of system is measured by suites of standardized tasks called performance benchmarks.
- we can compute system performance accurately in transactions per second for a specific workload.

the correct way to average out the throughputs on different transaction types is to take harmonic mean of throughputs.

* TPC Benchmarks →

- (Transaction Processing performance council) has defined series of benchmark standards for database systems.
- The performance metric is throughput, expressed as transactions per second (TPS).

↳ TPC A, TPC B, TPC E, TPC D, TPC H

* OLTP & OLAP →

Online transactions processing (OLTP) → (UI) →

- works on current data
- Day to day transaction operations
- normalized data structure.
- simple queries
- used by employs, managers
- Require fast response time
- Data in OLTP system is updated real time
- Oracle, MySQL, DB2

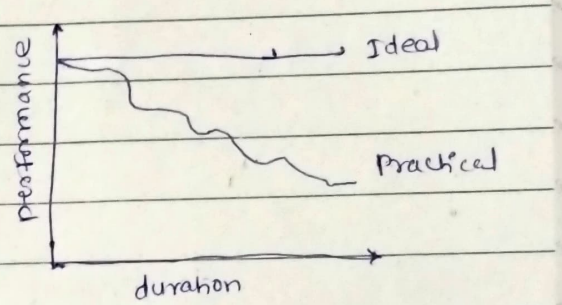
Online Analytical processing →

- works on historical data
- Data analysis & decision making
- Star schema or snowflake schema.
- Complex queries
- Used by Analyst, decision makers
- can have longer response time
- Data in OLAP refreshed periodically
- Tableu, power Bi, SAP

* Issues in application development :-

① Scalability →

- DB are not scalable, because of file, OS
- Size of file increases it decreases the performance.



② UI / UX → design

③ security →