CLIENT SERVER COMPUTING

- **UNIT IV: Data warehouses – OLTP – Decision Support Systems–Executive Information system–comparing Decision Support and OLTP systems–Production vs Information Databases - The Data Warehouse-Client/Server Transaction Processing – The ACID properties – Transaction Models.**
- TEXT BOOK:
- **Robert Orfali, Dan Harkey& Jeri Edwards, "Client/Server Survival Guide", Wiley INDIA Edition, 3rd Edition, 2011.**
- **Prepared by : B.Loganathan**

Data warehouse : Definition

- A warehouse as a separate database for decision support, which typically contains vast amounts of information.
- A collection of data objects that have been inventoried for distribution to a business community.
- A warehouse is an active intelligent store of data that can manage and aggregate information from many sources, distribute it where needed, and activate business policies.

Elements of Data Warehousing

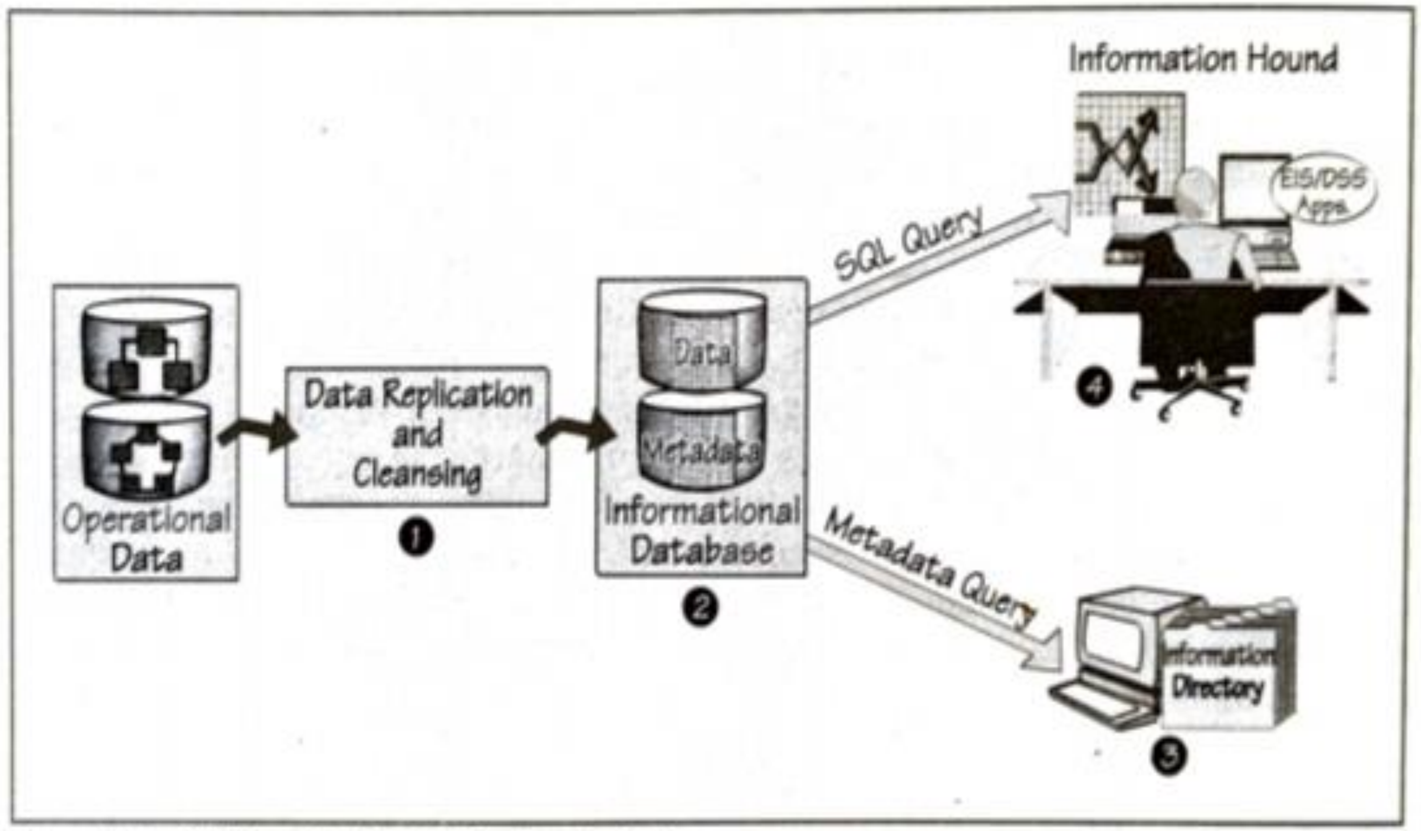


Figure 12-1. The Elements of a Data Warehousing System.

- The first step on the road to data warehousing is to understand the constituent elements that make up a solution. Almost all data warehousing systems provide the following four elements. (see the above diagram)
- **1.The data replication manager-**It sometimes called "the warehouse manager"-manages the copying and distribution of data across databases as defined by the information hound. The hound defines the data that needs to be copied, the source and destination platforms, the frequency of updates, and the data transforms.

- **2.The informational database** is a relational database that organizes and stores copies of data from data sources in a format that meets the needs of information hounds.
- The decision-support server that transforms, aggregates, and adds value to data from various production sources. It also stores metadata (or data about data) that describes the contents of the informational database.

- **3.The information directory** combines the functions of a technical directory, business directory, and information navigator.

- Its primary purpose is to help the information hound find out what data is available on the different databases, what format it's in, and how to access it. It also helps the DBAs manage the data warehouse.
- **4.EIS/DSS tool support** (Executive Information System/Decision Support System) is provided via SQL. Most vendors support ODBC and some other protocol. Some vendors-for example, Red Brick-provide extended SQL dialects for fast queries and joins. The tools are more interested in sequential access of large data quantities than access to a single record.

Client/Server Transaction Processing : The ACID Properties

- ACID stands for Atomicity, Consistency, Isolation, and Durability.
- A transaction is an action that changes the state of an enterprise-for example, a customer depositing money in a checking account constitutes a banking transaction. Technically speaking, a transaction is a collection of actions imbued with ACID properties.

- **Atomicity** means that a transaction is an indivisible unit of work: all of its actions succeed or they all fail. It's an all-or-nothing proposition. The actions under the transaction's umbrella may include the message queues, updates to a database, and the display of results on the client's Screen. Atomicity is defined from the perspective of the consumer of the transaction.
- **Consistency** means that after a transaction executes, it must leave the system in a correct state or it must abort. If the transaction cannot achieve a stable end state, it must return the system to its initial state.

- **Isolation** means that a transaction's behavior is not affected by other transactions that execute concurrently. The transaction must serialize all accesses to shared resources and guarantee that concurrent programs will not corrupt each other’s operations.

- **Durability** means that a transaction's effects are permanent after it commits. changes should survive system failures. The term "persistent" is a synonym for "durable."

OLTP

- *What is OLTP?*
- Database centred client/server applications fall into two categories : Decision Support System (DSS),Online Transaction Processing (OLTP). These categories provides dramatically different types of business solutions.
- OLTP systems are used to create applications in all walks of business.
- These include reservation system, point of sale, tracking system, inventory control etc…

- These are typically mission critical application that require 1-3 response time 100% of the time.
- The number of clients support by an OLTP system may vary dramatically throughout the day, week or year, but response time must be maintained.
- It applications require a tight control over the security and integrity of the database.
- The reliability and availability of the overall system must be very high.
- Data must be kept consistent and correct.
- In OLTP, the client typically interacts with the transaction server instead of database server.

- These interaction is necessary to provide the high performance.
- Transaction servers come in two flavours : *OLTP Lite*, provided by the stored procedures and *OLTP Heavy*, provided by TP monitors.
- OLTP applications require code to be written for both client component and server transactions.
- *What is Decision-Support System?*
- Decision Support System is used to analyze data and create reports.
- It provides the business professional and information hounds with a means to obtain exactly the information they need.

- DSS provides the user with flexible access to data and the tools to manipulate and present the data in all kinds of report formats.
- This systems are not generally time-critical and can tolerate less response time.
- Client/server decision support systems are typically not suitable for mission-critical production environments.
- They have poor integrity controls and limited multiple access capabilities.
- These are built using a new generation screen-layout tools that allow non-programmers to build GUI front-end and reports by painting, pointing and clicking.

- ***What is an Executive Information System?***
- Executive Information System (EIS) are even more powerful, easy to use and business-specific the DSS tools.
- They are certainly more expensive.
- The EIS tools are recently expanded their scope and offer a broader range of functions at the enterprise level.
- “E” in EIS stands for “Enterprise” instead of “Executive” because this system now have a hundreds of users with many roles such as executive, manager and business analyst.
- Some vendors prefer to call them “Everyone’s Information System” while still charging a small fortune for their tools.
- The evolving EIS/DSS system referred to as Online Analytical Processing (OLAP) or Multidimensional Analysis (MDA) tools.

Comparing Decision Support and OLTP systems

- Decision support systems application can be created directly by the user.
- Network administrators are still needed to help set up the client/server system and Database administrator (DBA) may help assemble collections of views, columns and tables that are relevant to the user.
- The design of client/server systems for OLTP is a lot more involved.

Comparing the Programming Effort for Decision Support and OLTP

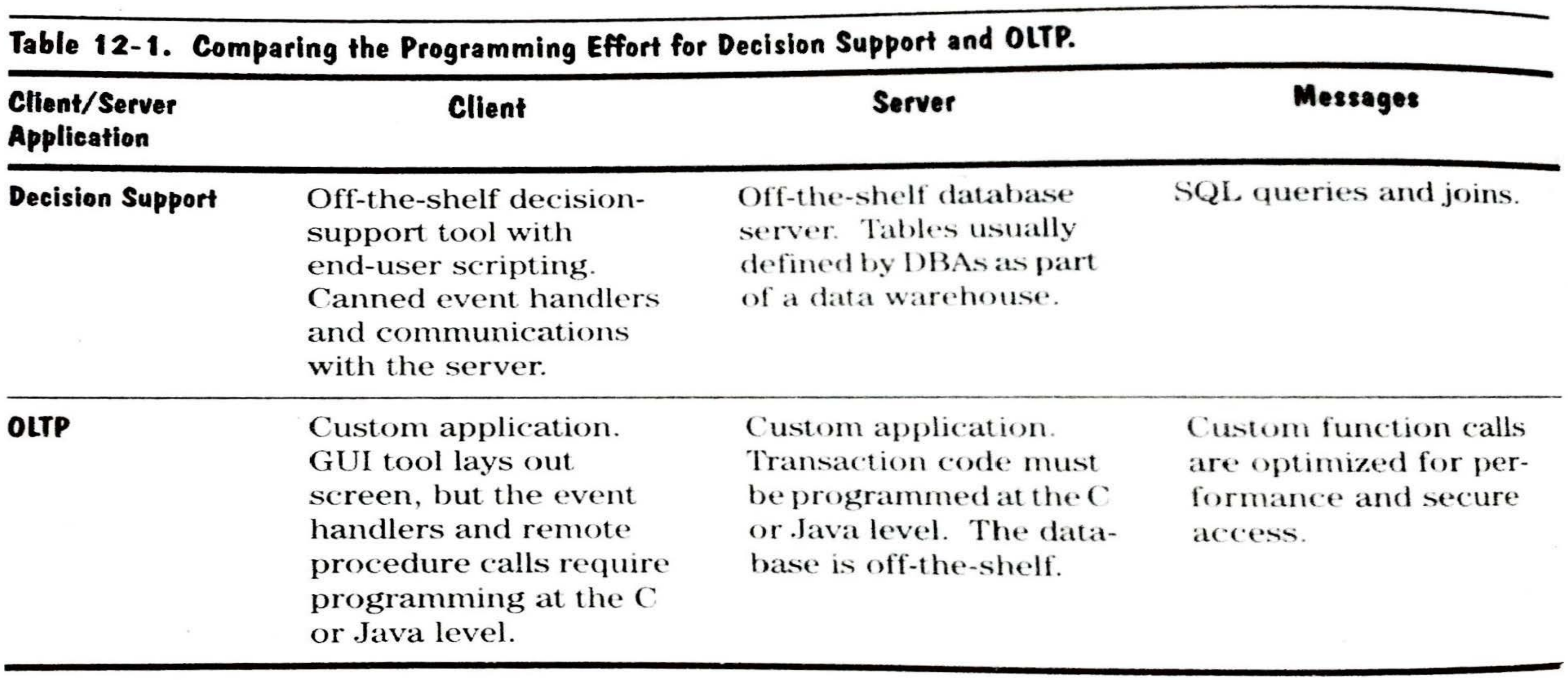


Table 12-1. Comparing the Programming Effort for Decision Support and OLTP.

Client/Server Application	Client	Server	Messages
Decision Support	Off-the-shelf decision-support tool with end-user scripting. Canned event handlers and communications with the server.	Off-the-shelf database server. Tables usually defined by DBAs as part of a data warehouse.	SQL queries and joins.
OLTP	Custom application. GUI tool lays out screen, but the event handlers and remote procedure calls require programming at the C or Java level.	Custom application. Transaction code must be programmed at the C or Java level. The database is off-the-shelf.	Custom function calls are optimized for performance and secure access.

Decision needs : OLTP versus DSS

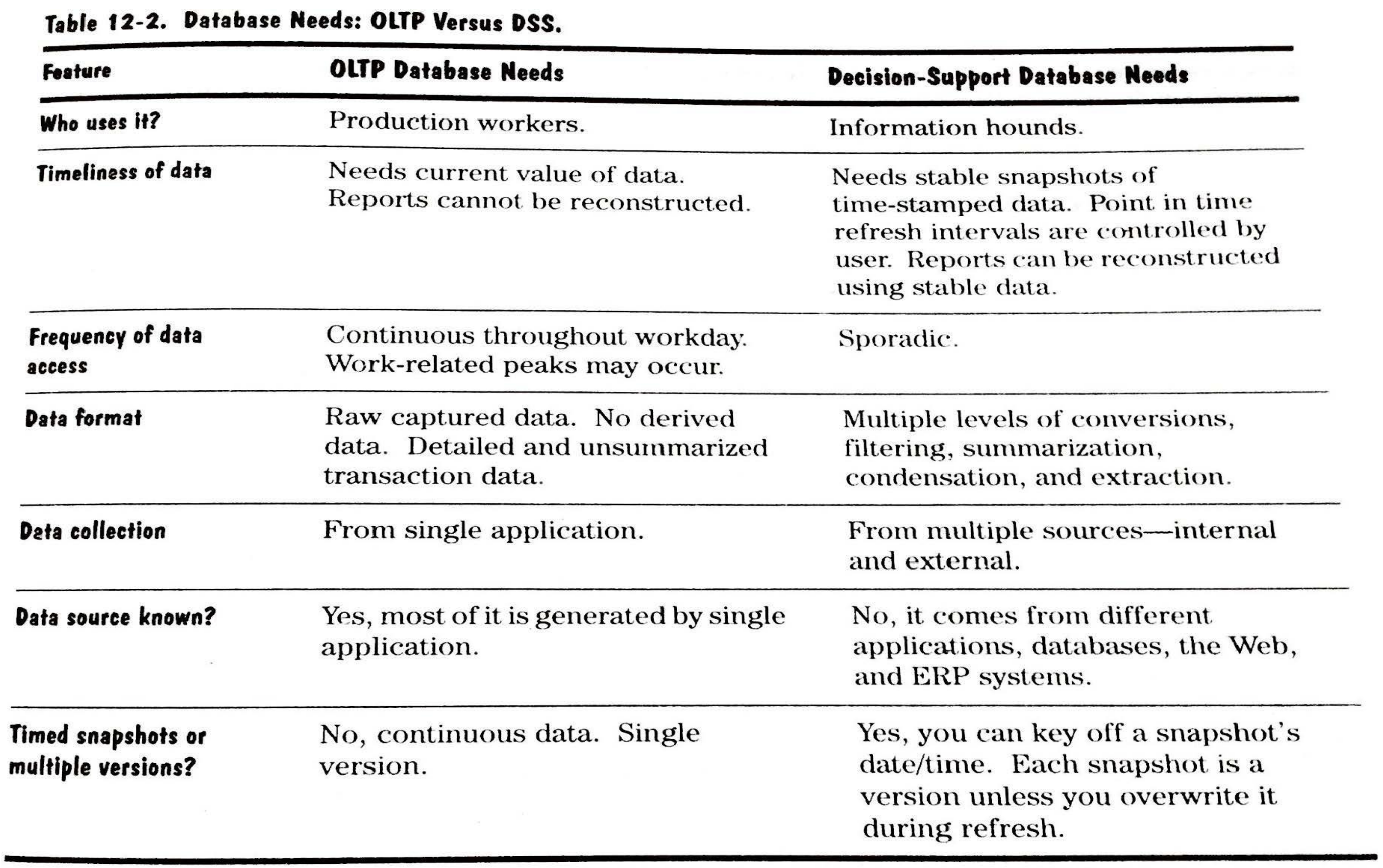


Table 12-2. Database Needs: OLTP Versus DSS.

Feature	OLTP Database Needs	Decision-Support Database Needs
Who uses it?	Production workers.	Information hounds.
Timeliness of data	Needs current value of data. Reports cannot be reconstructed.	Needs stable snapshots of time-stamped data. Point in time refresh intervals are controlled by user. Reports can be reconstructed using stable data.
Frequency of data access	Continuous throughout workday. Work-related peaks may occur.	Sporadic.
Data format	Raw captured data. No derived data. Detailed and unsummarized transaction data.	Multiple levels of conversions, filtering, summarization, condensation, and extraction.
Data collection	From single application.	From multiple sources—internal and external.
Data source known?	Yes, most of it is generated by single application.	No, it comes from different applications, databases, the Web, and ERP systems.
Timed snapshots or multiple versions?	No, continuous data. Single version.	Yes, you can key off a snapshot's date/time. Each snapshot is a version unless you overwrite it during refresh.

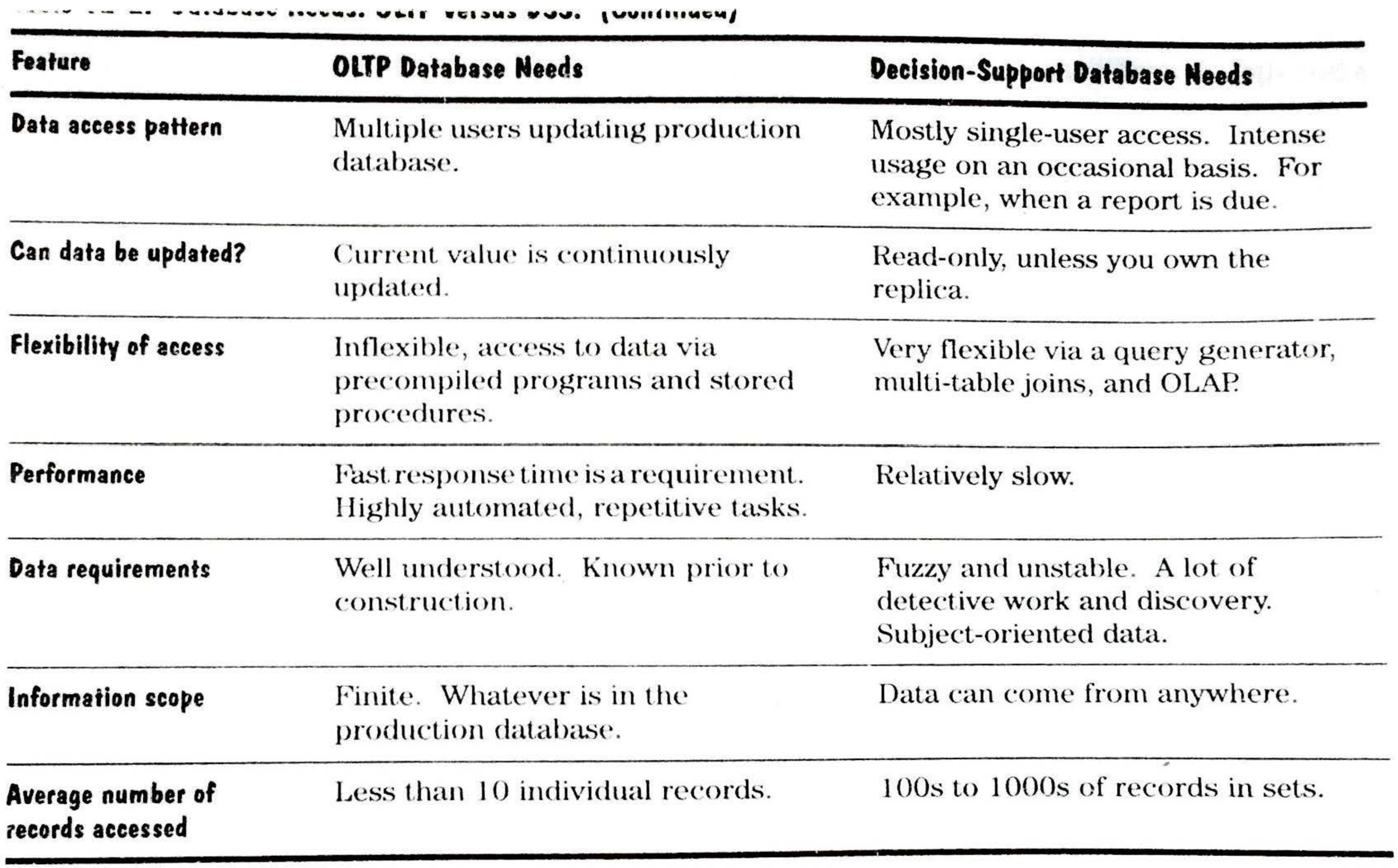


Table 12-2. Database Needs: OLTP versus DSS. (Continued)

Feature	OLTP Database Needs	Decision-Support Database Needs
Data access pattern	Multiple users updating production database.	Mostly single-user access. Intense usage on an occasional basis. For example, when a report is due.
Can data be updated?	Current value is continuously updated.	Read-only, unless you own the replica.
Flexibility of access	Inflexible, access to data via precompiled programs and stored procedures.	Very flexible via a query generator, multi-table joins, and OLAP.
Performance	Fast response time is a requirement. Highly automated, repetitive tasks.	Relatively slow.
Data requirements	Well understood. Known prior to construction.	Fuzzy and unstable. A lot of detective work and discovery. Subject-oriented data.
Information scope	Finite. Whatever is in the production database.	Data can come from anywhere.
Average number of records accessed	Less than 10 individual records.	100s to 1000s of records in sets.

Transaction Models

- Flat transactions: These are the workhorses of the current generation of transactional systems. They're called flat because all the work done within a transaction's boundaries is at the same level (see shaded area in the diagram).
- The transaction starts with begin _transaction and ends with either a commit _transaction or abort _transaction. It's an all-or-nothing proposition and there is no way to commit or abort parts of a flat transaction. All the actions are indivisible, which is what we wanted in the first place.

Flat Transaction

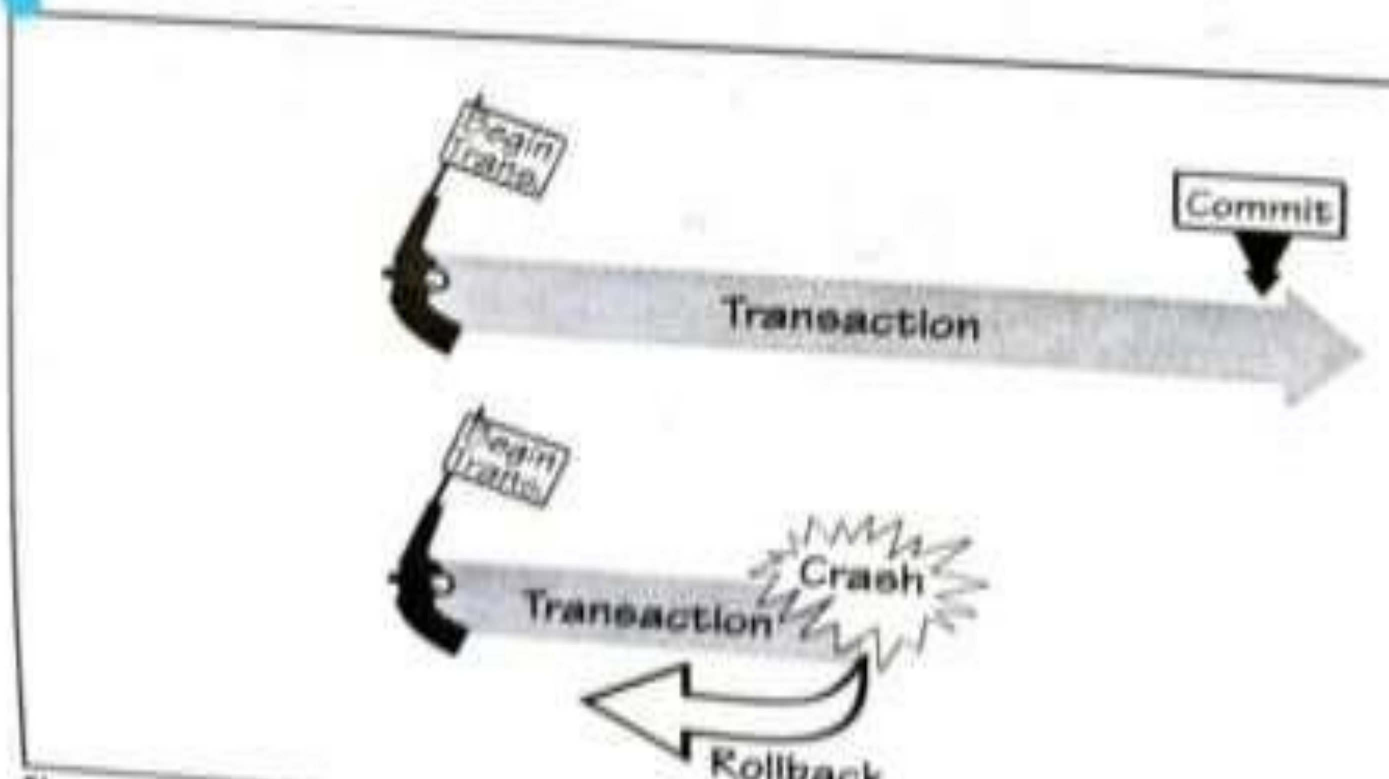


Figure 15-1. The Flat Transaction: An All-or-Nothing Proposition

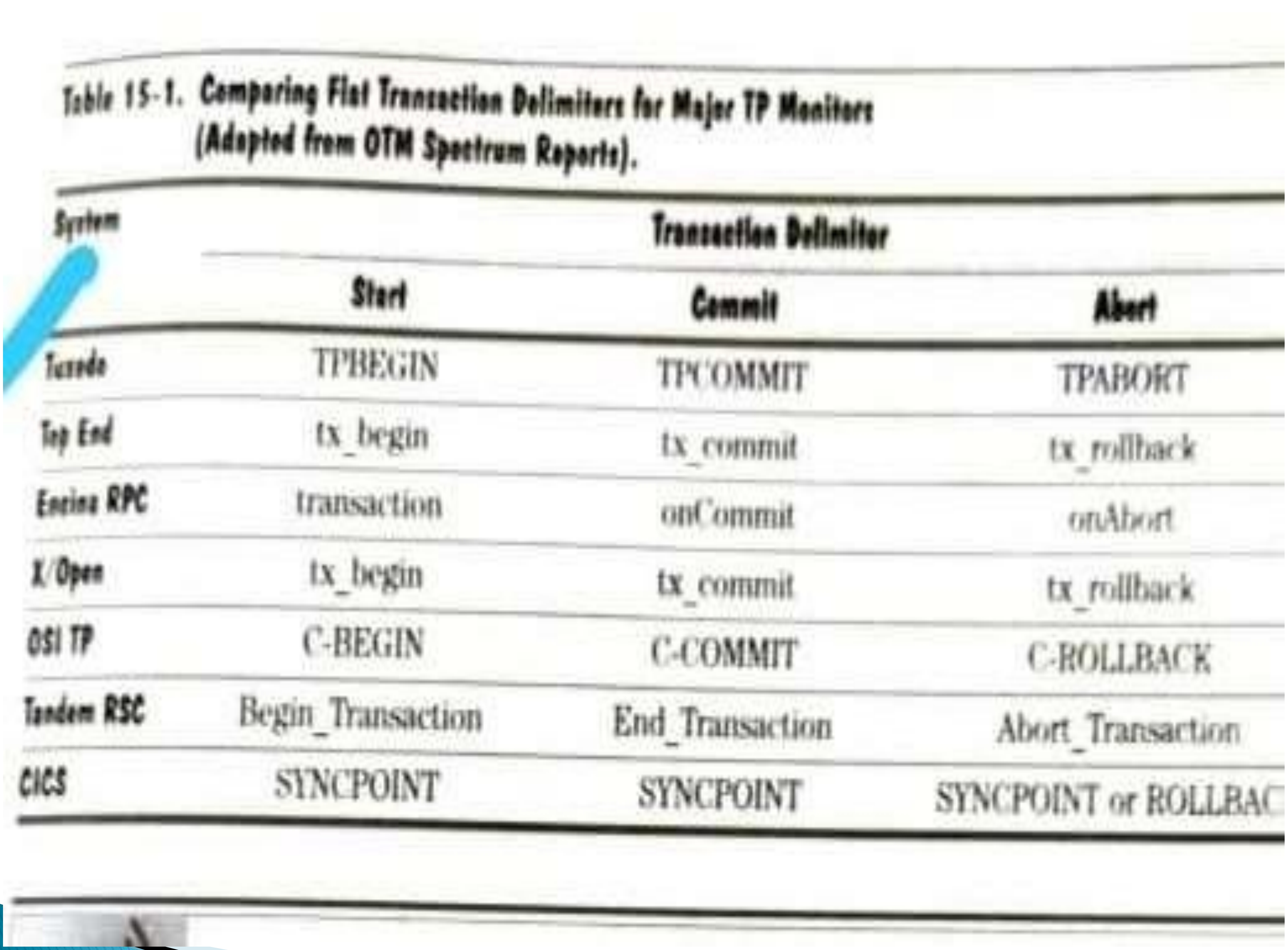


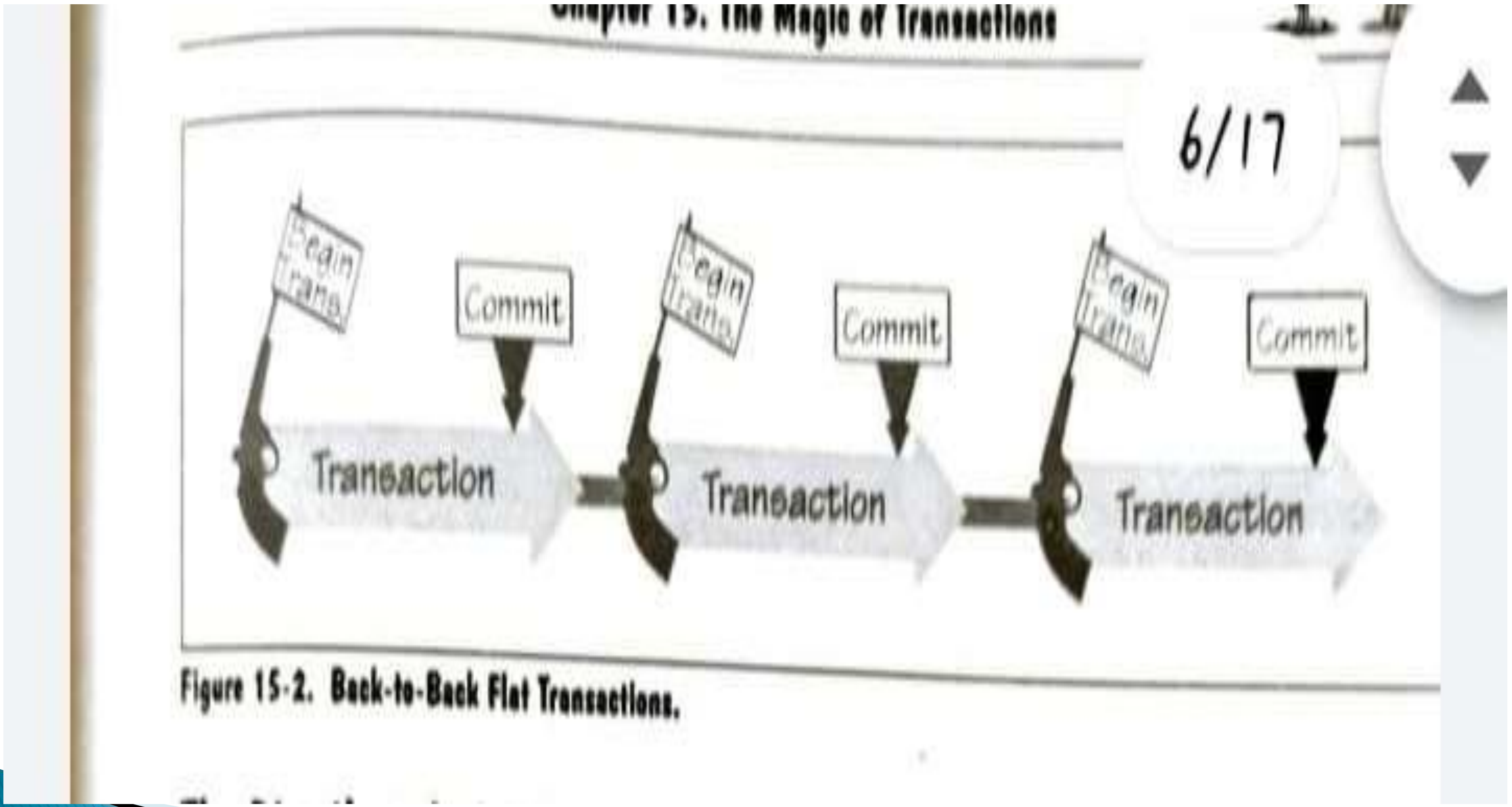
Table 15-1. Comparing Flat Transaction Delimiters for Major TP Monitors (Adapted from OTM Spectrum Reports).

System	Transaction Delimiter		
	Start	Commit	Abort
Tuxedo	TPBEGIN	TPCOMMIT	TPABORT
Top End	tx_begin	tx_commit	tx_rollback
Encina RPC	transaction	onCommit	onAbort
X/Open	tx_begin	tx_commit	tx_rollback
OSI TP	C-BEGIN	C-COMMIT	C-ROLLBACK
Tandem RSC	Begin_Transaction	End_Transaction	Abort_Transaction
CICS	SYNCPOINT	SYNCPOINT	SYNCPOINT or ROLLBAC

Baby Stepping With Flat Transactions

- A typical flat transaction does not last more than two or three seconds to avoid monopolizing critical system resources such as database locks.
- As a result, OLTP client/server programs are divided into short transactions that execute back-to-back to produce results. (Shown in diagram)
- We call this effect transaction baby step-ping or getting work done by moving in "baby steps" from one stable state to the next.

Back-to-Back Flat Transaction


Begin Trans.
Commit
Transaction
Begin Trans.
Commit
Transaction
Begin Trans.
Commit
Transaction
6/17

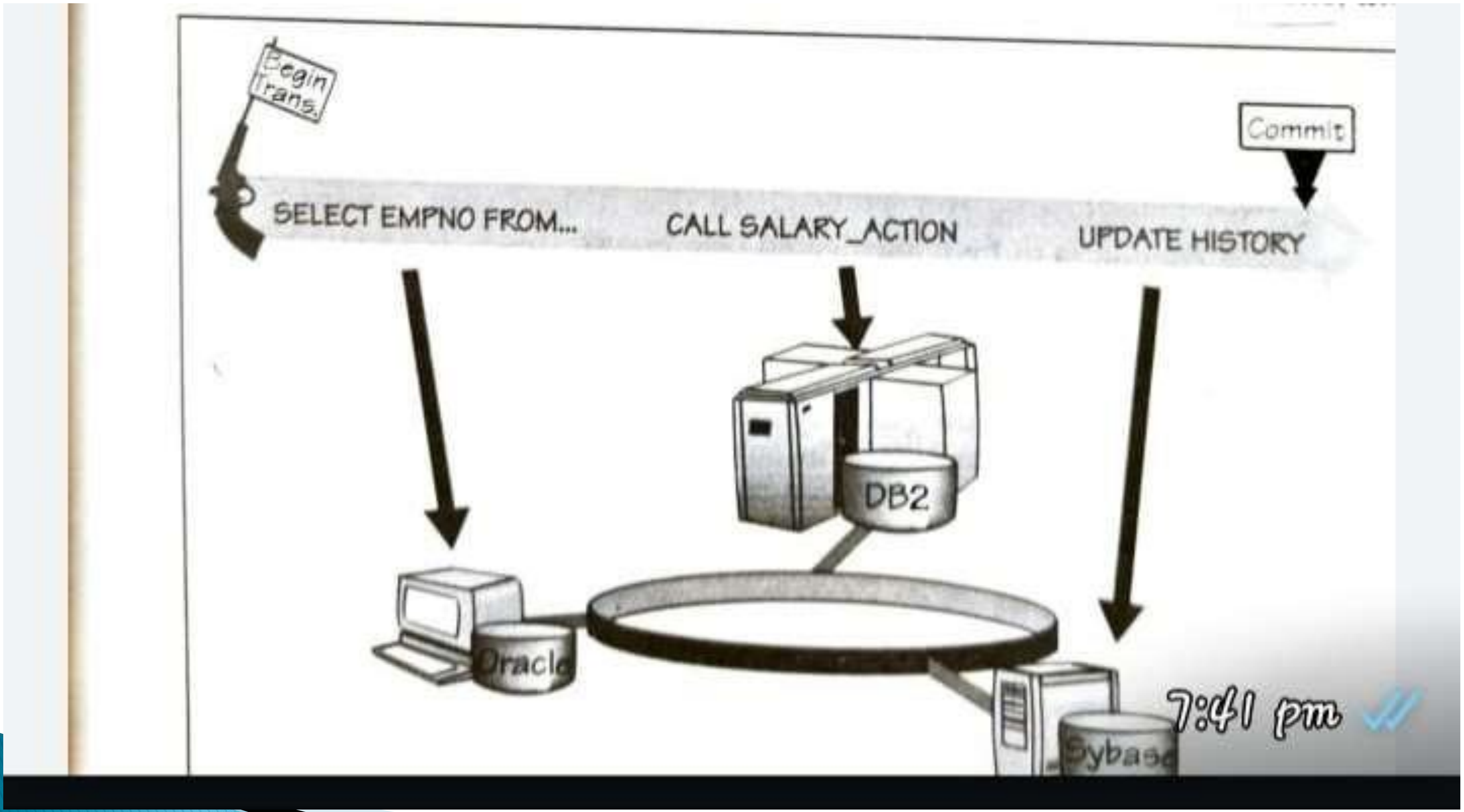
Chapter 15. The Magic of Transactions

Figure 15-2. Back-to-Back Flat Transactions.

The Distributed Flat Transaction

- Even though a high level of parallelism may be involved, as far as the programmer is concerned, it is still just a flat transaction. (Shown in diagram)
- The programmer is not aware of the considerable amount of "under-the-cover" activity that is required to make the multisite transaction appear flat.
- The transaction must travel across multiple sites to get to the resources it needs. Each site is TP Monitor must manage its local piece of the transaction.

Multisite Distributed Flat Transaction



- Within a site, the TP Monitor coordinates the transactions with the local ACID subsystems and source managers, including database managers, queue managers, persistent objects and message transports.
- For example, the TP Monitor will ensure that when database gets updated, a message gets delivered and an entry is made in a workflow queue.
- Either all of these actions will occur (exactly once) or none. In addition, one of the TP Monitors must coordinate the actions of all its fellow monitors.

- This is all done using a two-phase commit protocol, which coordinates transaction is commit or abort across multiple sites.

- **<u>Limitations of the Flat Transaction</u>**
- The following are examples of business transactions that require a more flexible approach :
- i)*Compound business transactions that need to be partially rolled back:*
- The classic example is a complex trip that includes travel arrangements hotel reservations, and a car rental.

- What happens if we simple want to cancel the car reservation but preserve the rest of the reservations. We can't do that within a flat transaction and the entire reservation is rolled back.
- This means we must give up the hotel and plane reservations just to get rid (avoid) of the car. The hotel/car reservation problem is used to justify the need for nested or chained transactions.
- We use multiple flat transactions to simulate the compound one.

- ***ii) Business transactions with humans in the loop:***
- **This is a classic GUI client/server transaction where a set of choices are presented to the user on a screen, and the server must wait for the decision. In the meantime, locks are held for the records on the screen.**
- **What happens if an operator that's viewing some airline seats decides to go to lunch? How long are the seats locked out?**
- **If it is executed as a single flat transaction, the seats will be held as long as that user is thinking or eating. Nobody else can get to those seats.**
- **This is obviously not a very good way to run a business.**

- The solution is to split the reservation into two transactions: a query transaction that displays the available seats, and a reservation transaction that performs the actual reservation.
- ***Iii)Business transactions that span long periods of time:***
- These are typical engineering Computer-Aided Design (CAD) transactions that may require CAD-managed components to be worked on for days and passed from engineer to engineer.

- The CAD transaction must be able to suspend itself and resume after shutdowns, preserve ongoing work across shutdowns, and know where it left off and what needs to be done next. In essence, it becomes a workflow manager.
- Flat transactions must be augmented by a workflow program to handle such long-lived work. This is an area where alternative transactional database check-in check-out transactions, replica management, versioning, and workflow look very promising.

- ***Iv)Business transactions with a lot of bulk :***
- How do we handle one million record updates under transactional control?
- Must the entire transaction be rolled back if a failure occurs after record 999,999 is updated.
- On the other hand, if we make each update a separate transaction, it is much slower and a million separate commits are required.
- ***v)Business transactions that span across companies or the Internet:***
- The problem here is a political one.

- A MOM (Message Oriented Middleware) solution allows organizations to split the unit of work into many transactions that can be executed asynchronously, processed on different machines, and coordinated by independent TP Monitors within each company.
- From a software perspective, we ended up breaking a single two-phase commit flat transaction into three independent flat transactions that execute on company A's TP Monitor, MOM, and company B 's TP Monitor.
- The MOM transaction ensures that the transaction has safely made it from company A's computer to company B's computer.

Alternatives: Chained and Nested Transactions

- Most of the proposed alternatives to the flat transaction are based on mechanisms that extend the flow of control beyond the linear unit of work.
- *Two* of the most obvious ways to extend the flow of control are by chaining units of work in linear sequences of "*mini*" transactions or by creating some kind of nested hierarchy of work called the *nested* transaction.

- ***Syncpoints, Chained Transactions and Sagas:***

- The chained transaction, as the name implies, introduces some form of linear control for sequencing through transactions.
- Syncpoints are known as savepoints. Within a flat transaction that allow periodic saves of accumulated work.
- The syncpoint lets roll back work and still maintain a live transaction. In contrast, a commit ends a transaction.
- Syncpoints also give better granularity of control over what we save and undo.

- But, the big difference is that the commit is durable while the syncpoint is volatile. If the system crashes during a transaction, all data accumulated in syncpoints is lost.
- Chained transactions are a variation of syncpoints that make the accumulated work durable. They allow us to commit work while staying within the transaction, so we do not have to give up our locks and resources.
- Sagas extend the chained transactions to let we roll back the entire chain. They do that by maintaining a chain of compensating transactions.

- We still get the crash resistance of the intermediate commits, but we have the choice of rolling back the entire chain under program control. This lets we treat the entire chain as an atomic unit of work.
- ***Nested Transactions:***

- Nested Transactions provide the ability to define transactions within other transactions. They do that by breaking a transaction into hierarchies of "subtransactions“.
- The main transaction starts the subtransactions, which behave as dependent transactions.

- A subtransaction can also start its own subtransactions, thus making the entire structure very recursive.
- Each subtransaction can issue a commit or rollback for its designated pieces of work.
- When a subtransaction commits, its results are accessible only to the parent that spawned it. A subtransaction's commit becomes permanent after it issues a local commit and all its ancestors commit.
- If a parent transaction does a rollback, all its descendent transactions are rolled back, regardless of whether they issued local commits.

- The main benefit of nesting is that a failure in a subtransaction can be trapped and retried using an alternative method, which still allows the main transaction to succeed.
- Nesting helps programmers write granular transactions. The only commercial implementation of nested transactions we know of is the “Encina TP Monitor”.
- Encina's Transactional C or C+ + allows us to declare the nesting directly in our code where it starts resembling regular procedure invocations.