ADT Differences:-

OLTP vs OLAP

OLTP	OLAP
▪ application oriented ▪ detailed ▪ accurate, as of the moment of access ▪ serves the clerical community ▪ can be updated ▪ requirements for processing understood before initial development ▪ compatible with the Software Development Life Cycle ▪ performance sensitive ▪ accessed a unit at a time ▪ transaction driven ▪ control of update a major concern in terms of ownership ▪ high availability ▪ managed in its entirety ▪ non redundancy ▪ static structure; variable contents ▪ small amount of data used in a process	▪ subject oriented ▪ summarized, otherwise refined ▪ represents values over time, snapshots ▪ serves the managerial community ▪ is not updated ▪ requirements for processing not completely understood before development ▪ completely different life cycle ▪ performance relaxed ▪ accessed a set at a time ▪ analysis driven ▪ control of update no issue ▪ relaxed availability ▪ managed by subsets ▪ redundancy ▪ flexible structure ▪ large amount of data used in a process

OODBMS vs ORDBMS

OODBMSs add DBMS functionalities to a programming language	ORDBMSs add new data types to RDBMS
Integration with host language	
OODBMS: seamless integration with C++/Small talk	ORDBMS: integration is only through embedded SQL in a host language
• **Application requirement**	
• few large objects fetched occasionally: few disk I/O • long duration transactions on in-memory objects • ability to cache objects in memory	• large collection of data • extensive disk I/O • short transactions
• **Query language**	
• Query processing is relatively inefficient • No standard available	• Query facilities is the centerpiece • SQL-based standards available: SQL3, SQL4

Asynchronous Replication vs Synchronous Replication

	Asynchronous Replication	**Synchronous Replication**
Data Loss	By its nature there may be some data loss [1]	Some solutions will guarantee no data loss [1]
Resilience	2 failures are required for there to be loss of service [2] Failures which lead to data corruption will not be replicated to the second copy of the data [3]	A single failure could lead to the loss of the service [2] Failures which lead to data corruption are faithfully replicated to the second copy of the data [3]
Cost	Asynchronous replication solutions are generally more cost effective	Synchronous replication tends to be considerably more expensive to buy and manage than a comparable asynchronous solution [4]
Performance	Less dependent on very low latency, high bandwidth network links between units of storage	Dependent on very low latency, high bandwidth network links between units of storage [5]
Management	Asynchronous replication within E2K7 is native technology [6]	Synchronous replication solutions will introduce 3rd party software into your design [6]

Differences between OLTP and OLAP

	OLTP	OLAP
Time Scale	This stores current data	This stores History data for analysis
Indexing	Optimizes update performance by minimizing the number of indexes	Optimizes adhoc queries by including lots of indexes
Normalization	This is fully normalized	Possibly partially De-normalized for performance reasons. As this is used for reporting.
Organization	Data stored revolves around business functions	Data stored revolves around information topics.
Stored Values	Stores typically coded data.	Stores descriptive data
Homegenity	Scattered among different databases or DBMS and using different value coding schemes	Centralized in data warehouse. Or in a collection of subject oriented data marts

In the OLAP world, there are mainly two different types: Multidimensional OLAP (MOLAP) and Relational OLAP (ROLAP). Hybrid OLAP (HOLAP) refers to technologies that combine MOLAP and ROLAP.

MOLAP

This is the more traditional way of OLAP analysis. In MOLAP, data is stored in a multidimensional cube. The storage is not in the relational database, but in proprietary formats.

Advantages:

- Excellent performance: MOLAP cubes are built for fast data retrieval , and is optimal for slicing and dicing operations.
- Can perform complex calculations: All calculations have been pre-generated when the cube is created. Hence, complex calculations are not only doable, but they return quickly.

Disadvantages:

- Limited in the amount of data it can handle: Because all calculations are performed when the cube is built, it is not possible to include a large amount of data in the cube itself. This is not to say that the data in the cube cannot be derived from a large amount of data. Indeed, this is possible. But in this case, only summary-level information will be included in the cube itself.
- Requires additional investment: Cube technology are often proprietary and do not already exist in the organization. Therefore, to adopt MOLAP technology, chances are additional investments in human and capital resources are needed.

ROLAP

This methodology relies on manipulating the data stored in the relational database to give the appearance of traditional OLAP's slicing and dicing functionality. In essence, each action of slicing and dicing is equivalent to adding a "WHERE" clause in the SQL statement.

Advantages:

- Can handle large amounts of data: The data size limitation of ROLAP technology is the limitation on data size of the underlying relational database. In other words, ROLAP itself places no limitation on data amount.
- Can leverage functionalities inherent in the relational database: Often, relational database already comes with a host of functionalities. ROLAP technologies, since they sit on top of the relational database, can therefore leverage these functionalities.

Disadvantages:

- Performance can be slow: Because each ROLAP report is essentially a SQL query (or multiple SQL queries) in the relational database, the query time can be long if the underlying data size is large.

- Limited by SQL functionalities: Because ROLAP technology mainly relies on generating SQL statements to query the relational database, and SQL statements do not fit all needs (for example, it is difficult to perform complex calculations using SQL), ROLAP technologies are therefore traditionally limited by what SQL can do. ROLAP vendors have mitigated this risk by building into the tool out-of-the-box complex functions as well as the ability to allow users to define their own functions.

HOLAP

HOLAP technologies attempt to combine the advantages of MOLAP and ROLAP. For summary-type information, HOLAP leverages cube technology for faster performance. When detail information is needed, HOLAP can "drill through" from the cube into the underlying relational data.

MOLAP Vs ROLAP Vs HOLAP:-

MOLAP	ROLAP	HOLAP
This is the more traditional way of OLAP analysis. In MOLAP, data is stored in a multidimensional cube. The storage is not in the relational database, but in proprietary formats	This methodology relies on manipulating the data stored in the relational database to give the appearance of traditional OLAP's slicing and dicing functionality. In essence, each action of slicing and dicing is equivalent to adding a "WHERE" clause in the SQL statement.	HOLAP technologies attempt to combine the advantages of MOLAP and ROLAP. For summary-type information, HOLAP leverages cube technology for faster performance. When detail information is needed, HOLAP can "drill through" from the cube into the underlying relational data.

Difference Distributed Database and Centralized Database:

Distributed Database	Centralized Database
distributed database system keeps its data in storage devices that are possibly located in different geographical locations and managed using a central DBMS.	centralized database keeps its data in storage devices that are in a single location connected to a single CPU

But with distributed databases, this bottleneck can be avoided since the databases are parallelized making the load balanced between several servers. But keeping the data up to date in distributed database system requires additional work, therefore increases the cost of maintenance and complexity and also requires additional software for this purpose.	A centralized database is easier to maintain and keep updated since all the data are stored in a single location. Furthermore, it is easier to maintain data integrity and avoid the requirement for data duplication. But, all the requests coming to access data are processed by a single entity such as a single mainframe, and therefore it could easily become a bottleneck.
designing databases for a distributed database is more complex	designing databases for a centralized database is less complex.

Difference operational systems Vs data warehousing systems

operational systems	**data warehousing systems**
Operational systems are generally designed to support high-volume *transaction processing* with minimal back-end reporting.	Data warehousing systems are generally designed to support high-volume *analytical processing* (i.e. *OLAP*) and subsequent, often elaborate *report generation*.
Operational systems are generally *process-oriented* or *process-driven*, meaning that they are focused on specific business processes or tasks. Example tasks include billing, registration, etc.	Data warehousing systems are generally *subject-oriented*, organized around business areas that the organization needs information about. Such subject areas are usually populated with data from one or more operational systems. As an example, revenue may be a subject area of a data warehouse that incorporates data from operational systems that contain student tuition data, alumni gift data, financial aid data, etc.
Operational systems are generally concerned with *current data*.	Data warehousing systems are generally concerned with *historical data*.
Data within operational systems are generally *updated regularly* according to need.	Data within a data warehouse is generally *non-volatile*, meaning that new data may be added regularly, but once loaded , the data is *rarely changed*, thus preserving an ever-growing *history of information*. In short, data within a data warehouse is generally *read-only*.
Operational systems are generally optimized to perform *fast inserts and updates* of relatively *small volumes of data*.	Data warehousing systems are generally optimized to perform *fast retrievals* of relatively *large volumes of data*.

Operational systems are generally *application-specific*, resulting in a multitude of partially or non-integrated systems and *redundant data*(e.g. billing data is not integrated with payroll data).	Data warehousing systems are generally *integrated* at a layer above the application layer, avoiding data redundancy problems.
Operational systems generally require a *non-trivial level of computing skills* amongst the end-user community.	Data warehousing systems generally appeal to an end-user community with a *wide range of computing skills*, from novice to expert users.

OLAP in FASMI comment……..?

OLAP AND THE DATA WAREHOUSE

As noted on the **DSS page**, On-line Analytical Processing permits a sophisticated multi-dimensional analysis of data that can, in turn, be used for decision making purposes. Though the boundaries of OLAP with respect to other forms of decision support are somewhat vague, OLAP products must provide at least the following minimal set of functions:

- **Roll-up**. The roll-up operation collapses the dimension hierarchy along a particular dimension(s) so as to present the remaining dimensions at a coarser level of granularity.
- **Drill-down**. In contrast, the drill-down function allows users to obtain a more detailed view of a given dimension.
- **Slice**. Here, the objective is to extract a slice of the original cube corresponding to a single value of a given dimension. No aggregation is required with this option. Instead, we are allowing the user to focus in on values of interest.
- **Dice**. A related operation is the dice. In this case, we are defining a subcube of the original space. In other words, by specifying value ranges on one or more dimensions, the user can highlight meaningful blocks of aggregated data.
- **Pivot**. The pivot is a simple but effective operation that allows OLAP users to visualize cube values in more natural and intuitive ways.

While the previous list formally describes the key OLAP functions, it is often helpful for prospective customers or users to think of OLAP systems in more informal terms. The **OLAP Report** has defined a metric they call FASMI or **Fast Analysis of Shared Multidimensional Information**. In essence, FASMI is a means by which to grade or compare OLAP products. The FASMI criteria are presented in the following list:

- **Fast**.: Vendors must be able to efficiently trade off pre-calculation costs and storage requirements with real-time query response. Studies have shown that users are likely to abort queries that take longer than thirty seconds to complete.
- **Analysis**.: Tools should not only provide the five fundamental operations but extras such as times series analysis, currency translation, and data mining capabilities. Most of the business and analytical logic should be available without sophisticated 4GL programming.
- **Shared**.: Security and concurrency control should be available when required. It must be noted however that most OLAP systems assume that user-level updates will not be necessary.
- **Multidimensional**.: This is the key FASMI requirement. Whether implemented with OLAP or MOLAP (discussed below), the user must see the data in subject-oriented hierarchies.
- **Information**.: Applications must be able to handle vast amounts of data. Again, regardless of the server model that is used, good OLAP applications may have to support data cubes

that scale to the terabyte range.

Difference Between Semi Join and Bloom Join

Semi Join vs Bloom Join

Semi join and Bloom join are two joining methods used in query processing for distributed databases. When processing queries in distributed databases, data needs to be transferred between databases located in different sites. This could be an expensive operation depending on the amount of data that needs to be transferred. Therefore, when processing queries in a distributed database environment, it is important to optimize the queries to minimize the amount of data transferred between sites. Semi join and bloom join are two methods that can be used to reduce the amount of data transfer and perform efficient query processing.

What is Semi Join?

Semi join is a method used for efficient query processing in a distributed database environments. Consider a situation where an Employee database (holding information such as employee's name, department number she is working for, etc) located at site 1 and a Department database (holding information such as department number, department name, location, etc) located at site 2. For example if we want to obtain the employee name and department name that she is working for (only of departments located in "New York"), by executing a query at a query processor located at site 3, there are several ways that data could be transferred between the three sites to achieve this task. But when transferring data, it is important to note that it is not necessary to transfer the whole database between the sites. Only some of the attributes (or tuples) that are required for the join need to be transferred between the sites to execute the query efficiently. Semi join is a method that can be used to reduce the amount of data shipped between the sites. In semi join, only the join column is transferred from one site to the other and then that transferred column is used to reduce the size of the shipped relations between the other sites. For the above example, you can just transfer the department number and department name of

tuples with location=”New York” from site 2 to site 1 and perform the joining at site 1 and transfer the final relation back to site 3.

What is Bloom Join?

As mentioned earlier, bloom join is another method used to avoid transferring unnecessary data between sites when executing queries in a distributed database environments. In bloom join, rather than transferring the join column itself, a compact representation of the join column is transferred between the sites. Bloom join uses a bloom filter which employs a bit vector to execute membership queries. Firstly, a bloom filter is built using the join column and it is transferred between the sites and then the joining operations are performed.

What is the difference between Semi Join and Bloom Join?

Even though both semi join and bloom join methods are used to minimize the amount of data transferred between the sites when executing queries in a distributed database environment, bloom join reduces the amount of data (number of tuples) transferred compared to semi join by utilizing the concept of bloom filters, which employ a bit vector to determine set memberships. Therefore using bloom join will be more efficient than using semi join.

Two-phase commit protocol

Two-phase commit is a standard protocol in distributed transactions for achieving ACID properties. Each transaction has a coordinator who initiates and coordinates the transaction (Begg & Connolly 2002, p.749).

In the two-phase commit the coordinator sends a prepare message to all participants (nodes) and waits for their answers. The coordinator then sends their answers to all other sites. Every participant waits for these answers from the coordinator before committing to or aborting the transaction. If committing, the coordinator records this into a log and sends a commit message to all participants. If for any reason a participant aborts the process, the coordinator sends a rollback message and the transaction is undone using the log file created earlier. The advantages of this are all participants reach a decision consistently, yet independently (Skeen).

However, the two-phase commit protocol also has limitations in that it is a blocking protocol (Begg & Connolly 2002, p.749). For example, participants will block resource processes while waiting for a message from the coordinator. If for any reason this fails, the participant will continue to wait and may never resolve its transaction. Therefore the resource could be blocked indefinitely. On the other hand, a coordinator will also block resources while waiting for replies from participants. In this case, a coordinator can also block indefinitely if no acknowledgement is received from the participant. Begg and Connolly suggest that the likelihood of a block happening is rare (2002, p. 749). This is most likely the reason why systems still use the two-phase commit protocol.

Three-phase commit protocol

An alternative to the two-phase commit protocol used by many database systems is the three-phase commit. Dale Skeen describes the three-phase commit as a non blocking protocol. He then goes on to say that it was developed to avoid the failures that occur in two-phase commit transactions.

As with the two-phase commit, the three-phase also has a coordinator who initiates and coordinates the transaction (Begg & Connolly 2002, p.750). However, the three-phase protocol introduces a third phase called the pre-commit. The aim of this is to 'remove the uncertainty period for participants that have committed and are waiting for the global abort or commit message from the coordinator' (Begg & Connolly 2002, p.750). When receiving a pre-commit message, participants know that all others have voted to commit. If a pre-commit message has not been received the participant will abort and release any blocked resources.

Two-phase commit protocol vs Three-phase commit

Two-phase commit protocol

Two-phase commit is a standard protocol in distributed transactions for achieving ACID properties. Each transaction has a coordinator who initiates and coordinates the transaction (Begg & Connolly 2002, p.749).

In the two-phase commit the coordinator sends a prepare message to all participants (nodes) and waits for their answers. The coordinator then sends their answers to all other sites. Every participant waits for these answers from the coordinator before committing to or aborting the transaction. If committing, the coordinator records this into a log and sends a commit message to all participants. If for any reason a participant aborts the process, the coordinator sends a rollback message and the transaction is undone using the log file created earlier. The advantages of this are all participants reach a decision consistently, yet independently (Skeen).

However, the two-phase commit protocol also has limitations in that it is a blocking protocol (Begg & Connolly 2002, p.749). For example, participants will block resource processes while waiting for a message from the coordinator. If for any reason this fails, the participant will continue to wait and may never resolve its transaction. Therefore the resource could be blocked indefinitely. On the other hand, a coordinator will also block resources while waiting for replies from participants. In this case, a coordinator can also block indefinitely if no acknowledgement is received from the participant. Begg and Connolly suggest that the likelihood of a block happening is rare (2002, p. 749). This is most likely the reason why systems still use the two-phase commit protocol.

Three-phase commit protocol

An alternative to the two-phase commit protocol used by many database systems is the three-phase commit. Dale Skeen describes the three-phase commit as a non blocking protocol. He then goes on to say that it was developed to avoid the failures that occur in two-phase commit transactions.

As with the two-phase commit, the three-phase also has a coordinator who initiates and coordinates the transaction (Begg & Connolly 2002, p.750). However, the three-phase protocol introduces a third phase called the pre-commit. The aim of this is to 'remove the uncertainty period for participants that have committed and are waiting for the global abort or commit message from the coordinator' (Begg & Connolly 2002, p.750). When receiving a pre-commit message, participants know that all others have voted to commit. If a pre-commit message has not been received the participant will abort and release any blocked resources.