CS9152 – DATABASE TECHNOLOGY

UNIT – V

CURRENT ISSUES

TEXT BOOK

1. Elisa Bertino, Barbara Catania, Gian Piero Zarri, "Intelligent Database Systems", Addison-Wesley, 2001.

REFERENCES

1. Carlo Zaniolo, Stefano Ceri, Christos Faloustsos, R.T.Snodgrass, V.S.Subrahmanian, "Advanced Database Systems", Morgan Kaufman, 1997.
2. N.Tamer Ozsu, Patrick Valduriez, "Principles of Distributed Database Systems", Prentice Hal International Inc. , 1999.
3. C.S.R Prabhu, "Object-Oriented Database Systems", Prentice Hall Of India, 1998.
4. Abdullah Uz Tansel Et Al, "Temporal Databases: Theory, Design And Principles",Benjamin Cummings Publishers , 1993.
5. Raghu Ramakrishnan, Johannes Gehrke, "Database Management Systems", Mcgraw Hill, Third Edition, 2004.
6. Henry F Korth, Abraham Silberschatz, S. Sudharshan, "Database System Concepts", Fourth Ediion, McGraw Hill , 2002.
7. R. Elmasri, S.B. Navathe, "Fundamentals of Database Systems", Pearson Education, 2004.

CURRENT ISSUES

Syllabus:

UNIT V CURRENT ISSUES 10
Rules – Knowledge Bases – Active and Deductive Databases – Parallel Databases – Multimedia Databases – Image Databases – Text Database

Table of Contents

Topic – 1: Rules

In 1985, database pioneer Dr. E.F. Codd laid out twelve rules of relational database design. These rules provide the theoretical (although sometimes not practical) underpinnings for modern database design. The rules may be summarized as follows:

- All database management must take place using the relational database's innate functionality
- All information in the database must be stored as values in a table
- All database information must be accessible through the combination of a table name, primary key and column name.
- The database must use NULL values to indicate missing or unknown information
- The database schema must be described using the relational database syntax
- The database may support multiple languages, but it must support at least one language that provides full database functionality (e.g. SQL)
- The system must be able to update all updatable views
- The database must provide single-operation insert, update and delete functionality
- Changes to the physical structure of the database must be transparent to applications and users.
- Changes to the logical structure of the database must be transparent to applications and users.
- The database must natively support integrity constraints.
- Changes to the distribution of the database (centralized vs. distributed) must be transparent to applications and users.
- Any languages supported by the database must not be able to subvert integrity controls

Topic – 2: Knowledge Bases

A **knowledge base** (or **knowledgebase**; abbreviated **KB**, **kb** or Δ) is a special kind of database for knowledge management. It provides the means for the computerized collection, organization, and retrieval of knowledge.

Types:

Knowledge bases are categorized into two major types:

• ***Machine-readable knowledge bases*** store knowledge in a computer-readable form, usually for the purpose of having automated deductive reasoning applied to them.

They contain a set of data, often in the form of **rules** that describe the knowledge in a logically consistent manner. Logical operators, such as *And* (conjunction), *Or* (disjunction), *material implication* and *negation* may be used to build it up from the atomic knowledge. Consequently, classical deduction can be used to reason about the knowledge in the knowledge base.

Human-readable knowledge bases are designed to allow people to retrieve and use the knowledge they contain, primarily for training purposes. They are commonly used to capture explicit knowledge of an organization, including **troubleshooting, articles, white papers, user manuals** and others. A primary benefit of such a knowledge base is that it can help a user to find an existing solution to his or her current problem (thus avoiding having to 're-invent the wheel').

• The most important aspect of a knowledge base is the quality of information it contains. The best knowledge bases have carefully written articles that are kept up to date, an excellent information retrieval system (such as a search engine), and a carefully designed content format and classification structure.

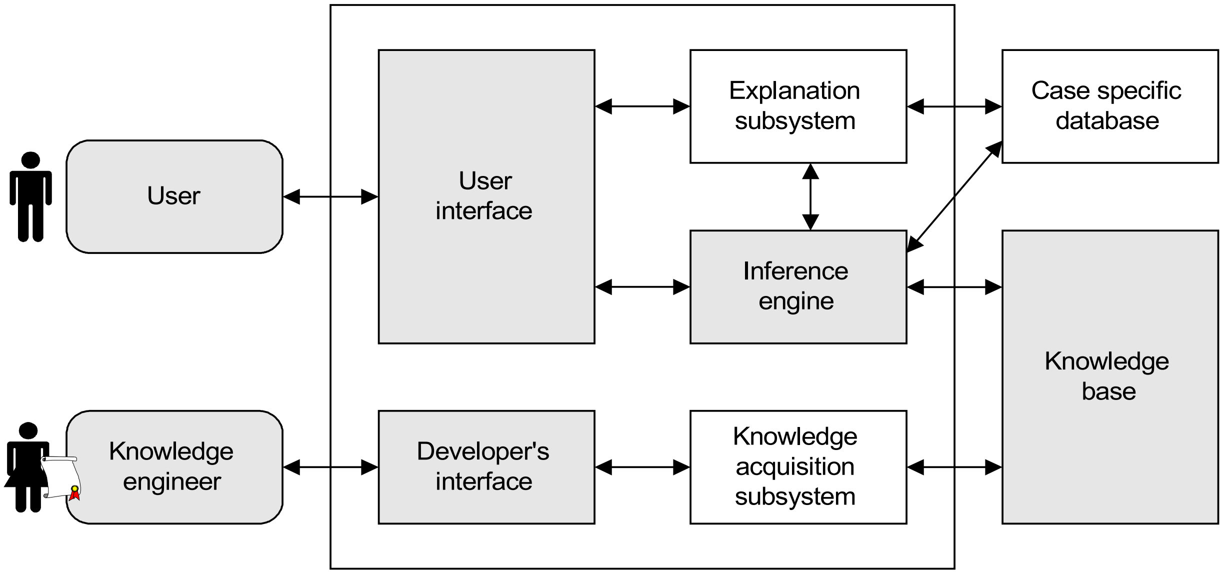
• A knowledge base may use an ontology to specify its structure (entity types and relationships) and classification scheme. An ontology, together with a set of instances of its classes, constitutes a knowledge base.

- Knowledge-based systems, expert systems
  - structure, characteristics
  - main components
  - advantages, disadvantages
- Base techniques of knowledge-based systems
  - rule-based techniques
  - inductive techniques
  - hybrid techniques
  - symbol-manipulation techniques
  - case-based techniques

  - (qualitative techniques, model-based techniques, temporal reasoning techniques, neural networks)

Structure and characteristics

- KBSs are computer systems
  - contain stored knowledge
  - solve problems like humans would
- KBSs are AI programs with program structure of new type
  - knowledge-base (rules, facts, meta-knowledge)
  - inference engine (reasoning and search strategy for solution, other services)
- characteristics of KBSs:
  - intelligent information processing systems
  - representation of domain of interest →symbolic representation
  - problem solving →by symbol-manipulation
  - ⇒symbolic programs



Main components

- knowledge-base (KB)
  - knowledge about the field of interest (in natural language-like formalism)
  - symbolically described system-specification
  - KNOWLEDGE-REPRESENTATION METHOD!
- inference engine
  - „engine" of problem solving (general problem solving knowledge)
  - supporting the operation of the other components
  - PROBLEM SOLVING METHOD!

- case-specific database
    - auxiliary component
    - specific information (information from outside, initial data of the concrete problem)
    - information obtained during reasoning
- explanation subsystem
  explanation of system' actions in case of user' request
  typical explanation facilities:
    - explanation during problem solving:
        - WHY... (explanative reasoning, intelligent help, tracing information about the actual reasoning steps)
        - WHAT IF... (hypothetical reasoning, conditional assignment and its consequences, can be withdrawn)
        - WHAT IS ... (gleaning in knowledge-base and case-specific database)
    - explanation after problem solving:
        - HOW ... (explanative reasoning, information about the way the result has been found)
        - WHY NOT ... (explanative reasoning, finding counter-examples)
        - WHAT IS ... (gleaning in knowledge-base and case-specific database)
- knowledge acquisition subsystem
    - main tasks:
        - checking the syntax of knowledge elements
        - checking the consistency of KB (verification, validation)
        - knowledge extraction, building KB
        - automatic logging and book-keeping of the changes of KB
        - tracing facilities (handling breakpoints, automatic monitoring and reporting the values of knowledge elements)
- user interface (→user)
    - dialogue on natural language (consultation/ suggestion)
- specially intefaces
    - database and other connections
- developer interface (→knowledge engineer, human expert)
- the main tasks of the knowledge engineer:
    - knowledge acquisition and design of KBS: determination, classification, refinement and formalization of methods, thumb-rules and procedures
    - selection of knowledge representation method and reasoning strategy
    - implementation of knowledge-based system
    - verification and validation of KB

- KB maintenance

Topic – 3: Active and Deductive Databases

Active Databases

- An *active database* is a database in which some operations are automatically executed once a given *situation* arises
- The situation may correspond to the fact that:
  - Some specified events arise, or
  - Specific conditions or state transitions are detected
- An *active rule* (trigger) is a language construct for defining the system reactions

An **active database** is a database that includes active rules, mostly in the form of ECA rules.

Active database systems enhance traditional database functionality with powerful rule-processing capabilities, providing a uniform and efficient mechanism for many database system applications.

Among these applications are integrity constraints, views, authorization, statistics gathering, monitoring and alerting, knowledge-based systems, expert systems, and workflow management.

Triggers is a technique for specifying certain types of active rules.

Event Condition Action (ECA) is a short-cut for referring to the structure of in event driven architecture and database systems.

• Such a rule did traditionally consist of three parts:

• The *event* part specifies the signal that triggers the invocation of the rule

• The *condition* part is a logical test that, if satisfied or evaluates to true, causes the action to be carried out

• The *action* part consists of updates or invocations on the local data

• This structure was used by the early research in active databases which started to use the term ECA. Current state of art ECA rule engines uses a many variations on rule structure.

- ***What is an event?***

"An event is something which happens, which is of interest, and which can be mapped onto some time instant"

- ***Types of event:***
    - **Data modifications**: insertions, deletions, modifications
    - **Data accesses**: queries on tables
    - **DBMS operations**: login of users, transaction management and authorization
    - **Temporal events**: January 12th 2006 at 10 (absolute), each 10 minutes (periodic events)...
    - **Application defined events (external event)**: room temperature too high
- ***What is a condition?***

"A condition is an additional check that is executed when the trigger is evaluated and ***before*** *the action is executed"*

- **Predicates**: WHERE clause of SQL; it is useful to have simple predicates because their evaluation is efficient
- **Queries**: the condition is true if and only if the query return the empty set (a possible meaning)
- **Application procedures**: call to a procedure

- ***What is an action?***

"An action is a sequence of operations that is executed when its trigger is considered and the trigger condition is true"

- **Types of actions:**
    - **Data modifications**: insertion, deletion, update
    - **Data access**: queries on tables
    - **Other commands**: data definition, transaction actions (commit, rollback), grant and revoke of permissions
    - **Application procedures**: call to a procedure

- Database system augmented with rule handling
    - Active approach to managing integrity constraints
    - *ECA rules*: event, condition, action
- Many other uses have been found for active rules
    - Maintaining materialized views
    - Managing derived data
    - Coordinating distributed data management

  - Providing transaction models
  - Etc.
- Provably correct universal solutions lacking…
  - Specifying rules
  - Rules analysis (termination, confluence, observable determinism)

- Perhaps the problem is that ADBs should not be viewed as DBs?

Active Database and Triggers

- The most common form of triggers is thus:

ON *event* IF *condition* THEN *action*

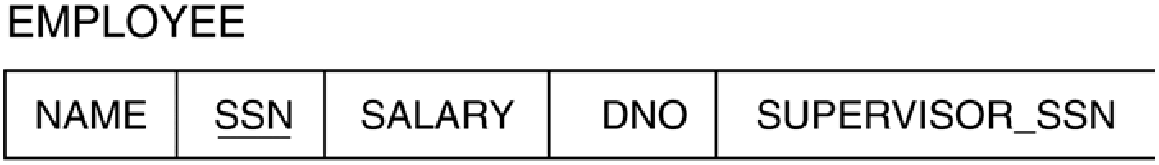
1. If the event arises, the condition is evaluated
2. if the condition is satisfied, the action is executed

- The active rule paradigm originates from the notion of production rules of Artificial Intelligence
- Production rules do not typically have events; they have the form (CA):

IF *condition* THEN *action*

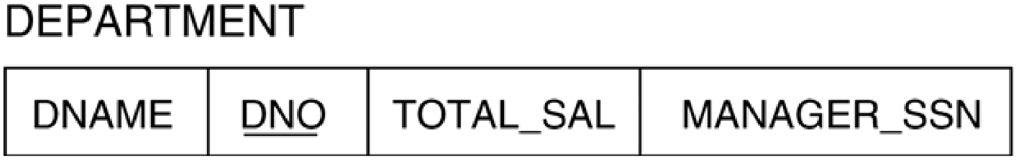
Example: Consider a simplified version of the
Company database

• In this version, the TOTAL_SAL attribute is a derived attribute, whose value should be the sum of the salaries of all employees who are assigned to the particular department.



EMPLOYEE

NAME	SSN	SALARY	DNO	SUPERVISOR_SSN

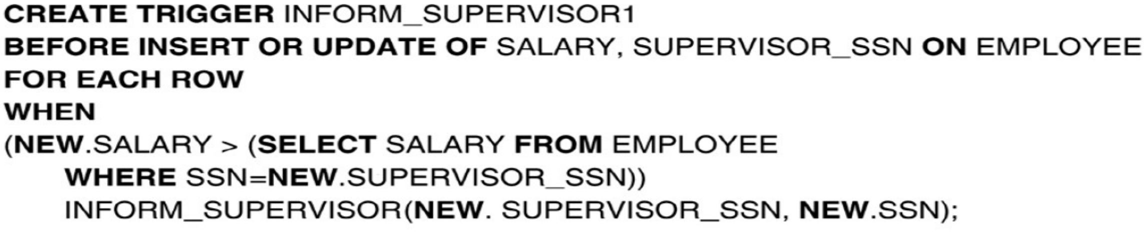


DEPARTMENT

DNAME	DNO	TOTAL_SAL	MANAGER_SSN

Maintaining the correct value for TOTAL_SAL can be done via an active rule. The events that may cause a change in the value of TOTAL_SAL are:

1) Inserting new employee tuples.
2) Changing the salary of existing employees.
3) Changing the assignment of existing employees from one department to another.
4) Deleting (one or more) employee tuples.



```
CREATE TRIGGER INFORM_SUPERVISOR1
BEFORE INSERT OR UPDATE OF SALARY, SUPERVISOR_SSN ON EMPLOYEE
FOR EACH ROW
WHEN
(NEW.SALARY > (SELECT SALARY FROM EMPLOYEE
    WHERE SSN=NEW.SUPERVISOR_SSN))
    INFORM_SUPERVISOR(NEW. SUPERVISOR_SSN, NEW.SSN);
```

Design Issues for Active Databases

The first issue concerns, **activation, deactivation**, and **grouping** of rules.
– The **activate** command will make the rule active again.
– The **deactivate** command will make the trigger event not be triggered.
– The **delete** command deletes the rule from the system.
– The**rule set** option can be used to group rules (so, the whole set of rules can be activated, deactivated, or dropped).
-The **PROCESS RULES** command can trigger a rule or rule set.

The second issue concerns whether the triggered action should be executed **before, after**, or **concurrently with** the trigger event.
• A related issue is whether the action being executed should be considered as a **separate action** or whether it should **be part of the same transaction** that triggered the rule.
• The **rule condition evaluation** is also known as **rule consideration**.

Three main possibilities for rule consideration:
1) Immediate consideration: the condition is evaluated as part of the same transaction as the trigger event, and is evaluated immediately; in one of the following forms
a) Before executing the trigger event.
b) After executing the trigger event.
c) Instead of executing the trigger event.
2) Deferred consideration: the condition is evaluated at the end of transaction that include the trigger event.
3) Detached consideration: the condition is evaluated as a separate transaction.

Main activities in an ADBMS:

1. Detect the events and activate the corresponding rules
2. Select and execute the activated rule (also called *reactive process*)

These two activities can be executed concurrently
A possible model (two activities):

```
Activity 1
While true do
        detect events
        activate the rules associated with the detected events
endWhile

Activity 2 (reactive process)
                While there are still active rules Do
                        (1) select a rule R for consideration
                        (2) evaluate the condition of R
                        (3) If the condition of R is true Then
                            execute the action of R  endIf
                endWhile
```

Applications for Active Databases

To allow **notification** of certain conditions that occur (e.g., to monitor the temperature of an industrial furnace).
• To **enforce integrity constraints** by specifying the types of events that may cause the constraints to be violated.
• Automatic **maintenance of derived data** (e.g., the derived attribute TOTAL_SAL in the simplified version of Company schema).

DBs Vs ISs

State	User data	User data, logs, histories, user profiles
Job	Updates & queries of data	Data-blocked serives to users
Output	Determined completely by query/update specification	Individualized based on user history & preferences
Integrity concerns	Static integrity (of a DB state)	Static & dynamic intergrity (of IS behaviour), maintained actively.
Nature	Static, algorithmic	Dynamic, interactive

	transformation of engine	service providing system

Information System = Database + Interactions

The two views of Active Databases

State	User data	User data, rule-related logs , histories, rule-related user profiles.
Job	Updates & queries of data	Data-blocked serives to users
Output	Determined completely by query/update specification	Individualized based on user history & preferences
Integrity concerns	Static integrity, maintained actively via rules.	Static & dynamic intergrity, maintained actively.
Nature	Static, algorithmic transformation of engine	Dynamic, interactive service providing system

- **Active DBs fall within that blurry area**
  - a DB augmented with active rule handling (to perform *system operations*)
  - a data-intensive IS restricted to rule-handling services

ADB Wish List

Rule instances

- Support multiple instances of the same rule
- Now possible only when the *condition* part of their ECA structure differs.
- Can be directly mapped to different instances of IS services.

Rule history

- Store the history of events, conditions, actions for each rule instance.
- To help transactions handle dynamic integrity violations during rule execution.

Rule interaction

- Allow rules to enable, disable, or wait for other rules.
- As separate functionality rather than by extending the *condition* part of ECA structure.
- Rules need not be aware of external control over their behavior.
- For easier formulization of synchronization across semantic services

Deductive Databases

A Deductive Database system is a database system that includes capabilities to define "Rules", which can deduce additional information from the facts that are stored in a database.
• Mathematical logic is a part of the theoretical foundation for the Deductive Database, that is why it is also called a "logic database".

DDB uses a declarative language.
• Declarative Language means a language that defines what a program wants to achieve, rather than one that specifies the details of how to achieve it.
• A deduction mechanism within the system can deduce new facts from the database by interpreting these rules.

Types of specifications of DDB
• There are two main types of specifications:
– Facts
– Rules

Facts
• Facts are specified as the same way the relations are specified in the Relational Database
– Except it is not necessary to include the attribute names.
– The meaning of an attribute value in a tuple is determined solely by its position in the tuple.

Rules
• They specify "virtual relations" that are not actually stored but that can be formed from the facts by applying deduction mechanisms based on the rule specifications.
– somewhat similar to relational views, but different in the sense that it may involve recursion.

Notations
• Deductive database uses Datalog notation, which is a subset of Prolog.
– Prolog is a language based on logic.
– Datalog is a deductive query language similar to Prolog but more suitable for database applications.

Prolog and Datalog notations

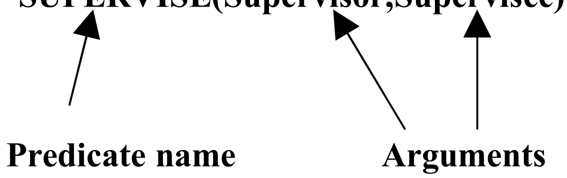
- The notation used in Prolog/Datalog is based on providing “predicates” with unique names. A predicate has an implicit meaning, which is stated by the predicate name and a fixed number of arguments.
- If the arguments are all constant values, then predicate states that the fact is true.

Prolog convention

- All constant values in a predicate are either numeric or character strings starting with lower case letters only.
- Variable names always start with an upper case letter.

Example of notation

- **SUPERVISE(Supervisor,Supervisee)**



Datalog notation

- **Datalog program is built from basic objects called “atomic formulas”**

– It is cutomary to define the syntax of logic-based languages by describing the syntax of atomic formulas and identifying how they can be combined to form a program**.**

Built-in predicates

- A number of built-in predicates are included in Datalog, which can be used to construct atomic formulas.
- Built-in predicates are of two types :
- Binary comparison predicates over ordered domains
- Comparison predicates over ordered or unordered domains.

Interpretations of Rules

- Two main alternatives for interpreting the theoretical meaning of rules :

– Proof-theoretic

– Model-theoretic

Proof-theoretic interpretation

- Axioms - the facts and rules to be true statements

– **Ground Axioms-** The facts that are given to be true.
– **Deductive Axioms-** Rules are called deductive axioms, because they can be used to deduce new facts.

Model Theoretic interpretation

• For a finite or infinite domain of constant values, we assign to a predicate every possible combination of values as arguments. Then we determine whether the predicate is true or false.

Model

• An interpretation is called a model for a specific set of rules, if those rules are always true under that interpretation.

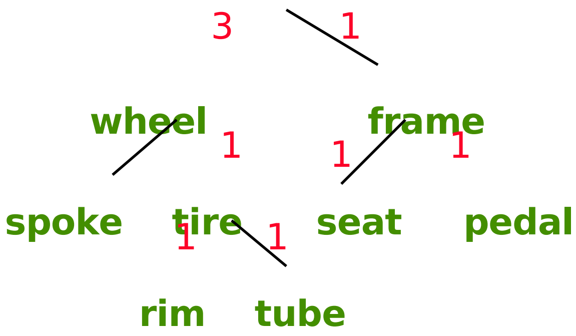
Motivation

- ❖ SQL-92 cannot express some queries:
    - ▪ Are we running low on any parts needed to build a ZX600 sports car?
    - ▪ What is the total component and assembly cost to build a ZX600 at today's part prices?
- ❖ Can we extend the query language to cover such queries?
    - ▪ Yes, by adding recursion.

Datalog

- ❖ SQL queries can be read as follows: "If some tuples exist in the From tables that satisfy the Where conditions, then the Select tuple is in the answer."
- ❖ Datalog is a query language that has the same if-then flavor:
    - ▪ New: The answer table can appear in the From clause, i.e., be defined recursively.
    - ▪ Prolog style syntax is commonly used

Example



Find the components of a trike?

We can write a relational algebra query to compute the answer on ***the given instance of Assembly*.**

But there is no R.A. (or SQL-92) query that computes the answer on ***all Assembly instances***.

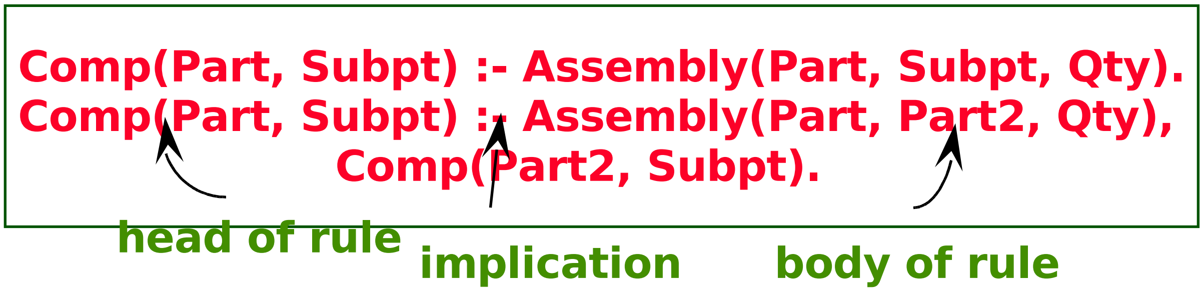
part	subpart	number
trike	wheel	3
trike	frame	1
frame	seat	1
frame	pedal	1
wheel	spoke	2
wheel	tire	1
tire	rim	1
tire	tube	1

Assembly instance

The Problem with R.A. and SQL-92

- Intuitively, we must join Assembly with itself to deduce that trike contains spoke and tire.
  - Takes us one level down Assembly hierarchy.
  - To find components that are one level deeper (e.g., rim), need another join.
  - To find all components, need as many joins as there are levels in the given instance!
- For any relational algebra expression, we can create an Assembly instance for which some answers are not computed by including more levels than the number of joins in the expression

A Datalog Query that Does the Job



Comp(Part, Subpt) :- Assembly(Part, Subpt, Qty).
Comp(Part, Subpt) :- Assembly(Part, Part2, Qty),
Comp(Part2, Subpt).

head of rule implication body of rule

Can read the second rule as follows:
"**For all** values of Part, Subpt and Qty,
if there is a tuple (Part, Part2, Qty) in Assembly
and a tuple (Part2, Subpt) in Comp,
then there must be a tuple (Part, Subpt) in Comp."

Using a Rule to Deduce New Tuples

- Each rule is a *template*: by assigning constants to the variables in such a way that each body "literal" is a tuple in the corresponding relation, we identify a tuple that must be in the head relation.
  - By setting Part=trike, Subpt=wheel, Qty=3 in the first rule, we can deduce that the tuple <trike,wheel> is in the relation Comp.
  - This is called an <u>inference</u> using the rule.
  - Given a set of tuples, we <u>apply</u> the rule by making all possible inferences with these tuples in the body.

<u>Topic – 4: Parallel Databases</u>

- Parallel machines are becoming quite common and affordable
  - Prices of microprocessors, memory and disks have dropped sharply
  - Recent desktop computers feature multiple processors and this trend is projected to accelerate
- Databases are growing increasingly large
  - large volumes of transaction data are collected and stored for later analysis.
  - multimedia objects like images are increasingly stored in databases
- Large-scale parallel database systems increasingly used for:
  - storing large volumes of data
  - processing time-consuming decision-support queries

  - providing high throughput for transaction processing

Parallelism in Databases

- Data can be partitioned across multiple disks for parallel I/O.
- Individual relational operations (e.g., sort, join, aggregation) can be executed in parallel
  - data can be partitioned and each processor can work independently on its own partition.
- Queries are expressed in high level language (SQL, translated to relational algebra)
  - makes parallelization easier.
- Different queries can be run in parallel with each other. Concurrency control takes care of conflicts.
- Thus, databases naturally lend themselves to parallelism.

I/O Parallelism

- Reduce the time required to retrieve relations from disk by partitioning
- the relations on multiple disks.
- Horizontal partitioning – tuples of a relation are divided among many disks such that each tuple resides on one disk.
- Partitioning techniques (number of disks = *n*):

Round-robin:

- Send the *i*th tuple inserted in the relation to disk *i* mod *n*.

Hash partitioning:

  - Choose one or more attributes as the partitioning attributes.
  - Choose hash function *h* with range 0…*n* - 1
  - Let *i* denote result of hash function *h* applied to the partitioning attribute value of a tuple. Send tuple to disk *i*.

- **Range partitioning:**
  - Choose an attribute as the partitioning attribute.
  - A partitioning vector [*v*o, *v*1, ..., *vn*-2] is chosen.
  - Let *v* be the partitioning attribute value of a tuple. Tuples such that *v*i ≤ *vi*+1 go to disk *I* + 1. Tuples with *v* < *v0* go to disk 0 and tuples with *v* ≥ *v*n-2 go to disk *n*-1.

E.g., with a partitioning vector [5,11], a tuple with partitioning attribute value of 2 will go to disk 0, a tuple with value 8 will go to disk 1, while a tuple with value 20 will go to disk2.

Comparison of Partitioning Techniques

- Evaluate how well partitioning techniques support the following types of data access:

1.Scanning the entire relation.

2.Locating a tuple associatively – **point queries**.

- E.g., $r.A = 25$.

3.Locating all tuples such that the value of a given attribute lies within a specified range – **range queries**.

- E.g., $10 \leq r.A < 25$.

Round robin:

- Advantages
  - Best suited for sequential scan of entire relation on each query.
  - All disks have almost an equal number of tuples; retrieval work is thus well balanced between disks.
- Range queries are difficult to process
  - No clustering -- tuples are scattered across all disks

Hash partitioning:

- Good for sequential access
  - Assuming hash function is good, and partitioning attributes form a key, tuples will be equally distributed between disks
  - Retrieval work is then well balanced between disks.
- Good for point queries on partitioning attribute
  - Can lookup single disk, leaving others available for answering other queries.
  - Index on partitioning attribute can be local to disk, making lookup and update more efficient
- No clustering, so difficult to answer range queries

- Range partitioning:
- Provides data clustering by partitioning attribute value.
- Good for sequential access
- Good for point queries on partitioning attribute: only one disk needs to be accessed.
- For range queries on partitioning attribute, one to a few disks may need to be accessed
  - Remaining disks are available for other queries.
  - Good if result tuples are from one to a few blocks.
  - If many blocks are to be fetched, they are still fetched from one to a few disks, and potential parallelism in disk access is wasted
    - Example of execution skew.

Partitioning a Relation across Disks

- If a relation contains only a few tuples which will fit into a single disk block, then assign the relation to a single disk.
- Large relations are preferably partitioned across all the available disks.
- If a relation consists of *m* disk blocks and there are *n* disks available in the system, then the relation should be allocated **min**(*m,n*) disks.

Handling of Skew

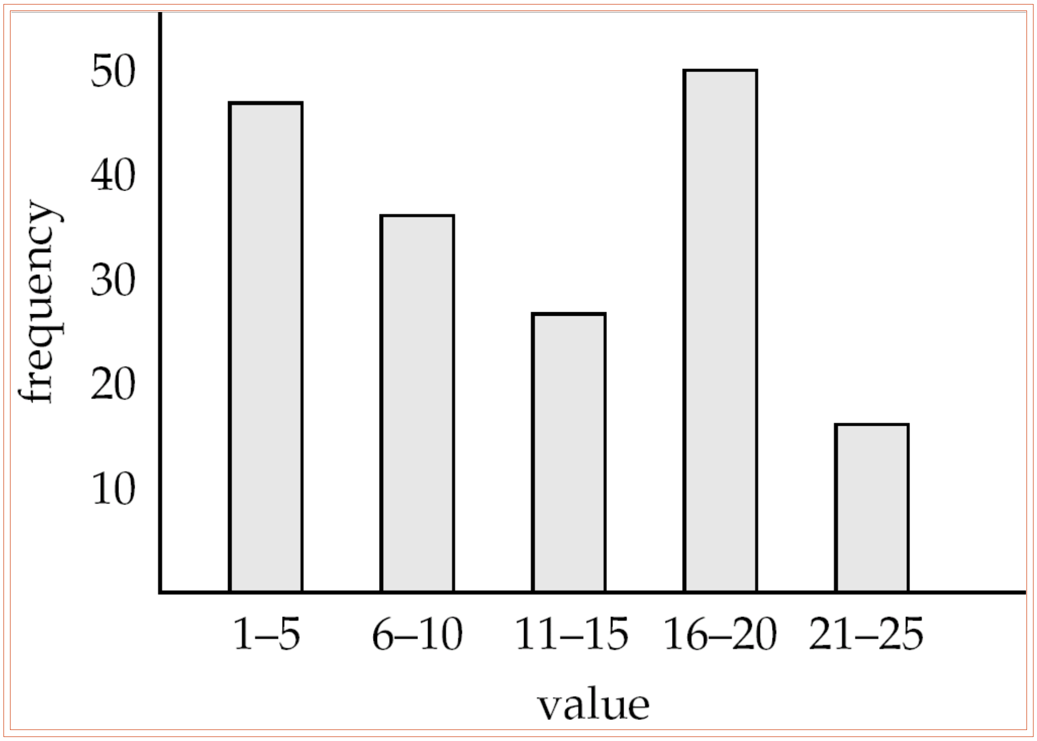
- The distribution of tuples to disks may be **skewed** — that is, some disks have many tuples, while others may have fewer tuples.
- **Types of skew:**
    - **Attribute-value skew.**
        - Some values appear in the partitioning attributes of many tuples; all the tuples with the same value for the partitioning attribute end up in the same partition.
        - Can occur with range-partitioning and hash-partitioning.
    - **Partition skew**.
        - With range-partitioning, badly chosen partition vector may assign too many tuples to some partitions and too few to others.
        - Less likely with hash-partitioning if a good hash-function is chosen.

Handling Skew in Range-Partitioning

- To create a balanced partitioning vector (assuming partitioning attribute forms a key of the relation):
    - Sort the relation on the partitioning attribute.
    - Construct the partition vector by scanning the relation in sorted order as follows.
        - After every 1/*nth* of the relation has been read, the value of the partitioning attribute of the next tuple is added to the partition vector.
    - *n* denotes the number of partitions to be constructed.
    - Duplicate entries or imbalances can result if duplicates are present in partitioning attributes.
- Alternative technique based on **histograms** used in practice

Handling Skew using Histograms

- Balanced partitioning vector can be constructed from histogram in a relatively straightforward fashion
    - Assume uniform distribution within each range of the histogram
- Histogram can be constructed by scanning relation, or sampling (blocks containing) tuples of the relation



Handling Skew Using Virtual Processor Partitioning

- Skew in range partitioning can be handled elegantly using **virtual processor partitioning**:
  - create a large number of partitions (say 10 to 20 times the number of processors)
  - Assign virtual processors to partitions either in round-robin fashion or based on estimated cost of processing each virtual partition
- Basic idea:
  - If any normal partition would have been skewed, it is very likely the skew is spread over a number of virtual partitions
  - Skewed virtual partitions get spread across a number of processors, so work gets distributed evenly!

Interquery Parallelism

- Queries/transactions execute in parallel with one another.
- Increases transaction throughput; used primarily to scale up a transaction processing system to support a larger number of transactions per second.
- Easiest form of parallelism to support, particularly in a shared-memory parallel database, because even sequential database systems support concurrent processing.

- More complicated to implement on shared-disk or shared-nothing architectures
  - Locking and logging must be coordinated by passing messages between processors.
  - Data in a local buffer may have been updated at another processor.
  - **Cache-coherency** has to be maintained — reads and writes of data in buffer must find latest version of data.

Cache Coherency Protocol

- Example of a cache coherency protocol for shared disk systems:
  - Before reading/writing to a page, the page must be locked in shared/exclusive mode.
  - On locking a page, the page must be read from disk
  - Before unlocking a page, the page must be written to disk if it was modified.
- More complex protocols with fewer disk reads/writes exist.
- Cache coherency protocols for shared-nothing systems are similar. Each database page is assigned a *home* processor. Requests to fetch the page or write it to disk are sent to the home processor.

Intraquery Parallelism

- Execution of a single query in parallel on multiple processors/disks; important for speeding up long-running queries.
- Two complementary forms of intraquery parallelism :
  - **Intraoperation Parallelism** – parallelize the execution of each individual operation in the query.
  - **Interoperation Parallelism** – execute the different operations in a query expression in parallel.

the first form scales better with increasing parallelism because the number of tuples processed by each operation is typically more than the number of operations in a query

Design of Parallel Systems

Some issues in the design of parallel systems:

- Parallel loading of data from external sources is needed in order to handle large volumes of incoming data.
- Resilience to failure of some processors or disks.
  - Probability of some disk or processor failing is higher in a parallel system.
  - Operation (perhaps with degraded performance) should be possible in spite of failure.
  - Redundancy achieved by storing extra copy of every data item at another processor.
- On-line reorganization of data and schema changes must be supported.

  - For example, index construction on terabyte databases can take hours or days even on a parallel system.
    - Need to allow other processing (insertions/deletions/updates) to be performed on relation even as index is being constructed.
  - Basic idea: index construction tracks changes and ``catches up'‘ on changes at the end.
- Also need support for on-line repartitioning and schema changes (executed concurrently with other processing).

Topic – 5: Multimedia Databases

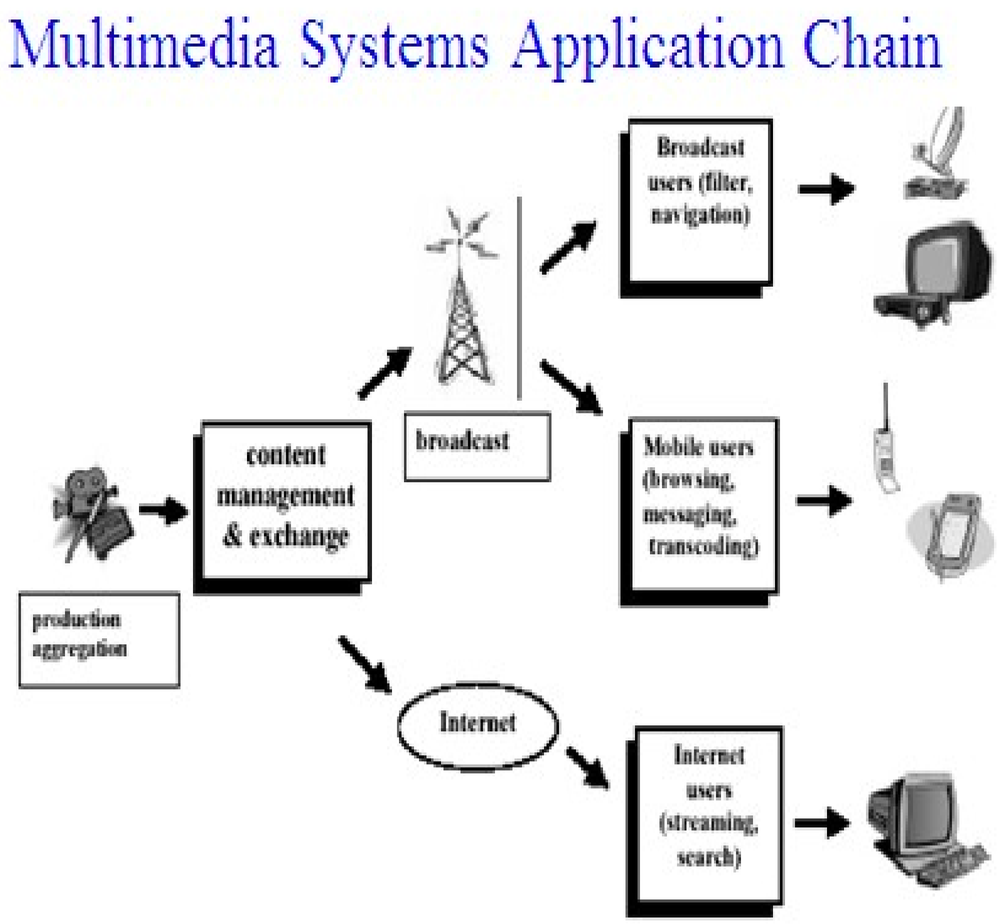
Multimedia System

A computer hardware/software system used for

- Acquiring and Storing
- Managing
- Indexing and Filtering
- Manipulating (quality, editing)
- Transmitting (multiple platforms)
- Accessing large amount of visual information like, Images, video, graphics, audios and associated multimedia

Examples: image and video databases, web media search engines, mobile media navigator, etc.

- Share Digital Information
- New Content Creation Tools
- Deployment of High-Speed Networks
- New Content Services
- Mobile Internet
- 3D graphics, network games
- Media portals
- Standards become available: coding, delivery, and description.


Multimedia Systems Application Chain
production aggregation
content management & exchange
broadcast
Broadcast users (filter, navigation)
Mobile users (browsing, messaging, transcoding)
Internet
Internet users (streaming, search)

Access multimedia information
anytime
anywhere
on any device
from any source
anything
Network/device transparent
Quality of service (graceful degradation)
Intelligent tools and interfaces
Automated protection and transaction

Multimedia data types

- Text
- Image
- Video
- Audio

- mixed multimedia data

Designing MMDBs

Characteristics of multimedia data that have impacts on the design of MMDBs include :
the huge size of MMDBs, temporal nature, richness of content, complexity of representation and subjective interpretation. The major challenges in designing multimedia databases arise from several requirements they need to satisfy such as the following:

• **Manage different types of input, output, and storage devices**. Data input can be
from a variety of devices such as scanners, digital camera for images, microphone, MIDI devices for audio, video cameras. Typical output devices are high-resolution monitors for images and video, and speakers for audio.

• **Handle a variety of data compression and storage formats**. The data encoding has a variety of formats even within a single application. For instance, in medical applications, the MRI images of brain has lossless or very stringent quality of lossy coding technique, while the X-ray images of bones can be less stringent. Also, the radiological image data, the ECG data, other patient data, etc. have widely varying formats.

Support different computing platforms and operating systems. Different users operate computers and devices suited to their needs and tastes. But they need the same kind of user-level view of the database.

• **Integrate different data models**. Some data such as numeric and textual data are best handled using a relational database model, while some others such as video documents are better handled using an object-oriented database model. So these two models should coexist together in MMDBs.

• **Offer a variety of user-friendly query systems suited to different kinds of media**. From a user point of view, easy-to-use queries and fast and accurate retrieval of information is highly desirable. The query for the same item can be in different forms. For example, a portion of interest in a video can be queried by using either

1) a few sample video frames as an example,
2) a clip of the corresponding audio track or
3) a textual description using keywords

• **Handle different kinds of indices**. The inexact and subjective nature of multimedia data has rendered keyword-based indices and exact and range searches used in traditional databases ineffective. For example, the retrieval of records of persons based on social security number is precisely defined, but the retrieval of records of persons having certain facial features from a database of facial images requires, content-based queries and similarity-based retrievals. This

requires indices that are content dependent, in addition to key-word indices.
- **Develop measures of data similarity that correspond well with perceptual similarity**. Measures of similarity for different media types need to be quantified to correspond well with the perceptual similarity of objects of those data types. These need to be incorporated into the search process.

Provide transparent view of geographically distributed data. MMDBs are likely to be a distributed nature. The media data resides in many different storage units possibly spread out geographically. This is partly due to the changing nature of computation and computing resources from centralized to networked and distributed.
- **Adhere to real-time constraints for the transmission of media data**. Video and

audio are inherently temporal in nature. For example, the frames of a video need to be presented at the rate of at least 30 frames/sec. for the eye to perceive continuity in the video.
- **Synchronize different media types while presenting to user**. It is likely that different media types corresponding to a single multimedia object are stored in different formats, on different devices, and have different rates of transfer. Thus they need to be periodically synchronized for presentation.

Multimedia Databases
- Multimedia database management

(NSF, Fuji Electric, AT&T)

– Video modeling and management

– Multimedia document management
- Distributed multimedia systems

(NSF, AFRL, IBM, Intel, Siemens)
- High-performance multimedia database architecture for storage management

(NSF, AT&T)

Topic – 6: Image Databases

Image Database is searchable electronic catalog or database which allows you to organize and list images by topics, modules, or categories. The Image Database will provide the student with important information such as image title, description, and thumbnail picture. Additional information can be provided such as creator of the image, filename, and keywords that will help students to search through the database for specific images. Before you and your students can use Image Database, you must add it to your course

An image retrieval system is a computer system for browsing, searching and retrieving images from a large database of digital images.

Most traditional and common methods of image retrieval utilize some method of adding metadata such as captioning, keywords, or descriptions to the images so that retrieval can be performed over the annotation words.

Manual image annotation is time-consuming, laborious and expensive; to address this, there has been a large amount of research done on automatic image annotation. Additionally, the increase in social web applications and the semantic web have inspired the development of several web-based image annotation tools.

The first microcomputer-based image database retrieval system was developed at MIT, in the 1980s, by Banireddy Prasaad, Amar Gupta, Hoo-min Toong, and Stuart Madnick.[1]

Image search is a specialized data search used to find images. To search for images, a user may provide query terms such as keyword, image file/link, or click on some image, and the system will return images "similar" to the query. The similarity used for search criteria could be meta tags, color distribution in images, region/shape attributes, etc.

- Image meta search - search of images based on associated metadata such as keywords, text, etc.
- Content-based image retrieval (CBIR) – the application of computer vision to the image retrieval. CBIR aims at avoiding the use of textual descriptions and instead retrieves images based on similarities in their contents (textures, colors, shapes etc.) to a user-supplied query image or user-specified image features.
  - List of CBIR Engines - list of engines which search for images based image visual content such as color, texture, shape/object, etc.

Data Scope

It is crucial to understand the scope and nature of image data in order to determine the complexity of image search system design. The design is also largely influenced by factors such as the diversity of user-base and expected user traffic for a search system. Along this dimension, search data can be classified into the following categories:

- *Archives* - usually contain large volumes of structured or semi-structured homogeneous data pertaining to specific topics.
- *Domain-Specific Collection* - this is a homogeneous collection providing access to controlled users with very specific objectives. Examples of such a collection are biomedical and satellite image databases.

- *Enterprise Collection* - a heterogeneous collection of images that is accessible to users within an organization's intranet. Pictures may be stored in many different locations.
- *Personal Collection* - usually consists of a largely homogeneous collection and is generally small in size, accessible primarily to its owner, and usually stored on a local storage media.
- *Web* - World Wide Web images are accessible to everyone with an Internet connection. These image collections are semi-structured, non-homogeneous and massive in volume, and are usually stored in large disk arrays.

There are evaluation workshops for image retrieval systems aiming to investigate and improve the performance of such systems.

- ImageCLEF - a continuing track of the Cross Language Evaluation Forum that evaluates systems using both textual and pure-image retrieval methods.
- Content-based Access of Image and Video Libraries - a series of IEEE workshops from 1998 to 2001.

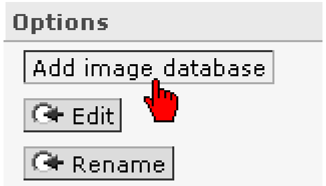
Create an Image Database

An Image Database can ultimately contain as many images as you would like. You can put all images in one database or create multiple databases.

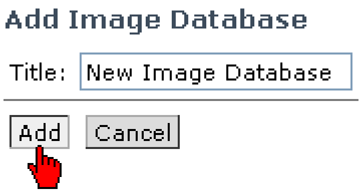
Upload the image files that you want to include in the database. See How to set up WebDAV to drag and drop files from your desktop to your course. Or see Manage Files to upload files.

From the Homepage or the Course Menu select the Image Database link.

The Image Database page displays. Select Add image database button from Options



The Add Image Database page displays. Type desired database title in **Title:** field and click the **Add** button.



Add Image Database

Title: New Image Database

The new image database displays in the Available databases. Select the link to the new image database you just created.

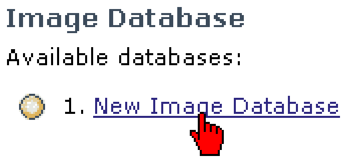


Image Database

Available databases:

1. New Image Database

The Image Database Screen displays. Select the **Add Image** button

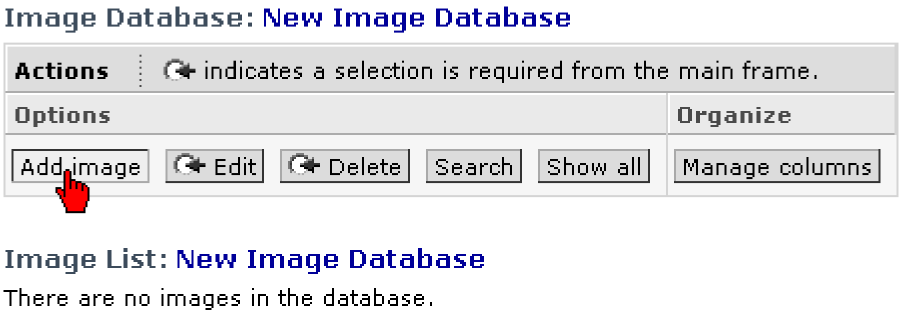


Image Database: New Image Database

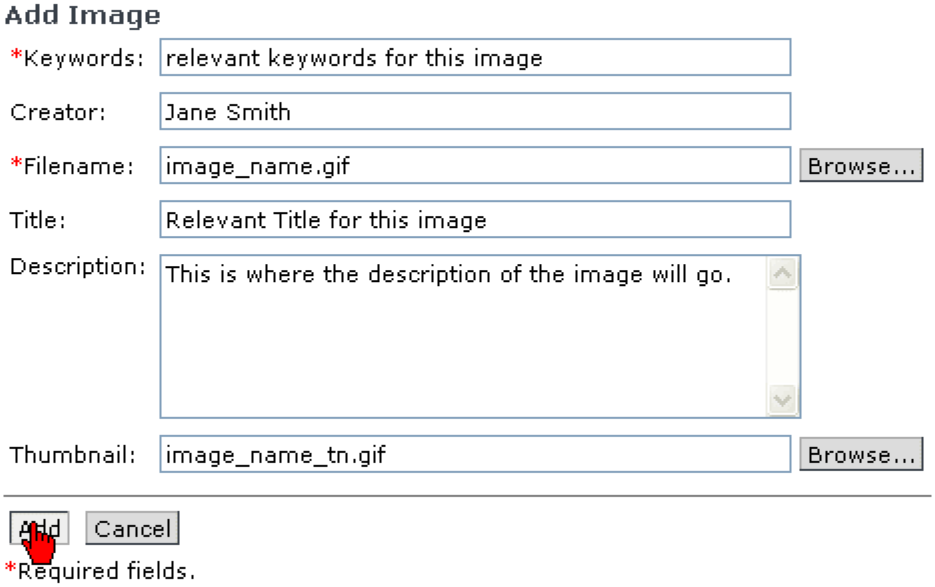
Actions | indicates a selection is required from the main frame.

Image List: New Image Database

There are no images in the database.

The Add Image screen displays. Type in relevant keywords in the ***Keywords** field. Type the owner of the image in the **Creator:** field. Type the path and filename in the ***Filename:** field or click the browse button and find the file in the My-Files area. Type a relevant title for this image in the **Title:** field. Type in the image description in the **Description:** field. Type the path and filename of the image thumbnail in the **Thumbnail:** field or click the browse button and find the file in the My-Files area. Select the **Add** button.

Note: Creator, Title, Description and Thumbnail are not required fields and do not require an entry.



Add Image

*Keywords:	relevant keywords for this image	
Creator:	Jane Smith	
*Filename:	image_name.gif	Browse...
Title:	Relevant Title for this image	
Description:	This is where the description of the image will go.	
Thumbnail:	image_name_tn.gif	Browse...

*Required fields.

Note: If you use a .gif or .jpg the database will automatically create a thumbnail when you select add.

The Image Database page displays with the new image and information.

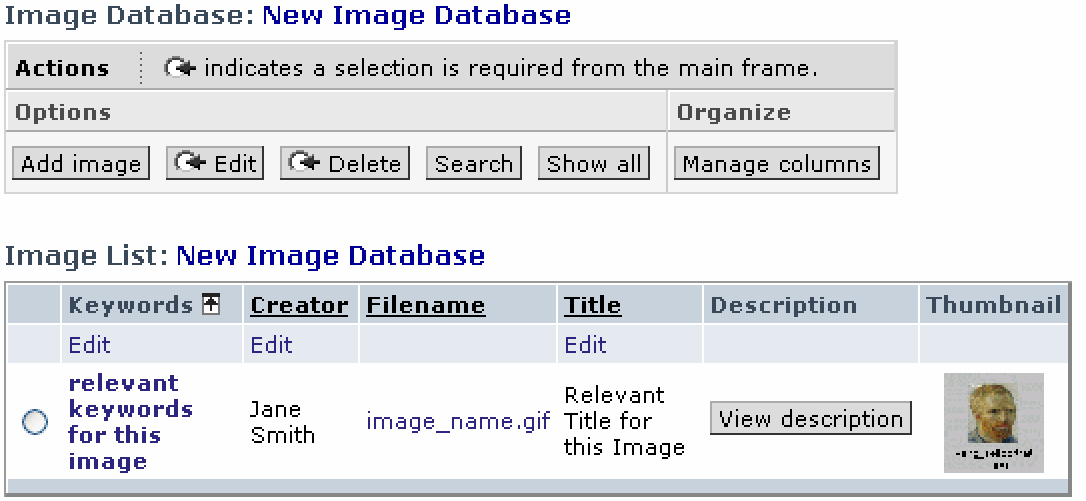


Image Database: New Image Database

Image List: New Image Database

	Keywords	Creator	Filename	Title	Description	Thumbnail
	Edit	Edit		Edit		
○	relevant keywords for this image	Jane Smith	image_name.gif	Relevant Title for this Image	View description	

To add additional images to the database repeat the above steps.

II. Edit an Image Record

You may find that you have information about an image that needs to be edited. If you have text in one column that needs to be changed, see Columns/Edit. If you have additional image information that needs to be changed, follow the steps below.

From the Homepage or the Course Menu select the Image Database link.

The Available Database page displays. Select the link to the image database that contains the image you want to edit. The Image Database page displays. Select the radio button beside the image you would like to edit and select the **Edit** button.

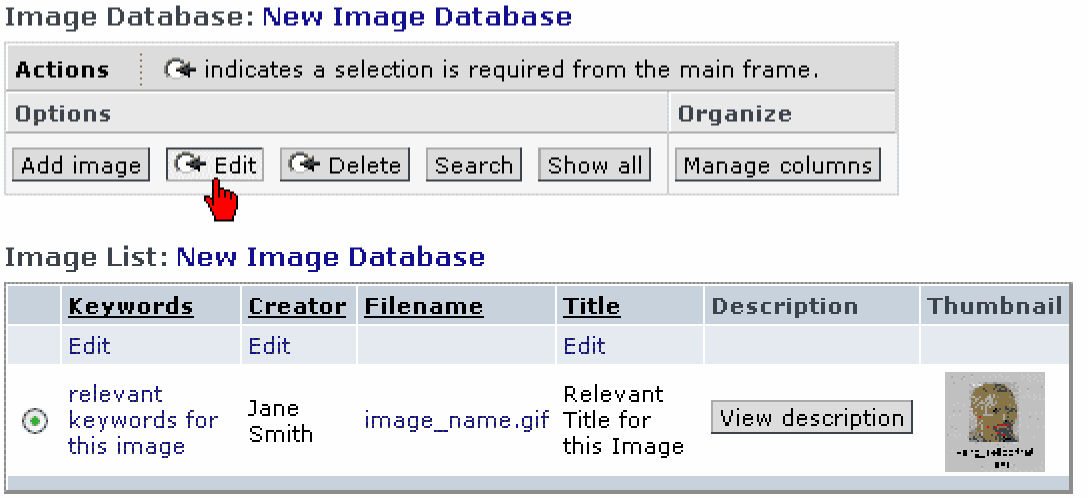
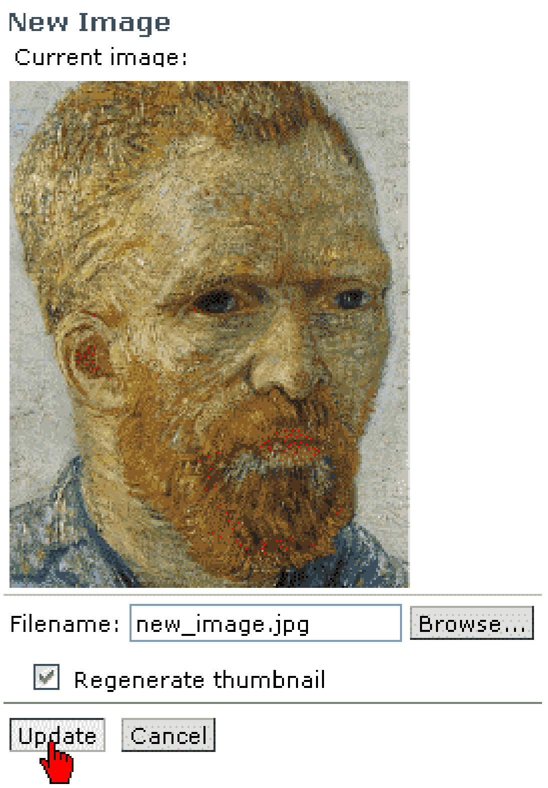


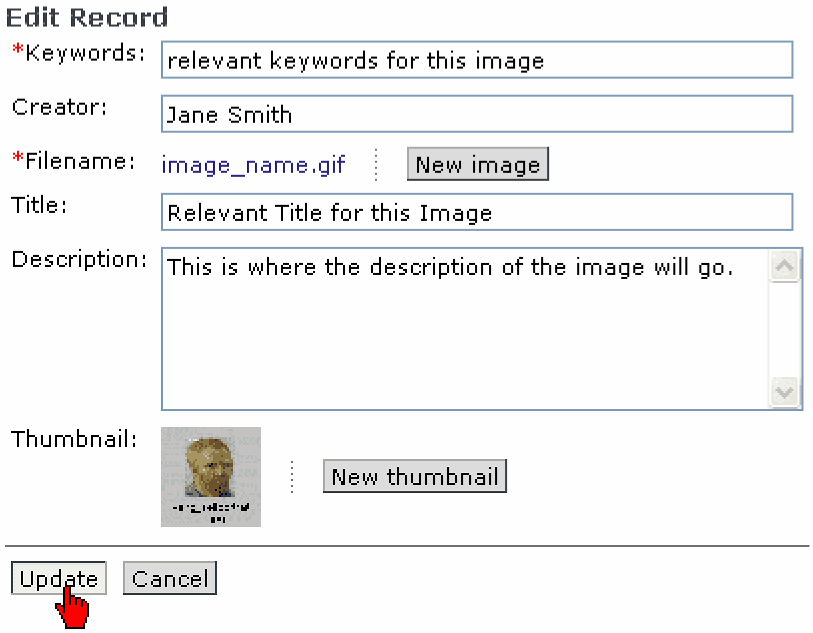
Image Database: New Image Database
Actions
indicates a selection is required from the main frame.
Options
Organize
Add image
Edit
Delete
Search
Show all
Manage columns
Image List: New Image Database

	Keywords	Creator	Filename	Title	Description	Thumbnail
	Edit	Edit		Edit		
●	relevant keywords for this image	Jane Smith	image_name.gif	Relevant Title for this Image	View description	

The Edit Record page displays. To change the ***Filename:** field select the **New Image** button. The New Image Screen displays. Type the path and filename in the field or click the browse button and find the file in the My-Files area. Select the Regenerate thumbnail checkbox if you would like the image database to create a new thumbnail for you. Select the **Update** button.



The Edit Record page displays again with the new image filename in the ***Filename:** field. If you did not have the image database regenerate the thumbnail for you on the previous screen, select the **New thumbnail** button. The New Thumbnail page displays. Type the path and filename of the image thumbnail in the **Thumbnail:** field or click the browse button and find the file in the My-Files area. Select the **Update** button. The Edit Record page displays again with the new image filename in the **Thumbnail:** field. Type the corrected information in ***Keywords** field, **Creator:** field, **Title:** field, and/or **Description:** field. Select the **Update** button.


Edit Record

*Keywords: relevant keywords for this image

Creator: Jane Smith

*Filename: image_name.gif

Title: Relevant Title for this Image

Description: This is where the description of the image will go.

Thumbnail:

The Image Database page displays with the new image and/or information.

III. Delete an Image Record

You may find that you no longer want an image to be included in your image database. You can delete images from a image database but they must be deleted one at a time.

Caution: You will not be able to "undo" this process. The image and all the associated data in it will be lost forever if it is deleted. If you are unsure, make a backup of the course before removing the database. See Restoring and Resetting a WebCT course into CE 4.1 for assistance with making a backup of your course.

From the Homepage or the Course Menu select the Image Database link.

The Available Database page displays. Select the link to the image database that contains the image you want to edit. The Image Database page displays. Select the radio button beside the image you would like to delete and select the **Delete** button.

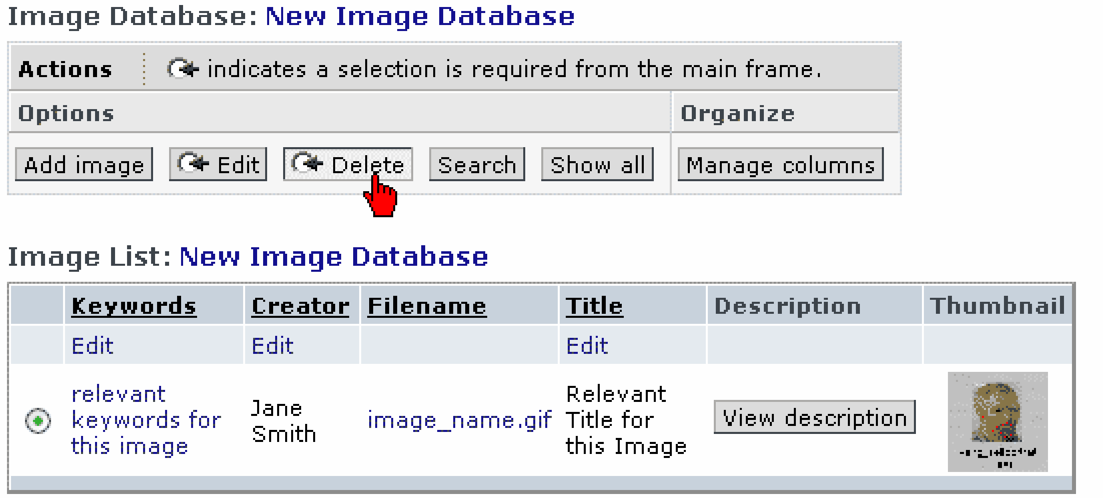


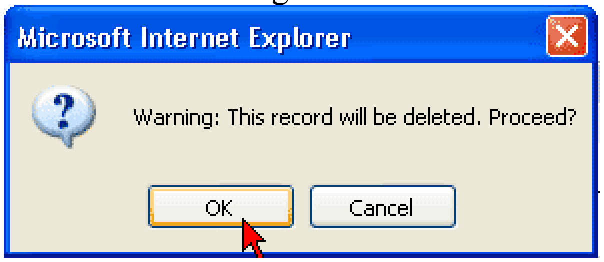
Image Database: New Image Database

Actions	⟳ indicates a selection is required from the main frame.
Options	Organize
Add image · ⟳ Edit · ⟳ Delete · Search · Show all	Manage columns

Image List: New Image Database

	Keywords	Creator	Filename	Title	Description	Thumbnail
	Edit	Edit		Edit		
◉	relevant keywords for this image	Jane Smith	image_name.gif	Relevant Title for this Image	View description	

The Delete Image confirmation window displays. Select **OK** button.



Microsoft Internet Explorer

Warning: This record will be deleted. Proceed?

OK Cancel

The Image Database page displays without the deleted image. To delete additional images from the database repeat the above steps.

Topic – 7: Text Database

A text *database* is a collection of related documents assembled into a single searchable unit. The individual documents can be massive or minuscule, but they should bear some relation to each other.

A database is composed of smaller units called *records*. In a text database, a record can be an entire document, a section within a document, a single page or a fragment of text within a page. When searching a database, one or more records containing information that satisfies the query will be retrieved.

A record can contain smaller regions of data called*fields*. A field usually defines a particular type of data common to several or all records within a database. For instance, in a database of corporate memos, wherein each memo makes up a record, thefollowing fields might be used: *TO, FROM, DATE, SUBJECT* and *TEXT*. The scope of a search can be narrowed by restricting it to one or more fields. Limit the search to the *FROM* field when searching for a sender's name. Only those records with the specified name in that field would be retrieved.

Stopwords

A full-text retrieval software indexes every word in a document, with the exception of *stopwords*. Stopwords are those terms that is programmed to ignore during the indexing and retrieval processes, in order to prevent the retrieval of extraneous records. Generally, a stopword list includes articles, pronouns, adjectives, adverbs and prepositions (*the, they, very, not, of,* etc.) that are most common in the English language.

Example:

PsycCrawler

Relevance Ranking

• The most powerful weapon in the PsycCrawler arsenal is *relevance ranking*. Simply put, relevance ranking arranges a set of retrieved records so that those most likely to be relevant to the request are shown first. That is, after PsycCrawler retrieves all documents that satisfy the search query, it uses relevance ranking to arrange them based on a measurement of similarity between the query and the content of each record. PsycCrawler performs a content analysis of records in the database by using a combination of the following indicators:

• **Breadth of Match**. The more distinct query terms that appear in a document, the higher the weight of relevance.

• **Inverse Document Frequency**. Rare terms (within the entire database) receive a higher weight of relevance.

• **Frequency**. The number of times a query term occurs in a document.

• **Density**. The comparable length of retrieved documents.

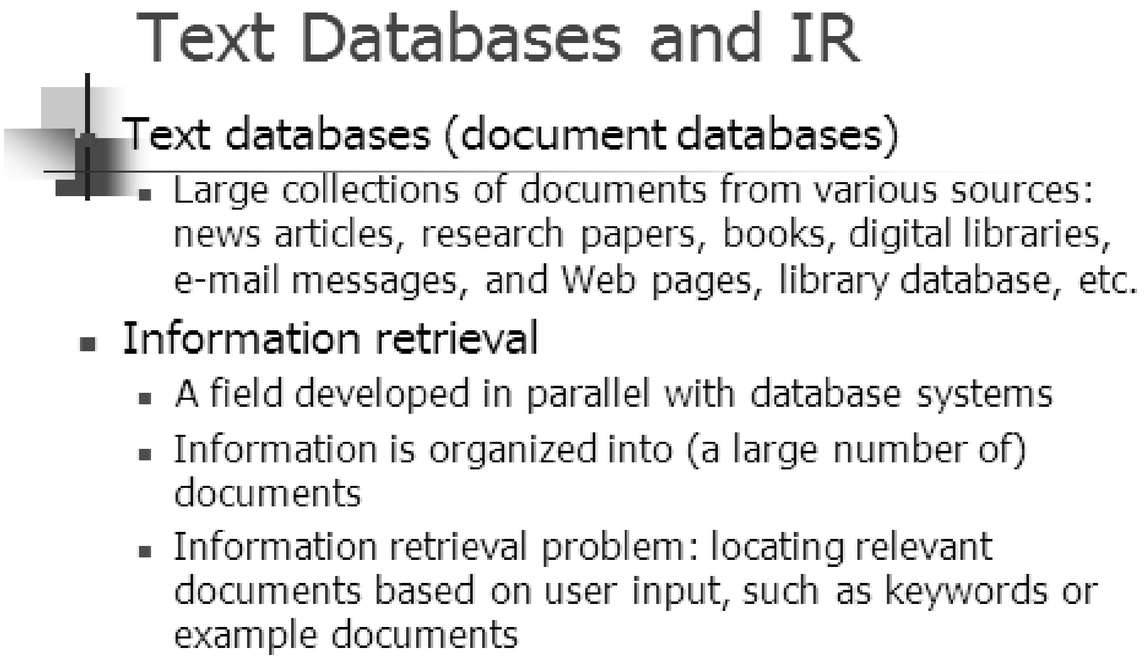
Consideration of these combined criteria produces intelligent on-the-fly evaluation of a record's likelihood of satisfying the intent behind the query.
This allows to find more relevant information with less effort. Regardless of how many records the search query retrieves, it is needed to review relatively few of them, because moving down the ranking means moving toward less relevant records. With relevance ranking, less time is spent in reviewing search results before deciding whether they are satisfactory.
The burden of composing complex logical queries, which are used to reduce the amount of retrieved data to manageable proportions is reduced. It is not necessary to care about how many records are retrieved, as long as the best information floats to the top.

Full Text Database

• A full-text database is a compilation of documents or other information in the form of a database in which the complete text of each referenced document is available for online viewing, printing, or downloading. In addition to text documents, images are often included, such as graphs, maps, photos, and diagrams.
A full-text database is searchable by keyword, phrase, or both.

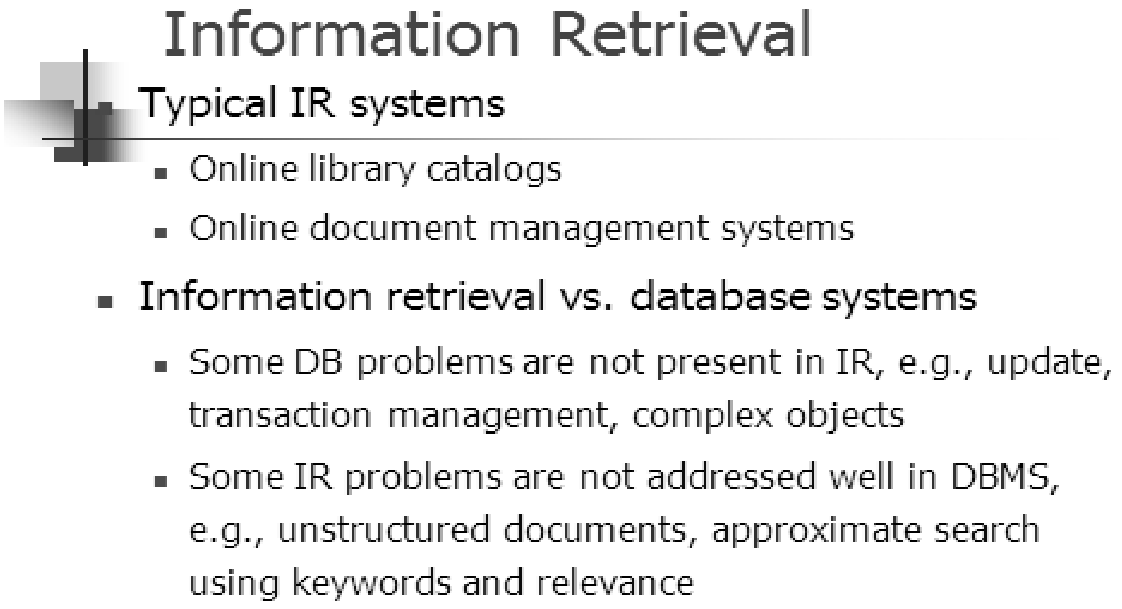
• When an item in a full-text database is viewed, it may appear in ASCII format (as a text file with the .txt extension), as a word-processed file (requiring a program such as Microsoft Word), as an HTML (Web page) file, or as a Portable Document Format (PDF) file. When a document appears as a PDF file, it is usually a scanned hardcopy of the original article, chapter, or book.

• Full-text databases are used by college and university libraries as a convenience to their students and staff. Full-text databases are ideally suited to online courses of study, where the student remains at home and obtains course materials by downloading them from the Internet. Access to these databases is normally restricted to registered personnel or to people who pay a specified fee per viewed item. Fulltext databases are also used by some corporations, law offices, and government agencies.
In the United States, the Internal Revenue Service and most state departments of revenue are good examples.



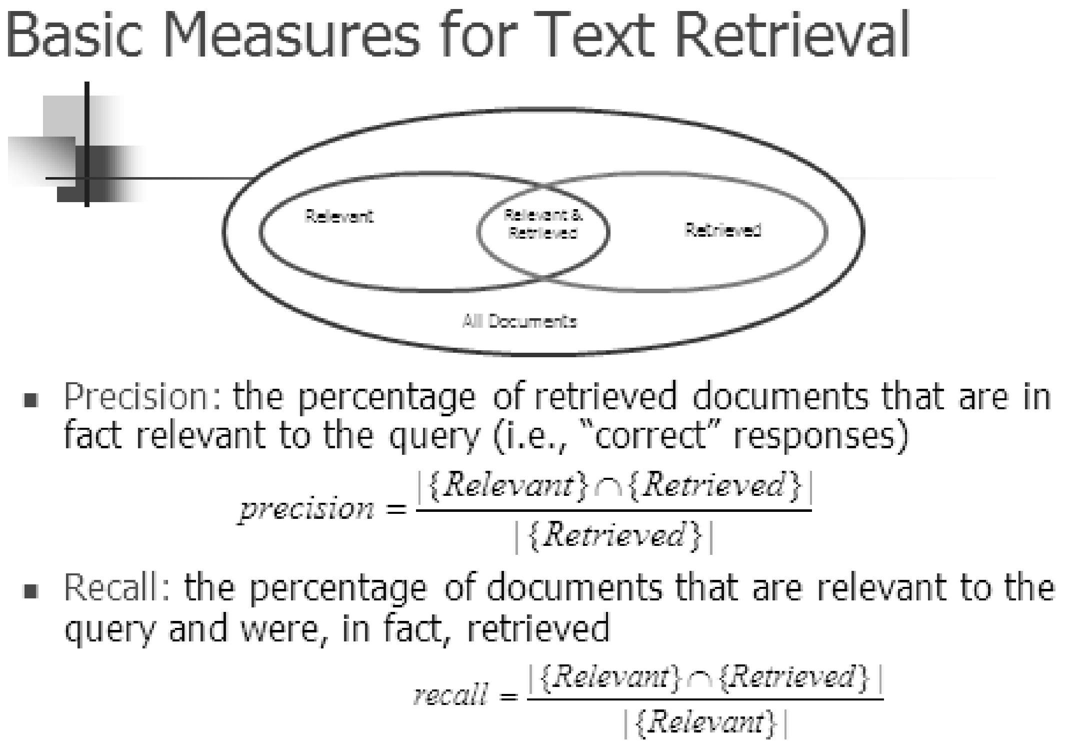
Text Databases and IR

- Text databases (document databases)
  - Large collections of documents from various sources: news articles, research papers, books, digital libraries, e-mail messages, and Web pages, library database, etc.
- Information retrieval
  - A field developed in parallel with database systems
  - Information is organized into (a large number of) documents
  - Information retrieval problem: locating relevant documents based on user input, such as keywords or example documents



Information Retrieval

- Typical IR systems
  - Online library catalogs
  - Online document management systems
- Information retrieval vs. database systems
  - Some DB problems are not present in IR, e.g., update, transaction management, complex objects
  - Some IR problems are not addressed well in DBMS, e.g., unstructured documents, approximate search using keywords and relevance



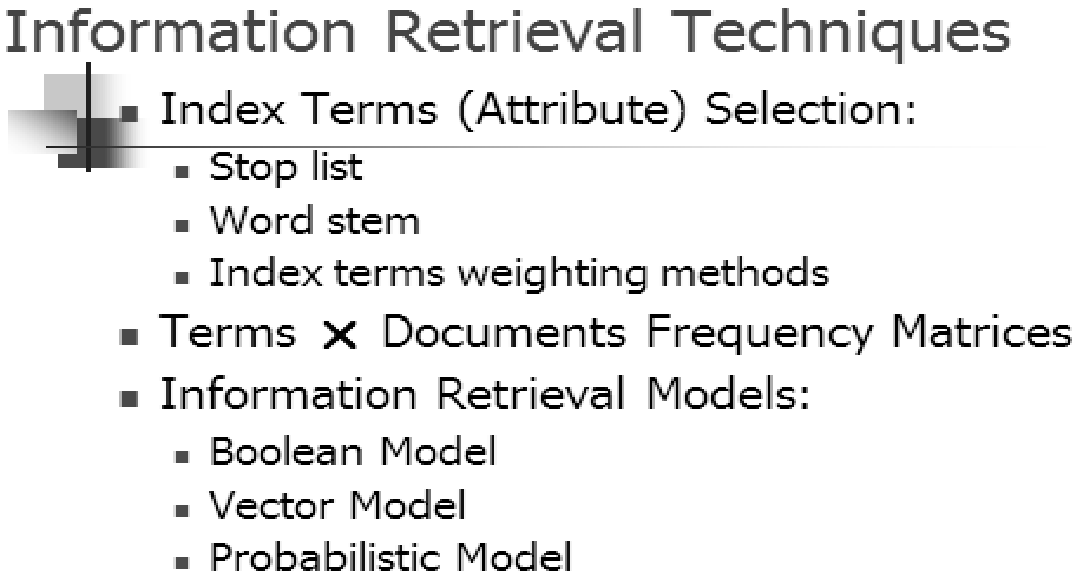
Basic Measures for Text Retrieval

- Precision: the percentage of retrieved documents that are in fact relevant to the query (i.e., “correct” responses)

$$precision = \frac{|\{Relevant\} \cap \{Retrieved\}|}{|\{Retrieved\}|}$$

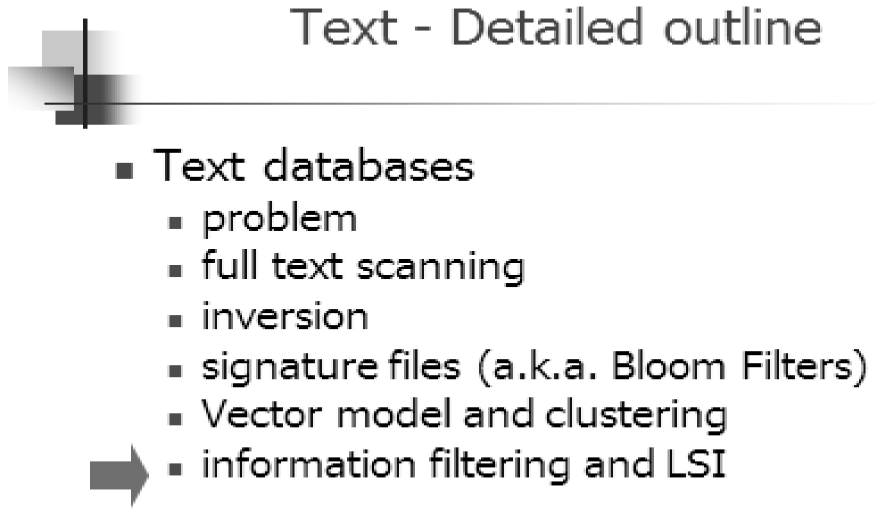
- Recall: the percentage of documents that are relevant to the query and were, in fact, retrieved

$$recall = \frac{|\{Relevant\} \cap \{Retrieved\}|}{|\{Relevant\}|}$$



Information Retrieval Techniques

- Index Terms (Attribute) Selection:
  - Stop list
  - Word stem
  - Index terms weighting methods
- Terms × Documents Frequency Matrices
- Information Retrieval Models:
  - Boolean Model
  - Vector Model
  - Probabilistic Model



Text - Detailed outline

- Text databases
  - problem
  - full text scanning
  - inversion
  - signature files (a.k.a. Bloom Filters)
  - Vector model and clustering
  - information filtering and LSI

Problem - Motivation

- Given a database of documents, find documents containing "*data*", "*retrieval*"
- Applications:
  - Web
  - law + patent offices

  - digital libraries
  - information filtering
- Types of queries:
  - boolean (‘data’ AND ‘retrieval’ AND NOT ...)
  - additional features (‘data’ ADJACENT ‘retrieval’)
  - keyword queries (‘data’, ‘retrieval’)
- How to search a large collection of documents?

Full-text scanning

- for single term:
  - (naive: O(N*M))

ABRACADABRA text

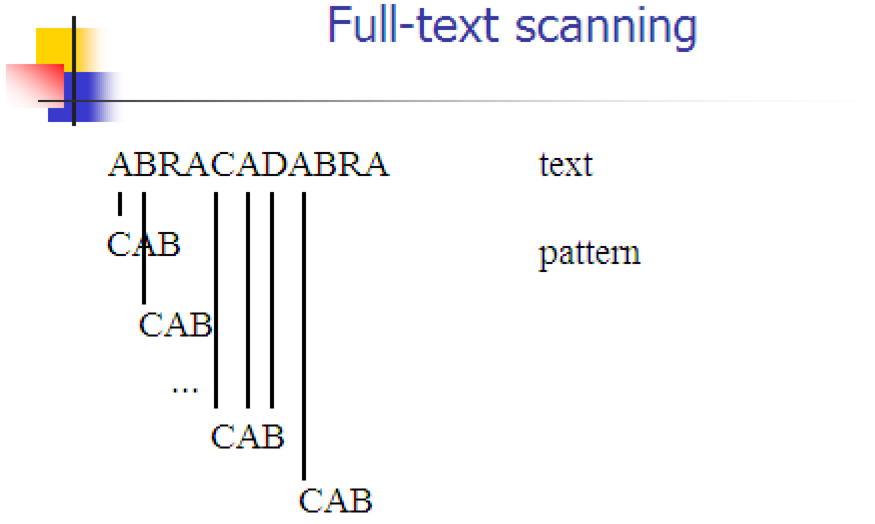
CAB pattern

- for single term:
  - (naive: O(N*M))
  - Knuth, Morris and Pratt (‘77)
    - build a small FSA; visit every text letter once only, by carefully shifting more than one step

text

ABRACADABRA

CAB pattern



Full-text scanning

ABRACADABRA text

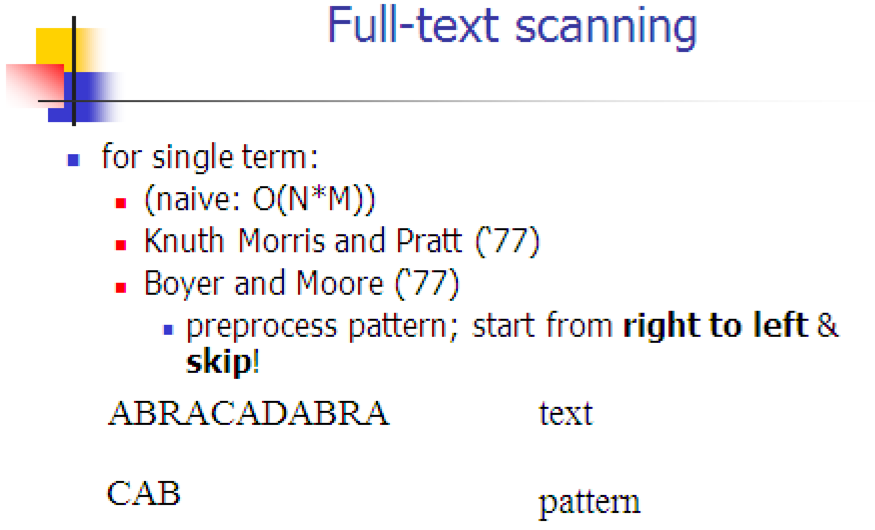
CAB pattern

CAB

...

CAB

CAB



Full-text scanning

- for single term:
  - (naive: O(N*M))
  - Knuth Morris and Pratt ('77)
  - Boyer and Moore ('77)
    - preprocess pattern; start from **right to left** & **skip**!

ABRACADABRA text

CAB pattern

Sample Questions

Topic – 1:

Topic – 2:

Topic – 3:

Topic – 4:

Topic – 5:

Topic – 6:

Topic – 7:

University Questions

1. Explain and highlight the features of Knowledge bases, Parallel databases. (16M)
2. Explain the following:

   a) Image database (8M)

   b) Text database (8M)

************************** End of Unit – V ***********************