Chapter 5

ADVANCED DATABASE MODELS, SYSTEMS, AND APPLICATIONS

Chapter Outline

After comprehensive study of this chapter, you will be able to:

- Active Database Concepts and Triggers
- Temporal Database Concepts
- Spatial Database Concepts
- Multimedia Database Concepts
- Deductive Database Concepts
- Introduction to Information Retrieval and Web Search

ACTIVE DATABASE CONCEPTS AND TRIGGERS

A trigger is a procedure which is automatically invoked by the DBMS in response to changes to the database, and is specified by the database administrator (DBA). A database with a set of associated triggers is generally called an active database. These databases are very difficult to be maintained because of the complexity that arises in understanding the effect of these triggers. In such database, DBMS initially verifies whether the particular trigger specified in the statement that modifies the database) is activated or not, prior to executing the statement. If the trigger is active then DBMS executes the condition part and then executes the action part only if the specified condition is evaluated to true. It is possible to activate more than one trigger within a single statement.

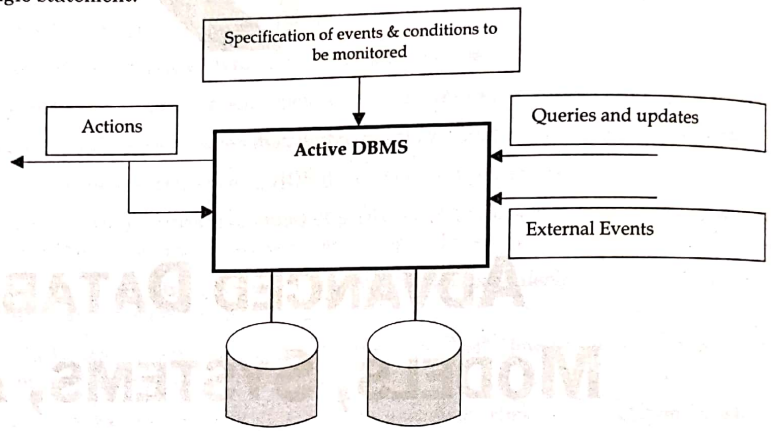


Figure 5.1: Active database architecture

Triggers are executed when a specified condition occurs during insert/delete/update. Triggers are action that fire automatically based on these conditions. Triggers follow an Event-condition-action (ECA) model.

- **Event**: An event is a change to the database which activates the trigger. It is database modification E.g., insert, delete, and update.
- **Condition:** A query that is run when the trigger is activated is called as a condition. It is any true/false expression. If no condition is specified then condition is always true.
- **Action:** A procedure which is executed when the trigger is activated and its condition is true. It is the sequence of SQL statements that will be automatically executed.

The syntax to create database trigger is as follows:

```
CREATE [OR REPLACE] TRIGGER <Trigger_name>
[BEFORE | AFTER | INSTEAD OF]
[DELETE | INSERT | UPDATE .....ON]<Name of underlying object>
```

```
[FOR EACH ROW]
[WHEN<condition for trigger to get execute>]
DECLARE
<Declaration Part>
BEGIN
<Execution Part>
EXCEPTION
<Exception handling part>
END;
```

- **Create or Replace:** The CREATE keyword specifies that we are creating a new trigger. The OR REPLACE keywords are optional. They are used to modify an existing trigger.
- **Trigger name:** Specify the name of the trigger that we want to create after the CREATE OR REPLACE keywords.
- **BEFORE | AFTER:** The BEFORE or AFTER option specifies when the trigger fires, either before or after a triggering event e.g., INSERT, UPDATE, or DELETE.
- **ON Table_name:** The Table_name is the name of the table associated with the trigger.
- **FOR EACH ROW:** The clause FOR EACH ROW specifies that the trigger is a row-level trigger. A row-level trigger fires once for each row inserted, updated, or deleted. Besides the row-level triggers, we have statement-level triggers. A statement-trigger fire once regardless of the number of rows affected by the triggering event. If you omit the FOR EACH ROW clause, the CREATE TRIGGER statement will create a statement-level trigger.
- **WHEN (condition):** This provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.
- The declaration part, execution part, exception handling part is same as that of the other PL/SQL blocks. Declaration part and exception handling part are optional.

:NEW and :OLD Clause

In a row level trigger, the trigger fires for each related row. And sometimes it is required to know the value before and after the DML statement. Oracle has provided two clauses in the RECORD-level trigger to hold these values. We can use these clauses to refer to the old and new values inside the trigger body.

a. **:NEW** – It holds a new value for the columns of the base table/view during the trigger execution
b. **:OLD** – It holds old value of the columns of the base table/view during the trigger execution

This clause should be used based on the DML event. Below table will specify which clause is valid for which DML statement (INSERT/UPDATE/DELETE).

	INSERT	UPDATE	DELETE
:NEW	VALID	VALID	INVALID. There is no new value in delete case.
:OLD	INVALID. There is no old value in insert case	VALID	VALID

Features of Active Database

- It possesses all the concepts of a conventional database i.e. data modelling facilities, query language, multi-user access, recovery, etc.
- It supports all the functions of a traditional database, including data definition, data manipulation, storage management, transaction management, concurrency control, and crash recovery.
- An active database supports definition and management of ECA-rules.
- An active database must detect event occurrences.
- An active database must be able to evaluate conditions and to execute actions.
- An active database includes an event driven architecture (often in the form of ECA rules)
- An active database must be able to evaluate conditions and to execute actions. This requirement means that it has to implement rule execution.

Types of Triggers in Oracle

Triggers can be classified based on the following parameters.

1. **Classification based on the level**
   - STATEMENT level Trigger: It fires one time for the specified event statement.
   - ROW level Trigger: It fires for each record that got affected in the specified event. (only for DML)
2. **Classification based on the timing**
   - BEFORE Trigger: It fires before the specified event has occurred.
   - AFTER Trigger: It fires after the specified event has occurred.
   - INSTEAD OF Trigger: A special type. You will learn more about the further topics. (only for DML)
3. **Classification based on the Event**
   - DML Trigger: It fires when the DML event is specified (INSERT/UPDATE/DELETE)
   - DDL Trigger: It fires when the DDL event is specified (CREATE/ALTER)
   - DATABASE Trigger: It fires when the database event is specified (LOGON/LOGOFF/STARTUP/SHUTDOWN)

ROW level Trigger

A row level trigger is fired each time the table is affected by the triggering statement. For example, if an UPDATE statement updates multiple rows of a table, a row trigger is fired once for each row affected by the UPDATE statement. If a triggering statement affects no rows, a row

trigger is not run. Row triggers are useful if the code in the trigger action depends on data provided by the triggering statement or rows that are affected. Following example illustrates a row trigger that uses the values of each row affected by the triggering statement.

Example: Let us consider a table which is named Student.

Student (Sid, Sname, age, salary)

Student

Sid	Sname	Age	Salary
1	Rajan	33	20000
2	Aarav	15	30000
3	Kamala	30	25000
4	Janaki	43	60000

Let us now create a row-level trigger for the Student table that would get executed by the DML statement like UPDATE, INSERT or DELETE on that table. The trigger will compute and show the age difference between current and previous values.

```
CREATE OR REPLACE TRIGGER Age_changes
BEFORE DELETE OR INSERT OR UPDATE ON Student
FOR EACH ROW
WHEN (NEW.Sid> 0)
DECLARE
age_diffNUMBER;
BEGIN
age_diff := :NEW.age  - :OLD.age;
dbms_output.put_line ('Previous age: ' | | :OLD.age);
dbms_output.put_line ('Current age: ' | | :NEW.age);
dbms_output.put_line ('Age difference: ' | | age_diff);
END;
```

To fire the above trigger, we need to do any DML operation like DELETE, INSERT or UPDATE on the table. Let us again insert some values in the Student table with the help of the below query:

```
INSERT INTO Student
VALUES (6, 'Aadesh', 16, 99);
```

Once the INSERT operation is completed in the Student table, the trigger age_changes gets executed.

The output of the query code:

```
Previous age:
Current age: 16
Age difference:
```

Since a new record is created and the previous age is unavailable, the previous age and Age difference computation comes as null in the above output.

Now, let us modify a record with UPDATE statement with the help of the below query:

```
UPDATE student
SET age = age + 5
WHERE Sid = 4;
```

Once the UPDATE operation is completed in the Student table, the trigger age_changes gets executed.

The output of the above query:

```
Previous age: 43
Current age: 48
Age difference: 5
```

Statement level Trigger

A statement trigger is fired once on behalf of the triggering statement, regardless of the number of rows in the table that the triggering statement affects, even if no rows are affected. For example, if a DELETE statement deletes several rows from a table, a statement-level DELETE trigger is fired only once. Statement triggers are useful if the code in the trigger action does not depend on the data provided by the triggering statement or the rows affected. For example, use a statement trigger to:

- Make a complex security check on the current time or user
- Generate a single audit record

Difference between Row level and Statement level triggers

Triggers are defined as stored programs which are automatically executed whenever some events such as CREATE, ALTER, UPDATE, INSERT, DELETE takes place. They can be defined on a database, table, view with which event is associated. Triggers can be broadly classified into Row Level and Statement Level triggers. Broadly, these can be differentiated as:

Row Level Triggers	Statement Level Triggers
Row level triggers executes once for each and every row in the transaction.	Statement level triggers executes only once for each single transaction.
Specifically used for data auditing purpose.	Used for enforcing all additional security on the transactions performed on the table.
"FOR EACH ROW" clause is present in CREATE TRIGGER command.	"FOR EACH ROW" clause is omitted in CREATE TRIGGER command.
Example: If 1500 rows are to be inserted into a table, the row level trigger would execute 1500 times.	**Example**: If 1500 rows are to be inserted into a table, the statement level trigger would execute only one time.

Applications of Active database

- Data monitoring activities such as CIM, Telecommunications Network Management, Program trading, Medical and Financial Decision Support Systems can greatly benefit from integration with active database.

- Production control, e.g., power plants.
- Maintenance tasks, e.g., inventory control.
- Financial applications, e.g., stock & bond trading.
- Telecommunication and network management.
- Air traffic control.
- Computer Integrated Manufacturing (CIM)
- Statistics gathering and authorization tools.

Weaknesses of active database

- Insufficient methodological support in design and analysis.
- Lack of standardization.
- Missing development and administration tools for triggers.
- Weak performance: This is one the main reasons that makes users reluctant to use active rules in the development of large applications.
- Optimizing large applications is rendered difficult by the separation of transactions and triggers and the misunderstanding of their subtle interactions.
- Lack of Support for application development in many active database management system prototypes.
- Distribution and Parallelism has not been widely treated as active databases have been considered primarily in centralized database environments.
- The layered approach is beneficial in terms of construction cost, but the entire system cannot be optimized comprehensively and this can cause runtime performance to be worse than in an integrated architecture which is also very costly to construct.

Temporal Database Concepts

A temporal database stores data relating to time instances. It offers temporal data types and stores information relating to past, present and future time.

A Temporal Database is a database with built-in support for handling time sensitive data. Usually, databases store information only about current state, and not about past states. For example in an employee database if the address or salary of a particular person changes, the database gets updated, the old value is no longer there. However for many applications, it is important to maintain the past or historical values and the time at which the data was updated. That is, the knowledge of evolution is required. That is where temporal databases are useful. It stores information about the past, present and future. Any data that is time dependent is called the temporal data and these are stored in temporal databases.

Temporal Databases store information about states of the real world across time. Temporal Database is a database with built-in support for handling data involving time. It stores information relating to past, present and future time of all events.

Examples of Temporal Databases

- **Healthcare Systems:** Doctors need the patients' health history for proper diagnosis. Information like the time a vaccination was given or the exact time when fever goes high etc.
- **Insurance Systems:** Information about claims, accident history, time when policies are in effect needs to be maintained.
- **Reservation Systems:** Date and time of all reservations is important.
- **Finance:** stock price histories need to be maintained.
- **Personnel management:** salary and position history need to be maintained
- **Banking:** credit histories

There are following three types of time available in temporal database:

- Valid time
- Transaction time and
- Bitemporal time

Valid time

Valid time is a time period during which a fact is true in the real word. Given a particular event or fact that is associated with a particular time point or time period in the database, the association may be interpreted to mean different things. The most natural interpretation is that the associated time is the time that the event occurred, or the period during which the fact was considered to be true in the real world. If this interpretation is used, the associated time is often referred to as the **valid time.** A temporal database using this interpretation is called a **valid time database.**

For example, in a company the Salary of the employees have a valid time and end time

Valid Time Table

Emp_no	Name	Salary	Valid Start Time	Valid End Time
E1	Indra	80000	2072-03-30	2073-03-30
E1	Indra	90000	2073-04-01	2074-03-30
E1	Indra	95000	2074-04-01	2075-03-30
E1	Indra	100000	2075-04-01	2077-03-30
E1	Indra	150000	2077-04-01	2078-03-30
E2	Bhupi	55000	2072-03-30	2076-03-30
E2	Bhupi	75000	2076-04-01	Now
E3	Rajan	40000	2072-03-30	2078-03-30

Transaction time

Transaction time is the time period during which a fact stored in the database was known.

However, a different interpretation can be used, where the associated time refers to the time when the information was actually stored in the database; that is, it is the value of the system time clock when the information is valid in the system. In this case, the associated time is called

the **transaction time**. A temporal database using this interpretation is called a **transaction time database**. Unlike valid time here we can rollback database. For example, in a company the Salary of the employees have a valid time and end time

Transaction Time Table

Emp_no	Name	Salary	Transaction Start Time	Transaction End Time
E1	Indra	80000	2072-03-30, 10:02:33	2073-03-30, 10:02:33
E1	Indra	90000	2073-04-01, 11:13:30	2074-03-30, 07:08:38
E1	Indra	95000	2074-04-01, 10:22:13	2075-03-30, 10:05:53
E1	Indra	100000	2075-04-01, 10:33:23	2077-03-30, 10:04:43
E1	Indra	150000	2077-04-01, 10:30:35	2078-03-30, 10:22:36
E2	Bhupi	55000	2072-03-30, 10:01:43	2076-03-30, 11:12:23
E2	Bhupi	75000	2076-04-01, 10:04:53	Now
E3	Rajan	40000	2072-03-30, 10:04:23	2078-03-30, 09:48:33

Bitemporal data

It combines both valid ad transaction time. It stores data with respect of both valid time and transaction time. In some applications, only one of the dimensions is needed and in other cases both time dimensions are required, in which case the temporal database is called a bitemporal database.

Bitemporal Time Table

Emp_no	Name	Salary	Transaction Time	Valid Start Time	Valid End Time
E1	Indra	80000	2078-03-30, 10:02:33	2072-03-30	2073-03-30
E1	Indra	90000	2078-03-30, 10:02:33	2073-04-01	2074-03-30
E1	Indra	95000	2078-03-30, 10:02:33	2074-04-01	2075-03-30
E1	Indra	100000	2078-03-30, 10:02:33	2075-04-01	2077-03-30
E1	Indra	150000	2078-03-30, 10:02:33	2077-04-01	2077-08-30
E2	Bhupi	55000	2078-03-30, 10:02:33	2072-03-30	2076-03-30
E2	Bhupi	75000	2078-03-30, 10:02:33	2076-04-01	Now
E3	Rajan	40000	2078-03-30, 10:02:33	2072-03-30	2077-03-30

Spatial Database Concepts

A spatial database is a database that is enhanced to store and access spatial data or data that defines a geometric space. These data are often associated with geographic locations and features, or constructed features like cities. Data on spatial databases are stored as coordinates, points, lines, polygons and topology. Some spatial databases handle more complex data like three-dimensional objects, topological coverage and linear networks.

Spatial databases address many of the limitations of static data files. Spatial databases can contain large amounts of data in multiple tables with linking mechanisms that maintain data integrity. They can enforce restrictions on data entry to limit collection of inconsistent data. As they grow, they can span multiple physical machines as well as maintain copies of themselves for redundancy. Spatial databases can also maintain changes to a set of spatial data and track which users are making edits for approval and auditing.

It is important to make the distinction between typical databases and spatial databases. Spatial databases are standard databases that have been extended to accept spatial data types and queries. Spatial data types store feature geometry that describes shape and location. The geometry of spatial features is compressed and stored in a binary field along with the attribute data that describe the feature. In addition, the database application code is extended so that typical queries using alphanumeric characters and logical operators are extended to take advantage of location, proximity, and topology. For instance, in a typical customer database a company might query for all customers whose last name begins with a certain letter. In a spatial database, they can query for all customers within proximity of a particular store or find clusters of customers. These types of spatial transactions are not available in typical databases.

Examples of non-spatial data

The spatial data represent a geographical space. They are characterised by the point, lines and the polygons. The point represents positional characteristics of the some of the geographical features such as school, hospitals, wells, villages, towns etc. on the map.

Some examples of non-spatial data are:

- Names, phone numbers, email addresses of people etc.

Examples of spatial data

The data describing the spatial data is called as non-spatial or attribute data. For example if you have a map showing position of your school you can attach the information as the name of the school, number of students in each of the class, available facilities like library, Labs, Canteen, Playground etc. In another word you will be defining the attributes of the spatial data. Thus non-spatial data are also known as attribute data. Some examples of spatial data are:

- Weather and climate data
- Rivers, Farms, ecological impact
- Medical imaging
- NASA satellites imagery-terabytes of data per day

(a) Hospitalization

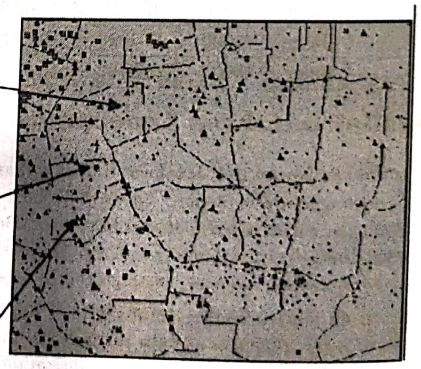
Gid	Disease	Age	Gender	The Geom
1	Cancer	23	M	Point . . .
2	Infarct	25	M	Point . . .
. . .	. . .	. . .	. . .	. . .

(b) Factory

Gid	Name	Address	Business	The geom.
1	Gerdau	Xxxxxx	Siderurgical	Point . . .
2	Infarct	Zzzzzz	Metallurgical	Point . . .
. . .	. . .	. . .	. . .	. . .

(c) Cellular Antenna

Gid	Owenber	Power level	Date	The geom.
1	Vivo	2	03/01/1999	Point . . .
2	Vivo	1a	10.101/1995	Point . . .
. . .	. . .	. . .	. . .	. . .



(d) GEOMETRY_COLUMNS

F table schema	F table name	F geometry column	.....	SRID
Public	Hospitalization	The Geom	.....	– 1
Public	Factory	The_Geom	.....	– 1
Public	Cellular Antenna	The_Geom	.....	– 1

Figure: Spatial Database

Multimedia Database Concepts

Multimedia database is the collection of interrelated multimedia data that includes text, graphics (sketches, drawings), images, animations, video, audio etc. and have vast amounts of multisource multimedia data. The framework that manages different types of multimedia data which can be stored, delivered and utilized in different ways is known as multimedia database management system.

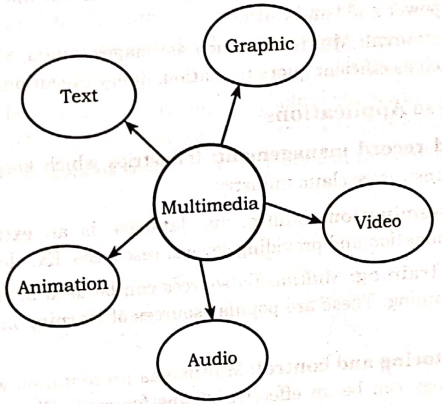


Figure5.2: Components of multimedia database

Multimedia databases provide features that allow users to store and query different types of multimedia information, which includes images (such as photos or drawings), video clips (such as movies, newsreels, or home videos), audio clips (such as songs, phone messages, or speeches), and documents (such as books or articles). The main types of database queries that are needed involve locating multimedia sources that contain certain objects of interest. For example, one may want to locate all video clips in a video database that include a certain person, say Michael Jackson. One may also want to retrieve video clips based on certain activities included in them, such as video clips where a soccer goal is scored by a certain player or team. The contents of a multimedia database management system can be:

- **Media data:** It is the actual data which represents an object.
- **Media format data:** The information such as resolution, sampling rate, encoding system, etc. about the format of the media data under consideration after is undergoes acquisition, processing, and encoding is the media format data.

- **Media keyword data:** Media keyword data are the keyword description related to the generation of data. This data is also known as content descriptive data. Examples of content descriptive data are place, time, date of recording.
- **Media feature data:** Media feature data contains data which is content dependent such as kind of texture, distribution of, and the different shapes present in the data.

Challenges to multimedia databases

- **Modelling:** Work in this area can improve the database versus information retrieval techniques.
- **Design:** The physical, conceptual, and logical design of multimedia databases is not addressed entirely leading to performance and tuning issues.
- **Storage:** The storage of databases on a standard disc can lead to problems like representation, mapping to disc hierarchies, compression, etc.
- **Performance:** Audio-video synchronization and audio playback applications are where physical limitations dominate. Parallel processing can reduce these problems, but these techniques have not been completely developed yet. Multimedia databases also consume a lot of processing power and bandwidth.
- **Queries and Retrieval:** Multimedia such as images, audio, video lead to retrieval and queries issues such as efficient query formation, query execution, etc.

Multimedia Database Applications

- **Documents and record management:** Industries which keep a lot of documentation and records. Ex: Insurance claim industry.
- **Knowledge dissemination:** Multimedia database is an extremely efficient tool for knowledge dissemination and providing several resources. Ex: electronic books
- **Education and training:** Multimedia sources can be used to create resources useful in education and training. These are popular sources of learning in recent days. Ex: Digital libraries.
- **Real-time monitoring and control:** Multimedia presentation when coupled with active database technology can be an effective means for controlling and monitoring complex tasks. Ex: Manufacture control
  - Marketing
  - Advertisement
  - Retailing
  - Entertainment
  - Travel

Deductive Database Concepts

A deductive database is a database system that can make deductions (i.e. conclude additional facts) based on rules and facts stored in the (deductive) database. Datalog is the language typically used to specify facts, rules and queries in deductive databases. A deductive database can be defined as an advanced database augmented with an inference system.

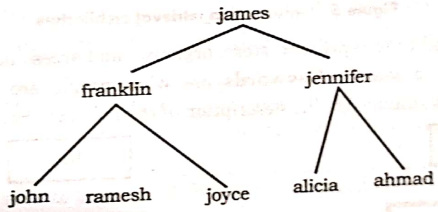
Database + Inference = Deductive database

By evaluating rules against facts, new facts can be derived, which in turn can be used to answer queries. It makes a database system more powerful.

In a deductive database system we typically specify rules through a declarative language—a language in which we specify what to achieve rather than how to achieve it. An inference engine (or deduction mechanism) within the system can deduce new facts from the database by interpreting these rules. The model used for deductive databases is closely related to the relational data model, and particularly to the domain relational calculus formalism. It is also related to the field of logic programming and the Prolog language. The deductive database work based on logic has used Prolog as a starting point. A variation of Prolog called Datalog is used to define rules declaratively in conjunction with an existing set of relations, which are themselves treated as literals in the language. Although the language structure of Datalog resembles that of Prolog, its operational semantics—that is, how a Datalog program is executed—is still different.

A deductive database uses two main types of specifications: facts and rules. **Facts** are specified in a manner similar to the way relations are specified, except that it is not necessary to include the attribute names. Recall that a tuple in a relation describes some real-world fact whose meaning is partly determined by the attribute names. In a deductive database, the meaning of an attribute value in a tuple is determined solely by its position within the tuple. **Rules** are somewhat similar to relational views. They specify virtual relations that are not actually stored but that can be formed from the facts by applying inference mechanisms based on the rule specifications. The main difference between rules and views is that rules may involve recursion and hence may yield virtual relations that cannot be defined in terms of basic relational views.

Example:



Facts

Supervise (Frankin, john)
Supervise (Frankin, Ramesh)
Supervise (Frankin, Joyce)
Supervise (James, Frankin)
Supervise (Jennifer, Alicia)
Supervise (Jennifer, Ahmad)
Supervise (James, Jennifer)

Rules

Superior(X, Y) ← supervise (X, Y),
Superior(X, Y) ← supervise (X, Z), superior (Z, Y),
Subordinary(X, Y) ← superior (Y, X),

Introduction to Information Retrieval and Web Search

Information retrieval is the process of retrieving documents from a collection inresponse to a query (or a search request) by a user.

Information Retrieval is the activity of obtaining material that can usually be documented on an unstructured nature i.e. usually text which satisfies an information need from within large collections which is stored on computers. For example, Information Retrieval can be when a user enters a query into the system.Nowadays hundreds of millions of people engage in IR every day when they use web search engines. Information Retrieval is believed to be the dominant form of Information access. The IR system assists the users in finding the information they require but it does not explicitly return the answers to the question. It notifies regarding the existence and location of documents that might consist of the required information. Information retrieval also extends support to users in browsing or filtering document collection or processing a set of retrieved documents.

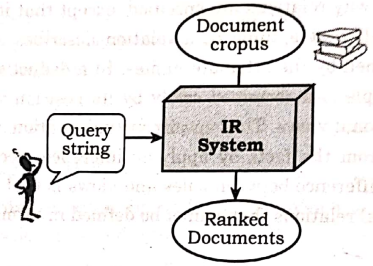


Figure 5.3: Information retrieval architecture

An IR system has the ability to represent, store, organize, and access information items. A set of keywords are required to search. **Keywords** are what people are searching for in search engines. These keywords summarize the description of the information.

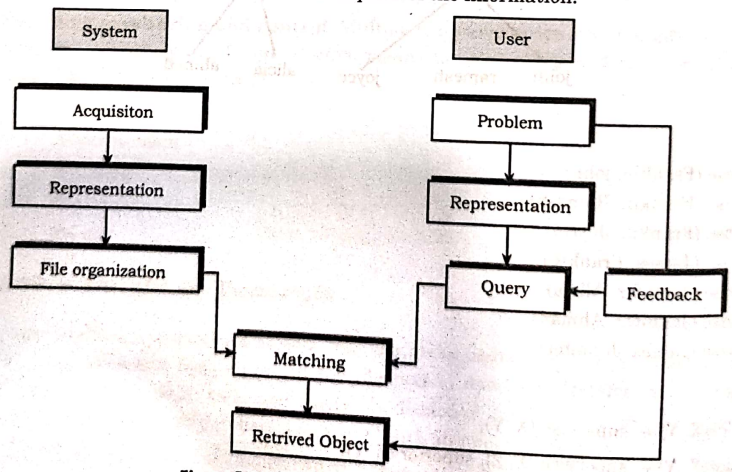
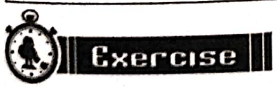


Figure 5.4: flow chart of information retrieval

- **Acquisition:** In this step, the selection of documents and other objects from various web resources that consist of text-based documents takes place. The required data is collected by web crawlers and stored in the database.
- **Representation:** It consists of indexing that contains free-text terms, controlled vocabulary, manual & automatic techniques as well. Example: Abstracting contains summarizing and Bibliographic description that contains author, title, sources, data, and metadata.
- **File Organization:** There are two types of file organization methods. I.e. **Sequential:** It contains documents by document data. **Inverted:** It contains term by term, list of records under each term. Combination of both.
- **Query:** An IR process starts when a user enters a query into the system. Queries are formal statements of information needs, for example, search strings in web search engines. In information retrieval, a query does not uniquely identify a single object in the collection. Instead, several objects may match the query, perhaps with different degrees of relevancy.

Difference between Information Retrieval and Data Retrieval

Information Retrieval	Data Retrieval
The software that deals with the organization, storage, retrieval, and evaluation of information from document repositories particularly textual information is called information retrieval.	Data retrieval deals with obtaining data from a database management system such as ODBMS. It is a process of identifying and retrieving the data from the database, based on the query provided by user or application.
Retrieves information about a subject.	Determines the keywords in the user query and retrieves the data.
Small errors are likely to go unnoticed.	A single error object means total failure.
Not always well structured and is semantically ambiguous.	Has a well-defined structure and semantics.
Does not provide a solution to the user of the database system.	Provides solutions to the user of the database system.
The results obtained are approximate matches.	The results obtained are exact matches.
Results are ordered by relevance.	Results are unordered by relevance.
It is a probabilistic model.	It is a deterministic model.



1. What are the uses of active database? Discuss active database with example.
2. Discuss some applications of active database. How do spatial database differ from regular database?
3. Why do we need temporal database? Discuss valid time, transaction time and bi-temporal relations.

4. What is active database? Discuss generalized mode for active databases?
5. Discuss how time is represented in temporal databases and compare the different time dimensions.
6. What is temporal database? Explain the terms date, time, timestamp, interval and period.
7. Define concept behind trigger. How statement level trigger differ from row level trigger? Explain.
8. Define deductive database. Write down their applications and importance.
9. What do you mean by information retrieval? Explain concept of IR in web search.
10. What is the application of temporal database? Describe different types of time occur in temporal database.
11. Define multimedia database. Describe challenges on multimedia database. Also describe their application areas.
12. What is multimedia database? What are the nature of multimedia database and applications?
13. Define multimedia database. Discuss benefits of multimedia databases. How do you query image database?
14. What are the uses of active database? Discuss active database with example.
15. What is structured data and what is unstructured data? Give an example of each from your experience.
16. Give a general definition of information retrieval (IR). What does information retrieval involve when we consider information on the Web?
17. Explain the main differences between the database and IR systems.
18. What are the differences between row-level and statement-level active rules?
19. What are deductive databases? What are the differences among valid time, transaction time, and bi-temporal relations?
20. Describe how the insert, delete, and update commands should be implemented on a valid time relation.

□□□