Distributed Database

A logical interrelated collection of shared data (and its description) physically distributed over a computer network.

Distributed DBMS → S/w system that permits the management of distributed database & makes distribution transparent to users.

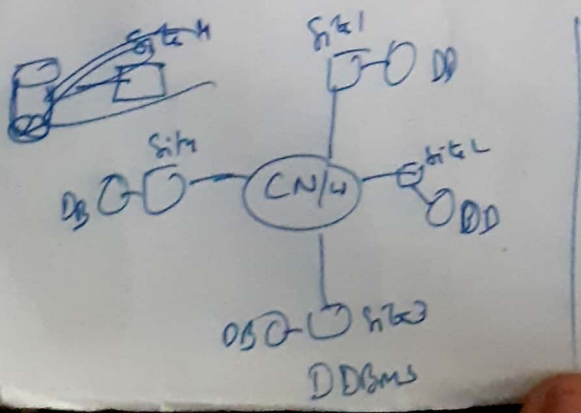
Scenario / Characteristics

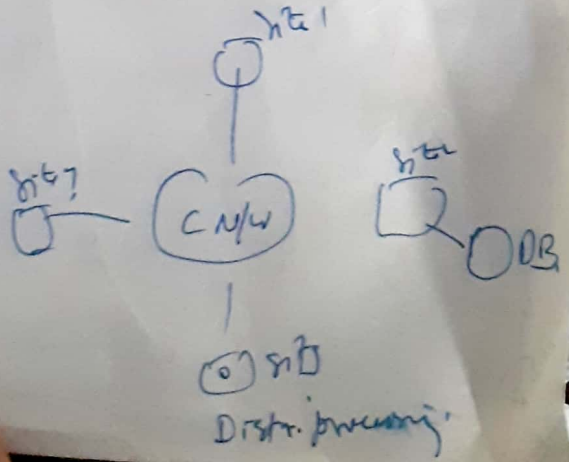
* Collection of logically related shared data.
* data split into no. of chunks (fragments).
* fragments may be replicated
* fragments/replicas are allocated to sites.
* Sites linked by comm. N/w.
* data at each site is under the ~~Control~~ control of DBMS
* Users access the distributed DB through Applications (local or global).
* Distribution transparent (invisible) to user. / so as to appear as centralized sys.

eg:- Banking System.

Diff b/w Distributed Processing & Distributed DB

↳ is a centralized database that can be accessed over a computer N/w.





Distributed Database

Advantages

① Reflects organisation structure : As larg organisation typically spread over several locations.

② Improved share ability & local autonomy. → by local DBMS (DBA)

③ Improved Availability → In Centralized syst, computer failure, terminates application. Entire sys does not fails.

④ Improved Reliability → Through Replication.

⑤ Improved Performance :- data located near the site of "greatest demand". Data Partitioned, so less burden on CPU & I/O Services.

⑥ Economics → it costs much less to create a system of smaller computers with the equivalent power of a single large comp.

⑦ Modular Growth → expansion handled easily.

etc.

Disadv

Complexity. → replication / fragmentation.

Cost : especially maintenance / Labor cost. / N/w ing cost.

Security :- n/w more prone to threats.

Integrity ctrl more difficult.

Lack of standards.

Lack of experience.

DB design more Complex → Diff Platforms.

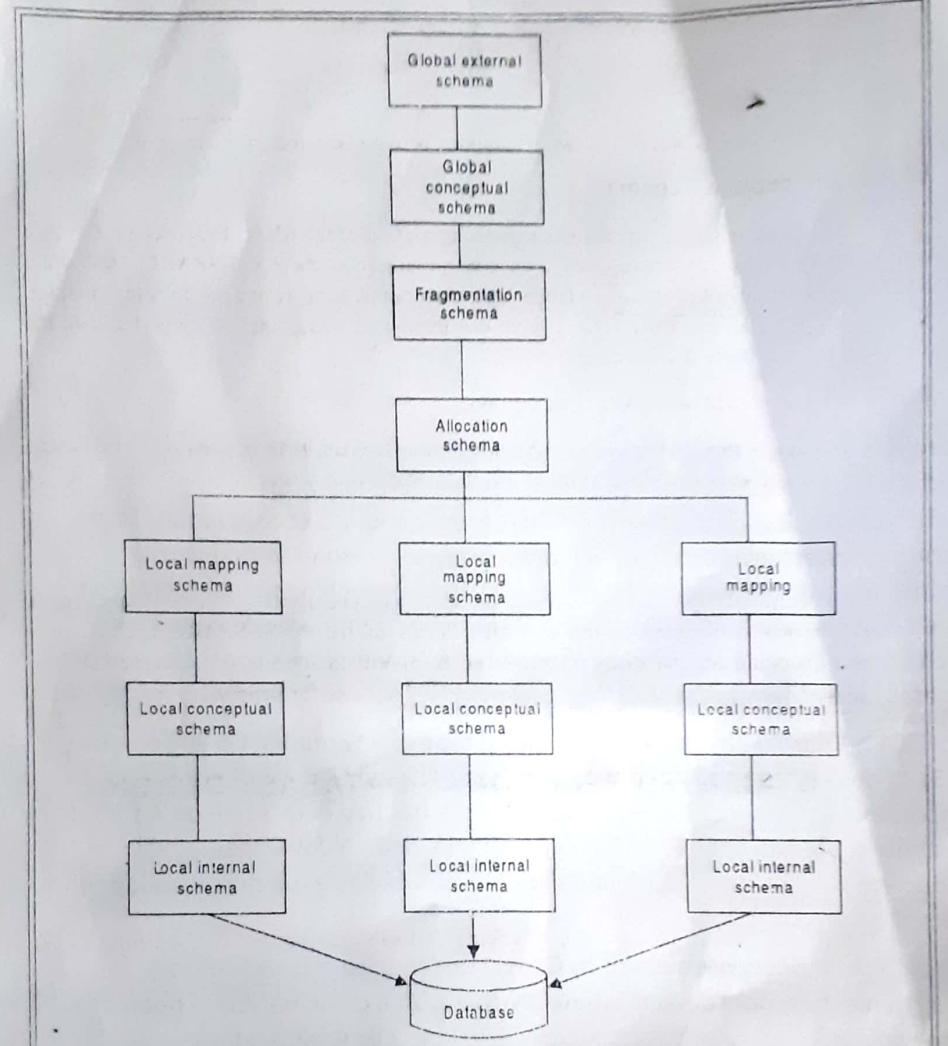


Figure 13.4 Reference architecture of a DDBMS

difficult to present an equivalent architecture that is generally applicable. However, it may be useful to present one possible reference architecture that addresses data distribution. The reference architecture shown in figure consists of the following schemas:

- A set of global external schemas;
- A global conceptual schema;

Data Fragmentation

Horizontal, Vertical, Mixed

Distribution Transparency

Fragmentation Transparency → user doesn't know how data is fragmented
→ highest level of transparency.

Location Transparency: Middle level. May know how fragmented but not know location of data.

Replication " → Closely related to Location T. User unaware of copies of fragments.

Local Mapping Transparency:- Lowest level.. (user needs to specify both fragment names + location of data.

Naming Transparency :- No same names to two items. Create Central naming server / Prefix item with site identifier.
or Aliasing.

T88

Transaction Transparency : ensures that all distributed transactions (subtransactions at sites) maintain ddb integrity + consistency.

Performance Transparency :- to perform as if it is centralized.

DBMS Transparency :- hides knowledge of local DBMS is different.

Distributed System & Comm. n/w → Intro not...

importance of n/w (Possibility) of ddBMS due to...
n/w can be configured in many ways.
Comparison of configuration on the basis of:-

(1) Installation Cost → cost of linking diff sites
(2) Comm. cost → (time + money) to send msg from site A to...
(3) Availability → despite failure of links + sites. (Topology) etc.

Date's Twelve Rules to be DDBMS

(1) Local Autonomy → preference to local data, site
(2) No reliance on central site. → No central servers for service like deadlock, query optimization, & mangement of global System Catalog
(3) Continuous operation
→ No planned system shutdowns for (Adding/Removing Site - fragment creation etc.)
(4) Local Independence → equivalent to Location Transparency.
(5) Fragmentation Independence → user's should access data, no matter how it is fragmented.
(6) Replication Independence
(7) Distributed Query Processing → query referencing data in more than one site.
(8) " Tx Processing :- Comply ACID.
(9) H/w Independence. → possible to run DDBMS on variety of platforms
(10) OS Independency.
(11) n/w "
(12) DB Independency.

2. Data Allocation → placement of data.

Centralized → (No distributed processing).

(2) Fragmented.

(3) Complete Replication (Reliability + availability. Increase along with performance → storage cost & commu. costs for updates are most expensive).

(4) Selective Replication.

Data Replication → storing data (dB) in more than one site or node.

→ improves availability. Can be Complete or Selective.

Adv.

→ Availability

→ Increased Parallelism

→ Less Data Movement over N/W (if has update).

Disadv

Increased overhead on update

Require more hard disk space.

Expensive (req. advanced concurrency control & recovery techniques)

Fragmentation → Reasons of fragmentation.

(1) Usage → seems appropriate to work with subset of data required rather than whole.

(A) Efficiency: as data resides to close to where used frequently.

(B) Parallelism:

(C) Security → less data only local user

Disadv

Performance.

Integrity → difficult.

Homogenous vs heterogenous DDBMS

→ Same DBMS product at all sites.
→ easier to design & manage (H/o)
→ supports incremental growth.

→ diff DBMS/product
→ translations are req. for comm.
- to provide transparency, user must be able to make request in local site language.

Functions

→ Extended

Global External Schemas → logical description of dB as if it were not ddB. Corresponds to conceptual level of ANSI-SPARC, definitions of entities, relationships, constraints, security & integrity info.

Concept of data independence is also provided.

Fragmentation & Allocation Schema

Local Schemas → of Local DBMS.

Distributed Relational DB Design

* Fragmentation
* Allocation
* Replication
* Locality of Reference

→ locality of Reference.
→ Improved Reliability & Availability
→ Acceptable performance { Good allocation may result in bottleneck
Bad may result in underutilization of resources }

Balanced Storage capacities & costs. should be given consideration.

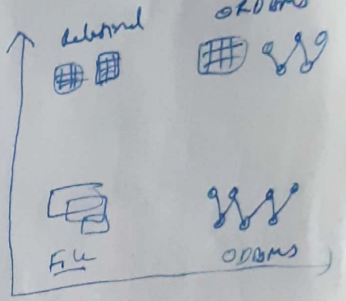
Minimum comm. cost.

ORDBMS

Relational + Object ⇒ ORDBMS.
Base — extensibility

Disadv of RDBMS → No extensibility + but base strong.
ODBMS → extensibility

ORDBMS → Variety of Version. (Models) → Depends upon extension model. (All share base rdbms + Query language).



ORDBMS Advantage

- Reuse + Sharing → Base facility (RDBMS) Reused
- Increased Productivity → for developer & end users
- use of experience in developing RDBMS
- easy to add new functionality

Disadv → Complexity, Costly, → Difficult to model, Confusing extensions.

Comparison of all dbs → verify!.

CREATE Type Student As OBJECT

(
)

CREATE Table S
(
fullstud, Student type

OODBMS & ORDBMS

RDBMS - Weaknesses

* Poor 'real world' entity representation * Semantic overloading.
* " integrity & enterprise constraint support * Homogeneous Data Structure
* Limited operations, Difficult to handle recursive queries,
* other related to concurrency, Schema changes

Object → data + fn (methods) defines behavior.

Entity → only State

object → state & behavior

Current State → by instance (by value of attributes)

→ Each object has OID :- uniquely defined, provides entity integrity.

→ Method: has name + body, Methods Comm.

→ message → object communicates through msgs. (methods).

Class → Group/kind / blue prints for defining objects.

→ attrib & fns defined once in class.

→ Each instance (object) has its own value for attrib.

Abstraction, Encapsulation & Information hiding

identifying essential aspects + ignoring unimportant properties

Separation of external aspects of an object from its internal details

Inheritance → (subclass, Superclass).

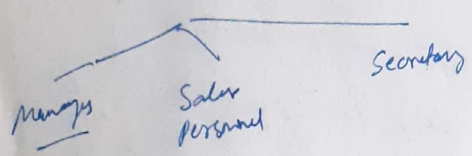
Types of Inheritance

Benefits → Software reusability, Code sharing, Consistency of In

Polymorphism

Storing objects in a Relational DB i.e

mapping classes to Relations → various strategies.



① Map each class/subclass to a relation

staff
manager
Sales personnel
secretary

} Lost semantic info.
No of class which relation represents superclass & which subclass

② Map each subclass to a relation

Manager
Sales Personnel
secretary

} Lost semantic info. in this mapping
Not clear that subclasses are subclasses of single generalization
To produce list of all staff → select tuples from each relation

③ Map hierarchy to single relation.

Staff (with all fields) + type flag

Unused values

→ discriminator attribute

→ Standardized language that describes a logical schema for an OOD.

→ used for preliminary design and conversion of object oriented design to relations using similar methods as conventional ER diagrams & relational databases.

OODB → enables us to create classes, organize objects, structure an inheritance hierarchy and call methods of other classes.
- also provides facilities associated with standard DB systems

Approaches for designing OOD.

→ Designed to store, retrieve & manage objects created by programs written in some OOL such as C++ or java

→ Designed to provide object-oriented facilities to users of non object-oriented prog. langs. such as C or Pascal.

Advantages of OODB

- Enriched Modeling Capabilities → allows "real world" to be modeled.
- Extensibility → allows new data types to be built from existing types.
  - Reusability.
- Can handle large variety of data types → eg:- pictures, voices, video,
- Removal of impedance mismatch → most OOD provide OML that is computationally complete compared with SQL, the standard RDBMS language
- More Expressive Query Language → Recursion etc.
- Support for schema Evolution → type Checking b/w data & applications.
- Support for long-duration Transactions -

→ Applicable to advanced dB applications → CAD, CASE, OIS + Multimedia Systems
→ Improved Performance.

DISADVANTAGES.

- Lack of Universal data model.
- Lack of Experience (use is relatively limited).
- Lack of Standards.
- Competition
- Query optimization Compromises encapsulation
- Locking at object level may impact performance. → in case of concurrency control.
- Complexity.
- Lack of Support for views.
- Lack of Support for Security

Mandatory features to be an OO DBMS

(1) Feature of Persistence → each object should be persistent as such. user should not not have to explicitly move or copy data to make object persistent.

(2) Able to handle Large dB. (using techniques like Data clustering, Data buffering, Query optimization, Index Management)

(3) Controlled Concurrency (requires support by concurrency ctrl sys of DBMS)

(4) Restoring Data Recovery → to the state before crash.

(5) Query facility (6) Construction of Complex objects → take types, lists, arrays, sets from simple objects like integer, char, string etc.

(7) Identity for object (OID). → provides entity integrity.

(8) Feature of Classes + types. → for defining similar objects.

(9) Property of Encapsulation, (10) Inheritance, (11) Late binding, Extensibility (new data type from existing one). (12) Computational Completeness