Log-based capture and Procedural Capture are two types of capture techniques used when implementing Primary Site Asynchronous replication.

	Topic	Log Based Capture	Procedural Capture
1	Technique.	In **log-based** Capture, the log maintained for recovery purposes is used to generate a record of updates.	In **procedural** Capture, a procedure (trigger) that is automatically invoked by the DBMS or an application program initiates the Capture process, which consists typically of taking a **snapshot** of the primary copy.
2	Overhead.	Log-based Capture has a smaller overhead than procedural Capture.	Procedural Capture has larger overhead as compared to Log-based capture.
3	Propagation delay.	It is driven by changes to the data so results in a smaller delay in propagating changes from primary copies to secondary copies	It is trigger driven, so there might be delay in propagating changes.
4	Log structure disadvantage.	Disadvantage is that implementing log-based Capture requires a detailed understanding of the structure of the log, which is quite system specific.	There is no requirement to understand the structure of log.
5	Vendor implementation.	A vendor cannot easily implement a log-based Capture mechanism that will capture changes made to data in another vendor's DBMS	A vendor can easily implement procedure-based capture mechanism.
6	Best applicable situation.	Log-based Capture in conjunction with continuous Apply minimizes the delay in propagating changes, and is less expensive substitute to synchronous replication. It is best combination when both primary and secondary copies are part of operational database and when the replicas need to be closely synchronized with primary copy as possible.	Procedural Capture in conjunction with application-driven Apply offer the most flexibility in processing source data and changes before altering the replica; this flexibility is often useful in data warehousing applications where the ability to `clean' and filter the retrieved data is more important than the currency of the replica.

OLTP vs OLAP			
	topics	**OLTP**	**OLAP**
1	Fullform	Stands for On Line transaction processing	Stands for Online Analytical Processing
2	Queries	Narrow, Planned, Short, Simple, frequent queries and/or modifications each involving a small number of tuples. Queries used for running business on current basis.	Few but broad, ad hoc, complex queries, may run for hours. Queries used for supporting managerial decision making involving large number or all tuples.
3	Data access mostly for	Mostly updates	Mostly reads
4	Database sizes	Mb-Tb of data	Gb-Tb of data
5	Type of Data	Raw data representing the current state of business	Summarised Consolidated Data, Historical, Point-in-time (snapshots) and predictions
6	Data updates	Up-to-date data	Data is updated at regular intervals
7	Consistency & Recoverability	Consistency and Recoverability is critical	Consistency and Recoverability is not critical.
8	Primary Users	Clerks, Sales Persons, Administrators	Managers, Business Analysts, Customers.
9	Designed for	Designed for Throughput & Performance	Designed for ease of Flexible access and use.

In Distributed Databases joins of relations at different sites can be very expensive. In the technique 'Fetch as Needed' (page oriented nested loops join/index nested loops join) the cost of shipping tuples dominates the total cost even for fast network) so it is not a good idea to use this technique. The another technique 'Ship to one Site' requires to ship the entire relation to second relation's site to carry out join there, though not all tuples in the first relation join with the tuples of second relation still they are shipped. Techniques 'Semi-joins' and 'Bloom join' identify the tuples that are not guaranteed to join thus we can avoid shipping them.

Semi Joins vs Bloom Joins

	Topic	**Semi-joins**	**Bloom-joins**
1	Idea	Instead of shipping full relation to query site we compute and ship the projection of relation thus avoiding shipping the tuples that do not join.	Instead of shipping the relation or the projection of relation we compute and ship the bit-vector of one relation to the other relation's site & then compute the reduction of second relation there using the same hash function used for computing the bit-vector.
2	Costs	The cost of computing and shipping projections is higher than the corresponding costs in bloom-join strategy	The cost of shipping bit-vector and reducing relation using vector are less than the corresponding costs in semi-join strategy
3	Processing Overhead	Processing cost is significantly more	Processing cost is significantly less than semi-joins

Centralized vs Distributed Databases			
	Topic	**Centralized Database**	**Distributed Database**
1	Data Storage	In centralized database the data(relations) is maintained at single site and the processing of individual transactions is essentially sequential.	In Distributed databases the data(relations) may be fragmented and/or replicated and stored across several different physical sites. Each physical site is managed by a DBMS capable or running independently of other sites.
2	Processing Power	A single server handles the database, thus the processing power is limited	A Collection of servers handle the same database thus the processing power increased very much.
3	Central Site Dependency	Since the database is only at a single server failure of that server renders the system non-funcional.	If a server fails, the only the relevant local site is affected the rest of the system remains functional and available.
4	Expandability	It is difficult to manage the increasing size of database	It is easier to manage the increasing size of global logical database.
5	Architecture & Database Design	Architecture and Database is simple as compared to Distributed DBMS	Architecture and Database design is complex as compared to Centralized Database
6	Cost	Deployment cost and other costs are cheaper	Deployment costs and other costs are higher.
7	Integrity	Integrity control is comparatively simpler	Integrity control is comparatively Difficult.
8	Availability	Availability of data is limited because of single site	Since replication of data is possible it results in increased availability

Parallel vs Distributed Databases			
	Topic	**Parallel**	**Distributed**
1	Motivation	Seeks to improve performance at single site through parallel implementation of operations such as loading, building indexes and evaluating queries by implementing different combinations of multiple processors, memories and disks.	Seeks to increase performance and data availability by maintaining copies of data(relations) or by maintaining fragments of a single logical database across several different physical sites(Homogenous or Heterogeneous) having their own hardware, dbms, servers that can operate independen-tly and are connected via data communication link.
2	Implements	*It Implements* – *Pipeline parallelism:* many machines each doing one step in a multi-step process. – *Partition parallelism:* many machines doing the same thing to different pieces of data	*It implements* - *Distributed Data Independence*: User need not know at which site data is located - *Distributed Transaction Atomicity*: Users must be able to write transactions spanning multiple servers in same way as transactions spanning single server.
3	Architectures	Has three architectures. 1.Shared Memory arch 2.Shared Disk arch 3.Shared Nothing arch	Has three architectures 1.Client Server Arch 2.Collaborating Server. 3.Middleware
4	Query	Query cannot be complex so as to span multiple servers.	Query can be complex so as to span multiple servers.
5	Fragmentation & Replication	Fragmentation & Replication of Data(Relations) not allowed	Fragmentation & Replication of data(relations) allowed.

Relational OLAP v/s Multidimentional OLAP			
	TOPIC	**ROLAP**	**MOLAP**
1	Fullform	Relational Online Analytical Processing.	Multidimensional Online Analytical Processing.
2	Datavolumes	Very large data volumes	Moderate datavolumes
3	Data Access	All data access from warehouse storage	Summary data access from proprietary MDDB(Multidimensional database) while detailed data access from warehouse
4	Language	Uses SQL language.	Uses proprietary data language.
5	Relies on	Relies on existing relational DBMS of the data warehouse.	Uses Multidimensional databases which are vendor's proprietary systems.
6	Storage Structures	Data is stored in form of rows and columns in relational tables. Model displays data to user in form of business dimensions.	Data for analysis is stored in specialized proprietary multidimensional databases where large multidimensional arrays form the storage structures & not the tables.
7	Semantic Layer	A semantic layer of Metadata is required to hide storage structures and to present data multi-dimensionally to user. This layer maps the dimensions to the relational table	Static multidimensional structures are created and stored, so no need for semantic layer or presentation layer for multidimensional views.
8	Datacubes	Pre-fabricated multidimensional cubes are not created beforehand and stored in special databases. ROLAP engine in analytical server creates datacubes on fly	Pre-calculated and pre-fabricated multidimensional datacubes are stored in multidimensional databases by MOLAP engine.
9	Complex analysis	There are limitations on complex analysis functions	There is large library of functions for complex calculations
10	Drill Operation	Drill through lowest level easier, but drill across not always easy	Extensive drill-down and slice-and-dice capabilities.
11	Tools	ORACLE 8I ARBORS SOFWARE ESSBASE	ORACLE EXPRESS SERVER, MICROSTRATEGYS DSS SERVER

OO-DBMS VS OR-DBMS			
	TOPIC	**OODBMS**	**ORDBMS**
1	FULLFORM	Object Oriented DBMS	Object Relational DBMS
2	MOTIVATION	Proposed as an alternative to relational systems. It is defined as programming language with support for persistent objects. Heavily influenced by Object Oriented Programming languages.	Extends relational database systems to accommodate functionality to support broader class of applications.
3	Attempt to	It can be thought of as an attempt to add DBMS functionality to Programming language environment.	It is attempt to provide bridge between Relational and Object Oriented Paradigm
4	Difficulty	It is a sort of persistent object oriented programming. Its difficult to use and does not support query language.	It is pretty easy for sql folks to grasp and supports query language.
5	Language	It has ODL/OQL . OQL is similar to SQL	It has extended SQL
6	Datatypes	ADT's(encapsulation, behavior goes with data), Inheritance supported	Supports ADT's and inheritance too
7	Good for	It is good for complex data, fixed set of manipulations, not good for adhoc queries.	Its good even for adhoc queries.

Pointer Swizzling

In some applications, objects are retrieved into memory and accessed frequently through their oids; dereferencing must be implemented very e_ciently. Some systems maintain a table of oids of objects that are (currently) in memory. When an object *O* is brought into memory, they check each oid contained in *O* and replace oids of in-memory objects by in-memory pointers to those objects. This technique is called **pointer swizzling** and makes references to in-memory objects very fast. The downside is that when an object is paged out, in-memory references to it must somehow be invalidated and replaced with its oid.