Week 1 Lecture 1

Class	BSCCS2001
Created	@August 19, 2021 1:46 PM
Materials	https://drive.google.com/drive/folders/19FhdYYKeH3ZshWhoZIJlP_MC1nVnUUmU?usp=sharing
Module #	1
Type	Lecture
Week #	1

Database Management Systems (DBMS)

DBMS: A database management system (or DBMS) is essentially nothing more than a computerized data-keeping system. (via IBM)

DBMS contains info about a particular enterprise

- Collection of interrelated data
- Set of programs to access the data
- An environment that is both *convenient* and *efficient* to use

Database Applications:

- Banking: transactions
- Airlines: reservations, schedules
- Universities: registration, grades
- Sales: customers, products, purchases
- Online retailers: order tracking, customized recommendations
- Manufacturing: production, inventory, orders, supply chain
- HR: employee records, salaries, tax deductions

Databases can be very large

Databases touch various aspects of our lives

University Database Example

Application program examples

- Add new students, instructors and courses
- Register students for courses and generate class rosters
- Assign grades to students, compute Grade Point Average (GPA) and generate transcripts

In early days, database applications were built directly on top of file systems

Drawbacks of using file systems to store data

- Data redundancy and inconsistency
  - Multiple file formats, duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation
  - Multiple files and formats
- Integrity problems
  - Integrity constraints (eg: account balance > 0) become "buried" in program code rather than being stated explicity
  - Hard to add new constraints or change existing ones
- Atomicity of updates
  - Failures may leave databases in an inconsistent state with partial updates carries out
  - **Example:** Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - **Example:** Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Security problems
  - Hard to provide user access to some, but not all, data

Database systems offer solutions to the above problems

Course pre-requisites:

Set Theory

- Definition of a set
  - Intensional definition
  - Extensional definition
  - Set-builder notation
- Membership, Subset, Superset, Power set, Universal set
- Operations on sets:
  - Unions, Intersections, Complement, Difference, Cartesian product
- De-Morgan's Law

Relations and Functions

- Definition of Relations
- Ordered pairs and Binary relations
  - Domain and Range

    - Image, Pre-image, Inverse
    - Properties: Reflexive, Symmetric, Anti-symmetric, Transitive, Total
- Definition of functions
- Properties of functions: Injective, Surjective, Bijective
- Composition of functions
- Inverse of functions

Propositional Logic

- Truth values and Truth tables
- Operators: conjunction (and), disjunction (or), negation (not), implication, equivalence
- Closure under Operations

Predicate Logic

- Predicates
- Quantification
    - Existential
    - Universal

Python

Algorithms and Programming in C

- Sorting
    - Merge sort
    - Quick sort
- Search
    - Linear search
    - Binary search
    - Interpolation search

Data Structures

- Arrays
- List
- Binary Search Tree
    - Balanced Tree
- B - Tree
- Hash table/map

Object-Oriented analysis and design

- **Refresher material**
    - Discrete Mathematics by Brilliant: https://brilliant.org/wiki/discrete-mathematics
    - Python
        - IITM online book: https://pypod.github.io
        - Cheatsheet: https://www.pythoncheatsheet.org
        - DataCamp Cheatsheet: https://www.datacamp.com/community/tutorials/python-data-science-cheat-sheet-basics

- C Language: https://www.youtube.com/watch?v=zYierUhIFNQ&list=PLhQjrBD2T382_R182iC2gNZI9HzWFMC_8&index=2 (part of CS50 2020 Lectures)

Week 1 Lecture 2

Class	BSCCS2001
Created	@August 19, 2021 3:48 PM
Materials	
Module #	2
Type	Lecture
Week #	1

Why DBMS?

Data Management

- Storage
- Retrieval
- Transaction
- Audit
- Archival

For

- Individuals
- Small / Big Enterprises
- Global

There has been 2 major approaches in this practice:

1. **Physical:**

   Physical Data or Records Management, more formally known as ***Book Keeping,*** has been using physical ledgers and journals for centuries

   The most significant development happened when Henry Brown patented a "receptacle for storing and preserving papers" on November 2, 1886

   Herman Hollerith adapted the punch cards used for weaving looms to act as the memory for a mechanical tabulating machine in 1890

2. **Electronic:**

   Electronic Data or Records management moves with the advances in technology, especially of memory, storage, computing and networking

   - 1950s: Computer programming started
   - 1960s: Data Management with punch cards / tapes and magnetic tapes
   - 1970s:
     - COBOL and CODASYL approach was introduced in 1971
     - On October 14, 1979, Apple II platform shipped VisiCalc, marking the birth of spreadsheets
     - Magnetic disks became prevelant
     - 1980s: RDBMS changed the face of data management
     - 1990s: With internet, data management started becoming global
     - 2000s: e-Commerce boomed, NoSQL was introduced for unstructured data management
     - 2010s: Data Science started riding high

Electronic Data Management Params

Electronic Data or Records management depends on various params including ...

- Durability
- Scalability
- Security
- Retrieval
- Ease of Use
- Consistency
- Efficiency
- Cost

Book Keeping

A book register was maintained on which the shop owner wrote the amount received from customers, the amount due for any customer, inventory details and so on ...

Problems with such an approach of book keeping:

- **Durability:** Physical damage to these registers is a possibility due to rodents, humidity, wear and tear
- **Scalability:** Very difficult to maintain over the years, some shops have numerous registers spanning over the years
- **Security:** Susceptible to tampering by the outsiders
- **Retrieval:** Time consuming process to search for previous entry
- **Consistency:** Prone to human errors

Not only small shops but large orgs also used to maintain their transactions in book registers

Spreadsheet files - A better solution

Mostly useful for single user or small enterprise applications

Spreadsheet software like Google Sheets: Due to disadvantages of maintaining ledger registers, organizations dealing with huge amount of data shifted to using spreadsheets for maintaining records in files

- **Durability:** These are computer applications and hence data is less prone to physical damage
- **Scalability:** Easier to search, insert and modify records as compared to book ledgers
- **Security:** Can be password protected
- **Easy to Use:** Computer applications are used to search and manipulate records in the spreadsheets leading to reduction in manpower needed to perform routing computations

- **Consistency:** Not guaranteed but spreadsheets are less prone to mistakes registers

Why leave filesystems?

Lack of efficiency in meeting growing needs

- With rapid scale up of data, there has been considerable increase in the time required to perform most operations
- A typical spreadsheet file may have an upper limit on the number of rows
- Ensuring consistency of data is a big challenge
- No means to check violations of constraints in the face of concurrent processing
- Unable to give different permissions to different people in a centralized manner
- A system crash could be catastrophic

The above mentioned limitations of filesystems paved the way for a comprehensive platform dedicated to management of data - the **Database Management System**

History of Database Systems

- 1950s and early 1960s
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s
  - Hard disks allowed direct access to data
  - Network and hierarchical data model in widespread use
  - **Ted Codd** defines the relational data model
    - Would win the ACM Turin Award for his work
    - IBM Research begins in System R prototype
    - UC Berkeley begins Ingres prototype
  - High-performance (for the era) transaction processing
- 1980s
  - Research relational prototypes evolve into commercial systems - SQL becomes industrial standard
  - Parallel and distributed database systems
  - Object oriented database systems
- 1990s
  - Large decision support and data mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce
- Early 2000s
  - XML and XQuery standards
  - Automated database administration
- Later 2000s
  - Giant data storage systems - Google BigTable, Yahoo PNuts, Amazon, ...

Week 1 Lecture 3

Class	BSCCS2001
Created	@August 19, 2021 4:47 PM
Materials	
Module #	3
Type	Lecture
Week #	1

Why DBMS? (part 2)

Case study of a Bank Transaction

Consider a simple banking system where a person can open a bank account, transfer funds to an existing account and check the history of all her transactions till date

The application performs the following checks

- If the account balance is not enough, it will now allow the fund transfer
- If the account numbers are not correct, it will flash a message and terminate the transaction
- If a transaction is successful, it prints a confirmation message

We will use this banking transaction system to compare various features of a file-based (.csv file) implementation viz-a-viz a DBMS-based implementation

- Account details are stored in
    - Accounts.csv for file-based implementation
    - Accounts table for DBMS implementation
- The transaction details are stored in
    - Ledger.csv for file-based implementation
    - Ledger table for DBMS implementation

Source: https://github.com/bhaskariitm/transition-from-files-to-db

Initiating a transaction

Python

```
def begin_Transaction(credit_account, debit_account, amount):
    temp = []
    success = 0

    # Open file handles to retrieve and store transaction data
    f_obj_Account1 = open('Accounts.csv', 'r')
    f_reader1 = csv.DictReader(f_obj_Account1)
    f_obj_Account2 = open('Accounts.csv', 'r')
    f_reader2 = csv.DictReader(f_obj_Account2)
    f_obj_Ledger = open('Ledger.csv', 'a+')
    f_writer = csv.DictWriter(f_obj_Ledger, fieldnames=col_name_Ledger)
```

SQL

```
-- Handled implicitly by the DBMS
```

Transaction

Python

```
try:
  for sRec in f_reader1:
    # CONDITION CHECK FOR ENOUGH BALANCE
    if sRec['AcctNo'] == debitAcc and int(sRec['Balance']) > int(amt):
      for rRec in f_reader2:
        if rRec['AcctNo'] == creditAcc:
          sRec['Balance'] = str(int(sRec['Balance']) - int(amt))    # DEBIT
          temp.append(sRec)
          # CRITICAL POINT
          f_writer.writerow({
            'Acct1':sRec['AcctNo'],
            'Acct2':rRec['AcctNo'],
            'Amount':amt,
            'D/C':'D'
          })
          rRec['Balance'] = str(int(rRec['Balance']) + int(amt))    # CREDIT
          temp.append(rRec)
          f_writer.writerow({'Account1': r_record['Account_no'], 'Account2': s_record['Account_no'], 'Amount': amount,'D/C': 'C'})
          success = success + 1
          break
        f_obj_Account1.seek(0)
        next(f_obj_Account1)
        for record in f_reader1:
            if record['Account_no'] != temp[0]['Account_no'] and record['Account_no'] != temp[1]['Account_no']:
                temp.append(record)
    except:
        print('\nWrong input entered !!!')
```

SQL

```
do $$
begin
amt = 5000
sendVal = '1800090';
recVal = '1800100';
select balance from accounts
into sbalance
where account_no = sendVal;
if sbalance < amt then
raise notice "Insufficient balance";
else
update accounts
  set balance = balance - amt
  where account_no = sendVal;
insert into ledger(sendAc, recAc, amnt, ttype)
values(sendVal, recVal, amt, 'D')
update accounts
  set balance = balance + amt
  where account_no = recVal;
insert into ledger(sendAc, recAc, amnt, ttype)
values(sendVal, recVal, amt, 'C')
commit;
raise notice "Successful";
end if;
end; $$
```

Closing a transaction

Python

```
f_obj_Account1.close()
f_obj_Account2.close()
f_obj_Ledger.close()
if success == 1:
    f_obj_Account = open('Accounts.csv', 'w+', newline='')
    f_writer = csv.DictWriter(f_obj_Account, fieldnames=col_name_Account)
    f_writer.writeheader()
    for data in temp:
        f_writer.writerow(data)
    f_obj_Account.close()
    print("\nTransaction is successfull !!")
else:
    print('\nTransaction failed : Confirm Account details')
```

SQL

```
-- Handled implicitly by the DBMS
```

Comparison

Parameter	File handling via Python	DBMS
Scalability with respect to amount of data	Very difficult to handle insert, update and querying of records	In-built features to provide high scalability for a large number of records
Scalability with respect to changes in structure	Extremely difficult to change the structure of records as in the case of adding or removing attributes	Adding or removing attributes can be done seamlessly using simple SQL queries
Time of execution	in seconds	in milliseconds
Persistence	Data processed using temporary data structures have to be manually updated to the file	Data persistence is ensured via automatic, system induced mechanisms
Robustness	Ensuring robustness of data has to be done manually	Backup, recovery and restore need minimum manual intervention
Security	Difficult to implement in Python (Security at OS level)	User-specific access at database level
Programmer's productivity	Most file access operations involve extensive coding to ensure persistence, robustness and security of data	Standard and simple built-in queries reduce the effort involved in coding thereby increasing a programmer's throughput
Arithmetic operations	Easy to do arithmetic computations	Limited set of arithmetic operations are available
Costs	Low costs for hardware, software and human resources	High costs of hardware, software and human resources

Parameterized Comparison

Scalability

File Handling in Python

- **Number of records:** As the # of records increases, the efficiency of flat files reduces:
    - the time spent in searching for the right records
    - the limitations of the OS in handling huge files
- **Structural Change:** To add an attribute, initializing the new attribute of each record with a default value has to be done by program. It is very difficult to detect and maintain relationships between entities if and when an attribute has to be removed

DBMS

- **Number of records:** Databases are built to efficiently scale up when the # of records increase drastically.
    - In-built mechanisms, like indexing, for quick access of right data

- **Structural Changes:** During adding an attribute, a default value can be defined that holds for all existing records - the new attribute gets initialized with default value. During deletion, constraints are used either not to allow the removal on ensure its safe removal

Time and Efficiency

- If the number of records is very small, the overhead in installing and configuring a database will be much more than the time advantage obtained from executing the queries
- However, in the number of records is really large, then the time required in the initialization process of a database will be negligible as compared to that of using SQL queries

File Handling in Python

- The effort needed to implement a file handler is quite less in Python
- In order to process a 1GB file, a program in Python would typically take a few seconds

DBMS

- The effort to install and configure a DB in a DB server in expensive and time consuming
- In order to process a 1GB file, an SQL query would typically take a few milliseconds

Programmer's Productivity

File Handling in Python

- **Building a file handler:** Since the constraints within and across entities have to be enforced manually, the effort involved in building a file handling application is huge
- **Maintenance:** To maintain the consistency of data, one must regularly check for sanity of data and the relationships between entities during inserts, updates and deletes
- **Handling huge data:** As the data grows beyond the capacity of the file handler, more efforts are needed

DBMS

- **Configuring the database:** The installation and configuration of a database is a specialized job of a DBA. A programmer, on the other hand, is saved the trouble
- **Maintenance:** DBMS has built-in mechanisms to ensure consistency and sanity of data being inserted, updated or deleted. The programmer does not need to do such checks
- **Handling huge data:** DBMS can handle even terabytes of data - Programmer does not have to worry

Arithmetic Operations

File Handling in Python

- Extensive support for arithmetic and logical operations on data using Python. These include complex numerical calculations and recursive computations

DBMS

- SQL provides limited support for arithmetic and logical operations. Any complex computation has to be done outside of SQL

Costs and Complexity

File Handling in Python

- File systems are cheaper to install and use. No specialized hardware, software or personnel are required to maintain filesystems

DBMS

- Large databases are served by dedicated database servers which need large storage and processing power
- DBMSs are expensive software that have to be installed and regularly updated
- Databases are inherently complex and need specialized people to work on it - like DBA (Database System Administrator)
- The above factors lead to huge costs in implementing and maintaining database management systems

Week 1 Lecture 4

Class	BSCCS2001
Created	@August 19, 2021 5:55 PM
Materials	
Module #	4
Type	Lecture
Week #	1

Introduction to DBMS

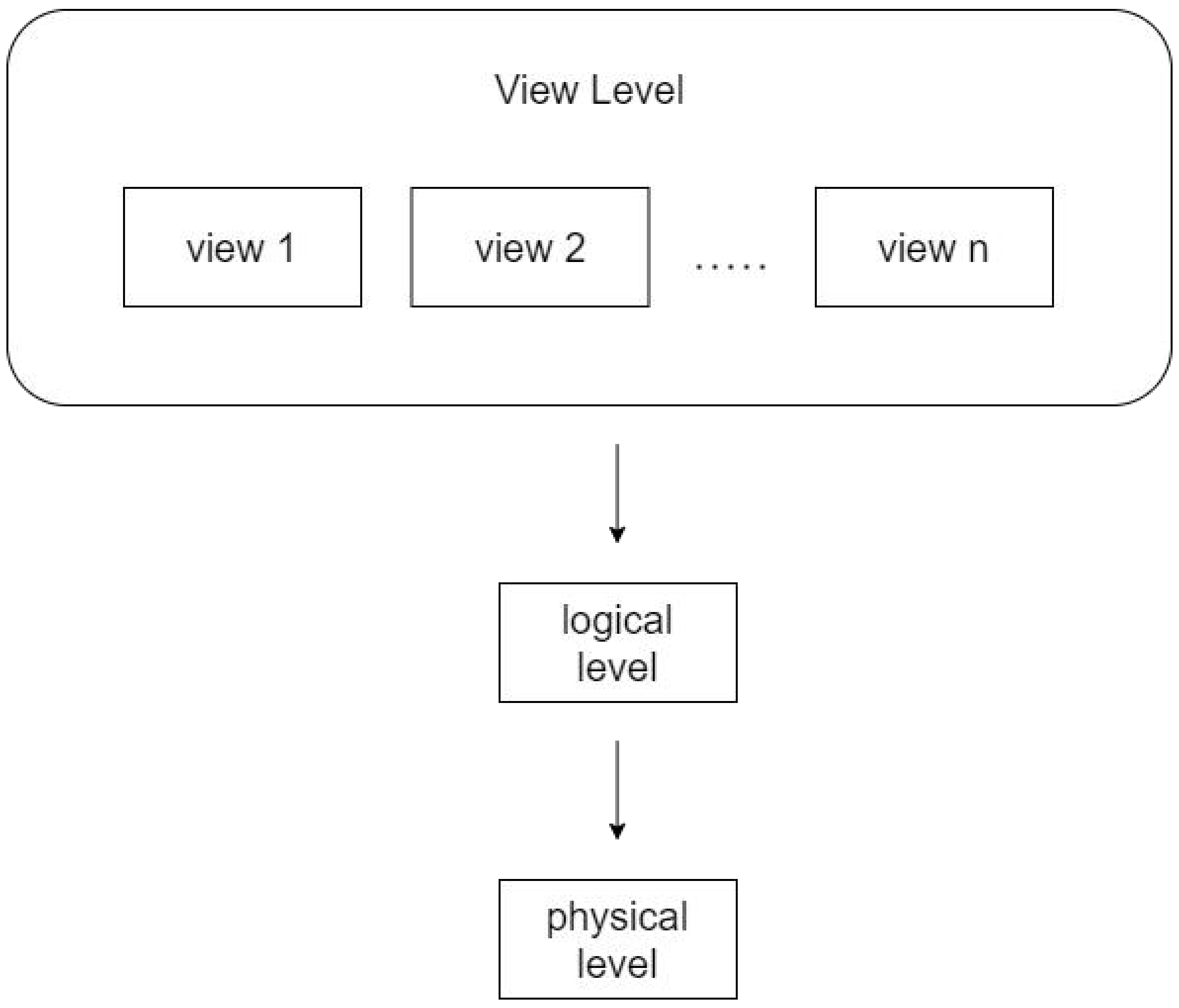
Levels of Abstraction

- **Physical Level:** describes how a record (eg: instructor) is stored
- **Logical Level:** describes data stored in a database and the relationships among the data fields

```
type instructor = record
  ID: string;
  name: string;
  dept_name: string;
  salary: integer;
end;
```

- **View Level:** application programs hide details of data types
  - Views can also hide information (such as employee's salary) for security purposes

An architecture for a database system



Schema and Instances

TLDR: *Schema* is the way in which data is organized and *Instance* is the actual value of the data

- **Schema**
  - **Logical Schema** - the overall logical structure of the database
    - Analogous to type information of a variable in a program (eg: int x = 5)
    - *Example:* The database consists of information about a set of customers and accounts in a bank and the relationship between them

    Customer Schema

Name	Customer ID	Account #	Aadhaar ID	Mobile #
Untitled				

    Account Schema

Account #	Account Type	Interest Rate	Min. Bal.	Balance
Untitled				

  - **Physical Schema** - the overall physical structure of the database

- **Instance**
  - The actual content of the database at a particular point in time
  - Analogous to the value of a variable

Customer Instance

Name	Customer ID	Account #	Aadhaar ID	Mobile #
Pavan Lakha	6728	917322	182719289372	9830100291
Lata Kala	8912	827183	918291204829	7189203928
Nand Prabhu	6617	372912	127837291021	8892021892

Account Instance

Account #	Account Type	Interest Rate	Min. Bal.	Balance

Account #	Account Type	Interest Rate	Min. Bal.	Balance
917322	Savings	4.0%	5000	7812
372912	Current	0.0%	0	291820
827183	Term Deposit	6.75%	10000	100000

- **Physical Data Independence** - the ability to modify the physical schema without changing the logical schema
  - Analogous to independence of *Interface* and *Implementation* in object-oriented systems
  - Applications depend on the logical schema
  - In general, the interfaces between various levels and components should be well defined so that changes in some parts do not seriously influence others.

Data Models

- A collection of tools that describe the following ...
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- **Relational model** (our focus in this course)
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Other older models
  - Network model
  - Hierarchical model
- Recent models for Semi-structured or Unstructured data
  - Converted to easily manageable formats
  - Content Addressable Storage (CAS) with metadata descriptors
  - XML format
  - RDBMS which support BLOBs

Relational Model

- All the data is stored in various tables
  - Tables are also called Relations
  - Columns are called attributes
  - They have particular names which tells us the schema
  - Rows are records that are the values

Data Definition Language (DDL)

- Specification notation for defining the database schema
  - Example

```
create table instructor (
  ID char(5),
  name varchar(20),
  dept_name varchar(20),
  salary numeric(8, 2))
```

- DDL compiler generates a set of table templates stored in a *data dictionary*
- Data dictionary contains metadata (that is, data about the data)
  - Database schema

- Integrity constraints
  - Primary key (ID uniquely identifies instructors)
- Authorization
  - Who can access what

Data Manipulation Language (DML)

Language for accessing and manipulating the data organized by the appropriate data model

- **DML:** also know as *Query Language*
- Two classes of languages
  - **Pure** - used for proving properties about computational power and for optimization
    - *Relational Algebra* (our focus in this course)
    - Tuple relational calculus
    - Domain relational calculus
  - **Commercial** - used in commercial systems
    - SQL is the most widely used commercial language

Structured Query Language (SQL)

- Most widely used commercial language
- ***SQL is NOT a Turing Machine equivalent language***. Read more here
  - Cannot be used to solve all problems that a C program, for example, can solve
- To be able to compute complex complex functions, SQL is usually embedded in some higher-level language
- Application programs generally access databases through one of ...
  - Language extensions to allow embedded SQL
  - Application Programming Interfaces or APIs (eg: ODBC / JDBC) which allow SQL queries to be sent to the databases

Database Design

The process of designing the general structure of the database:

- **Logical Design** - Deciding on the database schema. Database design requires that we find a good collection of relation schema
  - Business decision
    - What attributes should we record in the databases?
  - Computer Science decision
    - What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- **Physical Design** - Deciding on the physical layout of the database

Week 1 Lecture 5

Class	BSCCS2001
Created	@August 20, 2021 11:13 AM
Materials	
Module #	5
Type	Lecture
Week #	1

Introduction to DBMS (part 2)

Database Design

Design Approaches

- Need to come up with a methodology to ensure that each relation in the database is ***good***
- Two ways of doing so:
    - Entity Relationship Model *(primarily tries to capture the business requirements)*
        - Models an enterprise as a collection of entities and relationships
        - Represented diagrammatically by an entity-relationship diagram
    - Normalization Theory *(this is the Computer Science perspective)*
        - Formalize what designs are bad and test for them

Object-Relational Data Models

- Relational model: flat, atomic values
- Object Relational Data Models
    - Extend the relational data model by including object orientation and constructs to deal with added data types
    - Allow attributes of tuples to have complex types, including non-atomic values such as nested relations
    - Preserve relational foundations, in particular the declarative access to data, while extending modeling power
    - Provide upward compatibility with existing relational language

XML: eXtensible Markup Language

- Defined by the ***WWW Consortium (W3C)***
- What XML primarily says; *XML is a description of name-value pair*
  - It talks about a tag, so you can put a value on that
- Originally intended as a document markup language not a database language
- The ability to specify new tags and to create tag structures made XML a great way to exchange data, not just documents
- XML has become the basis for all new generation data interchange formats
- A wide variety of tools are available for parsing, browsing and querying XML documents

Database Engine

3 major components are:

- Storage Manager
- Query processing
- Transaction Manager

Storage Management

Storage Manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system

- The storage manager is responsible for the following tasks:
  - Interaction with the OS file manager
  - Efficient storing, retrieving and updating of data
- Issues:
  - Storage access
  - File organization
  - Indexing and hashing

Query Processing

- Parsing and Translation
- Optimization
- Evaluation

How a query is processed?

- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- Need to estimate the cost of operations
  - Depends critically on statistical information about relations which the database must maintain
  - Need to estimate statistics for intermediate results to compute cost of complex expressions

Transaction Management

- What is the system fails?
- What if more than one user is concurrently updating the same file?
- A **transaction** is a collection of operations that perform single logical function in a database application
- **Transaction-Management component** ensure that the database remains in a consistent (correct) state despite system failures (eg: power failures and operating system crashes) and transaction failures

- **Concurrency-control manager** controls the interaction among the concurrent transactions to ensure consistency of the database

Database Architecture

The architecture of a database system is greatly influenced by the underlying computer system on which the database is running:

- Centralized
- Client-Server
- Parallel (multi-processor)
- Distributed
- Cloud