Chapter-2

The Conceptions of Object Oriented Database Management Systems

It delivers a portrayal of the Object Oriented Database Management Systems conceptions important to the present apriorism and explains the possibility of this apriorism at intervals of this framework. This section aids 2 vital roles. This is often experienced by 1st describing every construct at the side of every mention to the prevailing work. Then where style replacements ascend, the choice preferred by this apriorism is defined literally through the explanations for the choice.

2.1 Object-Oriented Program design Language’s

The motivating power for the origination of OODBMSs is that the need to deliver DB practicability to object’s commencing object oriented program design languages (OOPDL), e.g. C++, C#, Advance-Java, Simula, Smalltalk, Visual Basic etc. The OOPDL object generalization prolongs the information building construct via comprising the subsequent features:

Type: The Structure & the behaviour of an object are concluded the description of its parts & strategies are recommended by type.

Structure: Objects design real-world entities which may include atomic parts, allusions to objects & objects (entrenched objects). Atomic parts are plane elements similar to integers, real, etc. Object allusions remain ‘cursors’ to alternative objects.

Behaviour: strategies are program code which influence & distribute object situation. This difference by typical languages which permit whimsical code to control data-structures.

Uniqueness: Objects require a novel uniqueness freelance of the object situation. The uniqueness helps as a tool to call & find the object.

The perimeters of the graph denote object allusions & are tagged through the titles of the destination objects. In “object-graph”, Objects will be seen as nodes of a directed-graph. A sample of an object-graph is illustrated in fig2.1
2.2 Object-Uniqueness

The effectiveness of OODBMS is determined by the way object uniqueness is enforced. Usually only 3 ways of applying uniqueness is presented: logical-symbol, physical-symbol, & structured-symbol. *Logical-symbols:* The distinctive identifier which is allocated to the object is used by the logical symbols to determine an object. The symbol is not dependent on the locality of the object. Reasonable object repositioning is the benefit of this method. Relaxed object access is the drawback of this style, since there's currently an additional level of insidiousness.

*Physical-symbols:* The quick object admittance is possible because physical symbols are used to determine an object through the physical address of the object but creates object repositioning expensive.

*Structured-symbols* cover each a physical & logical element. The physical element refers to the area where the article exists in & therefore the logical element finds the article inside the area. This methods permits objects to be retrieved a lot of economically than clean logical symbols however a lot of luxuriously than physical-identifiers. By means of this method objects may be transferred inexpensively inside an area however interregional object transfers are costlier.

Lakhamraju, Rastogi, Seshadri &Sudarshan [52] present an economical object rearrangement methodology for the system’s using physical symbols. They present a methodology which “arranges online reorganization possible, by minute or no influence on the reaction times of concomitantly performing transactions & on complete system output”.
Examining the achievement compromises of the altered methodologies is above the ambit of this apriorism. The plan completed in this apriorism abstracts ended the responsibility, i.e., every algorithms suggested can be acclimated in affiliation by any logical, physical, or structured symbols. We are not going to simulate the system behavior because our investigational studies too conceptual ended this concern.

2.3 Functions of Object Oriented Database Management Systems
OODBMS not only offer OOPDL features it is correspondingly responsible for the subsequent practicality
Persistence: introduce to the flexibility to keep up object state once the expiry of program implementation. OODBMSs enable giant group of objects to be keep in secondary “fixed” storage, however OOPDLs solely agree on those objects which will fit in RAM & swap file.
Storage-Management: is a process by which the objects are effectively stored on diskette, RAM & allocation through servers.
Ad hoc query Services: enable explanation of analytical queries that can accomplish actions on groups of objects.
Concurrency-control & Recovery: guarantees that co-occurring admittances to objects have no effect in damage of data reliability. Object state is sure to modification consistently, & is invulnerable to system failures.
In this apriorism we specialize in problems concerning storage management.

2.4 Structural problems
This segment summaries numerous OODBMS structural conception & style substitutes. Additionally the actual style substitute picked-in for this apriorism is similarly described & approved.

2.4.1 Client/Server Layer’s
The conception of a client & server is also utilized by OODBMSs. The requests of user are run by the client & the practicality of the database is delivered via the server. The OODMSs run time system & semantic run time system are consisting by client compulsory to correspond by the server. Economical fixed storage for object’s is implemented by the server by consuming
concurrency-management, the secondary-storage, using recovery, & alternative database protocol’s (e.g. mutiversion, timestamp etc.). Objects are serially retrieved by client programs. The object is ‘inspected’ by dereferencing object pointers one object at one time & therefore ‘traversing the objectgraph’. The exclusion happens once adhoc queries area unit utilized, within which instance action are executed on collections of objects. OODBMS’s runtime is responsible for the services if the object is retrieved by the client, it delivers the object & confirms that the tried action is permitted or not permitted on the object. A lock is required to start & finish the operation because concurrency control protocols work only on locks so a read or a write lock is required. Whenever the client receives the object from the server the postponed client process restarts.

![Diagram of network models](image)

(i) Client-Server
(ii) Peer to Peer

**Fig2.2 Alternate network-models**

### 2.4.2 Network-Models

Only 2 typical network-models designed for Object-oriented database management system’s are there: client-server; & peer to peer:

**Client-server:** As the name implies the client-server model is based on the concepts of many clients and one server. Server gives the services to different client requests. Client computer are simple terminals used to request to the server. Fig2.2 (i) displays a client-server network model. The main benefit of client-server model is that Client computers don’t need to perform heavy calculations because they are very normal machines On the other hand, the drawback is that the servers got to service demands from all users & therefore will turn into a cumbersome once the quantity of clients is huges. Samples of client-server Object-oriented database management system’s are Exodus [12], O2 [30] & ObjectStore [53].

**Peer to peer:** In peer to peer all terminals available in the network takes several amounts of client processes executing & a server process. Data storing is scattered over the different terminal.
Fig2.2 (ii) displays an archetypal peer to peer network structure. The reduction in access costs & decentralization of storage of data is the benefit of this approach this is because the data which is utilized by the local client may be stored on local servers. Server is a less probable to be an obstacle if decentralized data storage there. Augmented system complexity is the drawback of this method. Samples of peer to peer Object-oriented database management system’s are SHORE [13] & platypus [40].

This apriorism scrutinises completion problems encompassing diskette I/O optimization for a standalone single-node of the peer to peer network-model. In this manner the elimination of the network & remote caching behavior of the overall peer to peer network-model is possible & may therefore concentrate our devotion on dropping the consequences of diskette I/O. But the procedures established during this apriorism, though premeditated for this additional controlled model, are generalized to figure with the quality client-server & peer to peer network-models. The exceptional instance of the peer to peer network model is client-server network-model. During this apriorism the standalone peer to peer arrangement deliberated has numerous clients & a single server mutually using a shared cache.

2.4.3 Granularity-of-Caching

Main-memory caches are utilized perfectly to decrease Input/output & network charges in Object-oriented database management system’s. So this is the only purpose that we've got determined to concentrate on main-memory caches, but it's necessary to notice that because the recital alterity amid main-memory & higher-level cache’s expands, a lot of work has to be finished at the upper-level of caching. This necessary issue could be considered for upcoming work.

The grain as a result of which data is cached is a significant difference among caching schemes. The granularity problem can be noticed in agreement regard of 3 options:

Object-Grain: The eviction & storage of data is done at the object-grain. Valuable objects may be mined from a page & retain within the cache, whereas less valuable objects may be rejected is an benefit of this method .This result in enhanced cache utilization, that is especially necessary once the cache size is extremely tiny in comparison to the data-base size. The expense of sustaining data of processor & meta-data at this fine-grain is extraordinary. Leading page objects
ought to be derived one by one interested in the object cache. The Thor systems use such method [57].

*Page-Grain:* The eviction and storage of data is done at the page grain. Low charge of buffer management is the key benefit of this method. The cost is low because the size of the page is fixed & the transfer from disk to memory is at page-grain. A variable sized object is costlier to manage in comparison to rigid-sized pages. If the data is loaded & cached at the similar grain then there is no requirement of performing memory copies. Objects ought to be removed & derived in to the object cache simultaneously filling in object-grained caching. If objects don't seem to be healthy clustered, the cache will include several unusable objects, resulting in high memory wastage and this is main drawback of this method. The page-grained caching utilized by the systems consists of: EXODUS[12], O2[30], platypus [40], and Object Store [53].

*Dual-Grain:* Main-memory is distributed into numerous buffers through this method. Every buffer may be whichever object or page-grained. The main benefit of this method is that the page buffer can only have those pages that are healthy clustered & object buffer are copied with only those objects whose pages are poorly clustered. Kemper & Kossmann [1994] illustrates that dual-grained caching usually overtakes page-grained caching while utilizing the O07 bench-mark [11]. All though top quality clustering cannot be achieved by using naive cluster methods. We have a confidence that page-grained caching can outgo dual-grained caching once top standard cluster algorithm’s are utilized. Dual-grain caching is too utilized within SHORE [13].

Page-grained caching is selected to be analyzed in this apriorism. The rationale for this alternative is 3 fold: the recognition of the page-grained caching (EXODUS[12] , O2[30] ,platypus [40] and Object store [53]); the nice accomplishment of page-grained caching once the system is healthy clustered; & our objective of discovering the results of clustering on OODBS accomplishment. We accept that page-grained caching surpasses each dual & object-grained caching once the system is healthy clustered. This is often as a result of once the system is healthy clustered; the reserves created on decreased cache preservation charges balance any cache space waste charges acquired.

### 2.4.4 Data-transmission-Grain

The granularity has a great affect on the system accomplishment when the data is transmitted among client & server caches. The process of transferring data is possible
only in 2 grains in Object-oriented database management system’s:

*Object-server structure:* during this structure the data-transfer unit is a collection of object’s. Solely those objects that are required are reassigned to the client is the main benefit of this structure. The transmission cost per object is large is the drawback of this structure. The process of transfer is possible for at least one object or a collection of objects in both the cases the cost is high; in the first case if there is only one object is to transferred, the network latency is added for the only object, in the second case for transferring the collection of objects, the cost of the CPU for collecting the specified objects in single unit for transferring; put a large cost on the server. This system is utilized by Thor[57].

*Page-server structure:* Clients receive the data pages from the server. In this structure network latency plays a very important role in low cost per object because there is always one instance of network latency is added for each and every object exist in the page. So, there is not an extra cost required for collecting objects for transferring (as and when required). Several unusable objects could also be transmitted to the client if objects don't seem to be healthy clustered. Several object oriented data-bases routine this tactic: EXODUS [12], O2 [30], platypus [40] & Object Store [53].

During this apriorism we tend to concentrate on lone standalone node of the peer to peer network-model during which client’s & server’s utilized the identical main-memory cache. The transfer of data among server & clients in such model is completed with the common main-memory cache & therefore network latency originating from any page-grained or object-grained data transfer remain fictional. The optimum appropriate to use in page-server structures are the procedures established during this apriorism.

### 2.5 Conclusion

Once studying the various object oriented database management system ideas & architecture replacements, we tend to establish our assessment to discover problems encompassing storage management of standalone peer to peer object oriented database management systems by means of page-grained caching. The standalone structure permits us to avoid network problems, therefore facultative us to specialize in procedures that decrease the results of diskette IO. After all, the procedures which we established during this apriorism can be continued to plan in the common peer to peer multiple-node network model. The reason for choosing peer to peer
network model is that because it permits numerous clients & 1 server to survive on one node of
the system. Page-grained caching is preferred for its quality & its higher presentation once the
system is healthy clustered.