1. DDBS Introduction

Chapter 1

Introduction

Outline

- What is a distributed database system (DDBS)?
- Promises of DDBSs
- Complicating Factors
- Problem Areas

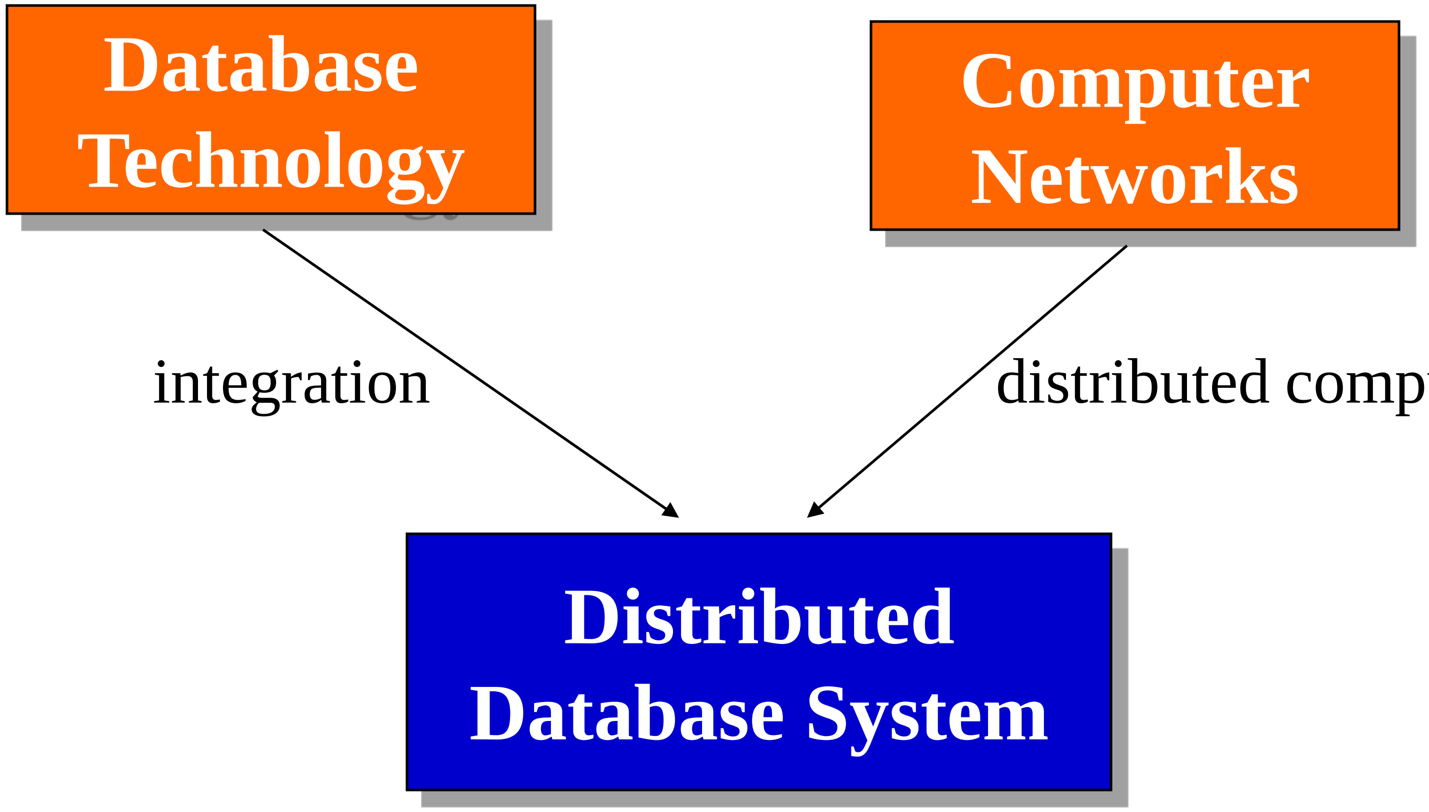
Motivating Example

- Multinational manufacturing company:
  - headquarters in New York
  - manufacturing plants in Chicago and Montreal
  - warehouses in Phoenix and Edmonton
  - R&D facilities in San Francisco
- Data and Information:
  - employee records (working location)
  - projects (R&D)
  - engineering data (manufacturing plants, R&D)
  - inventory (manufacturing, warehouse)

Features

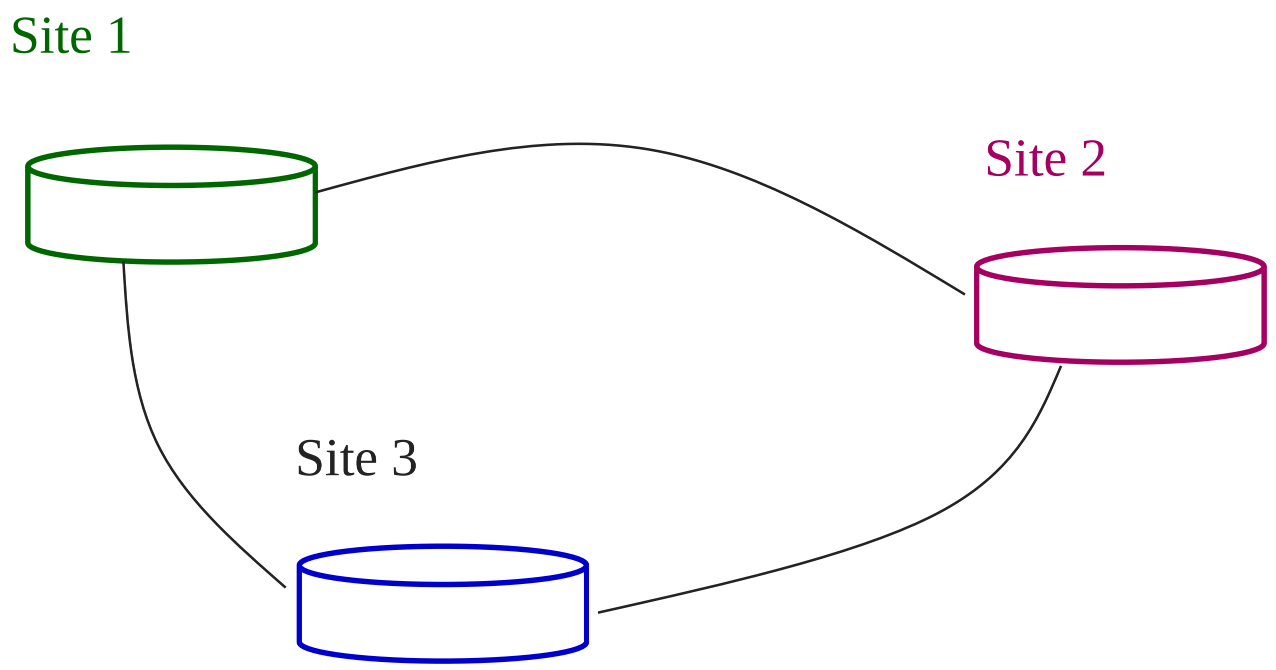
- Data are distributed over sites (e.g. employee, inventory)
- Queries (e.g. “get employees who are younger than 45”) involve more than one site.

Distributed Database System Technology



- The key is integration, not centralization
- Distributed database technology attempts to achieve integration without centralization.

Database Distributed at 3 Sites



DDBS = Database + Network

- Distributed database system technology is the union of what appear to be diametrically opposed approaches to data processing
  - DDBS = database + network
  - Database integrates operational data of an enterprise to provide a centralized and controlled access to that data.
  - Computer network promotes a work mode that goes against all centralization efforts and facilitates distributed computing.

Distributed Computing

- A distributed computing system consists of
  - a number of autonomous processing elements (not necessarily homogeneous), which
    - are interconnected by a computer network
    - cooperate in performing their assigned tasks

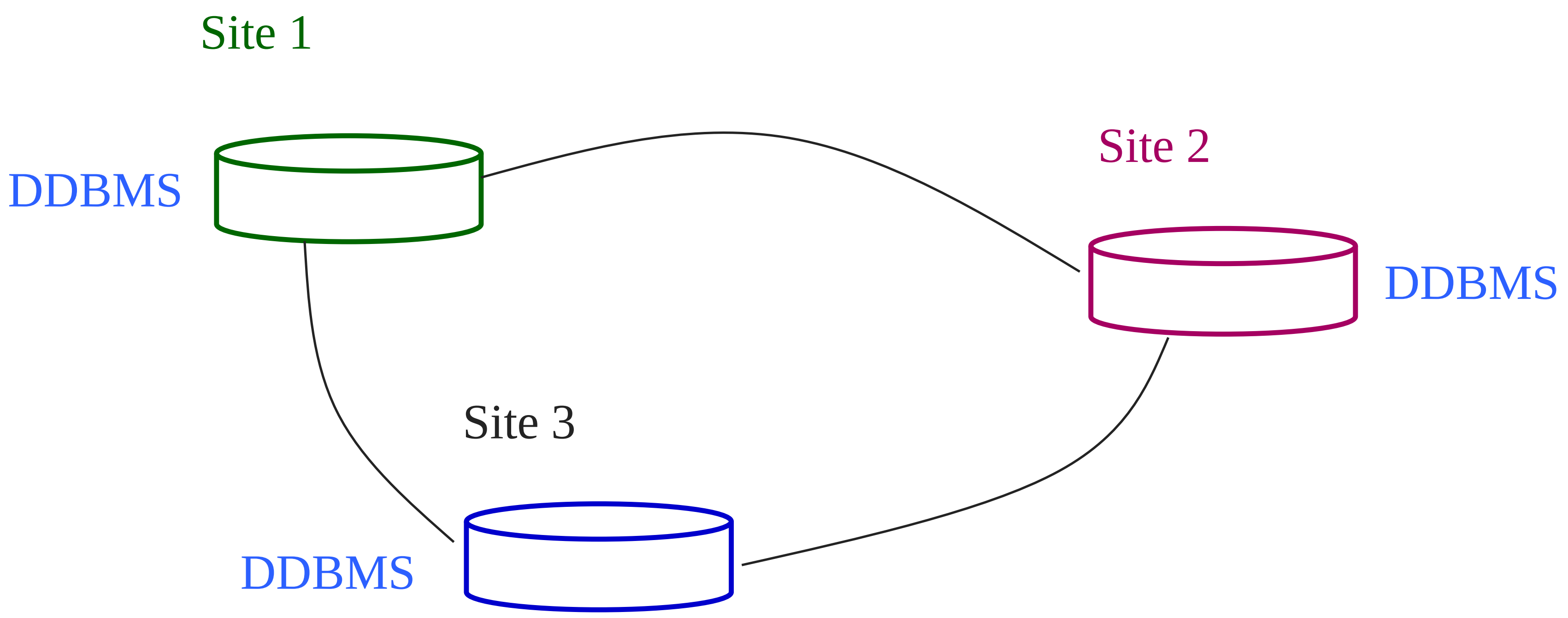
- What is distributed?
  - Processing Logic
  - Function
  - Data
  - Control

All these are necessary and important for distributed database technology.

What is a Distributed Database System?

- A distributed database (DDB) is a collection of multiple, *logically interrelated* databases, distributed over a computer network
  - i.e., storing data on multiple computers (nodes) over the network
- A distributed database management system (DDBMS) is the software that
  - manages the DDB;
  - provides an access mechanism that makes this distribution transparent to the users.
- Distributed database system (DDBS):
  - **DDBS = DDB + DDBMS**

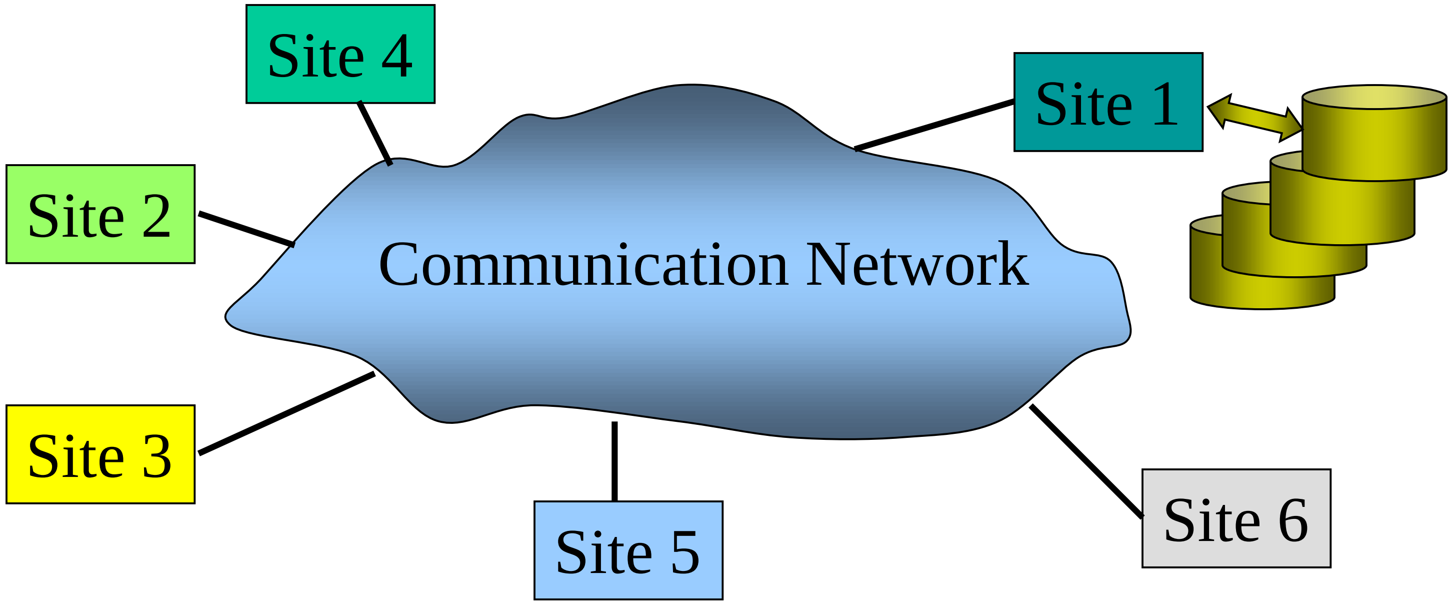
Database Distributed at 3 Sites



What is not a DDBS?

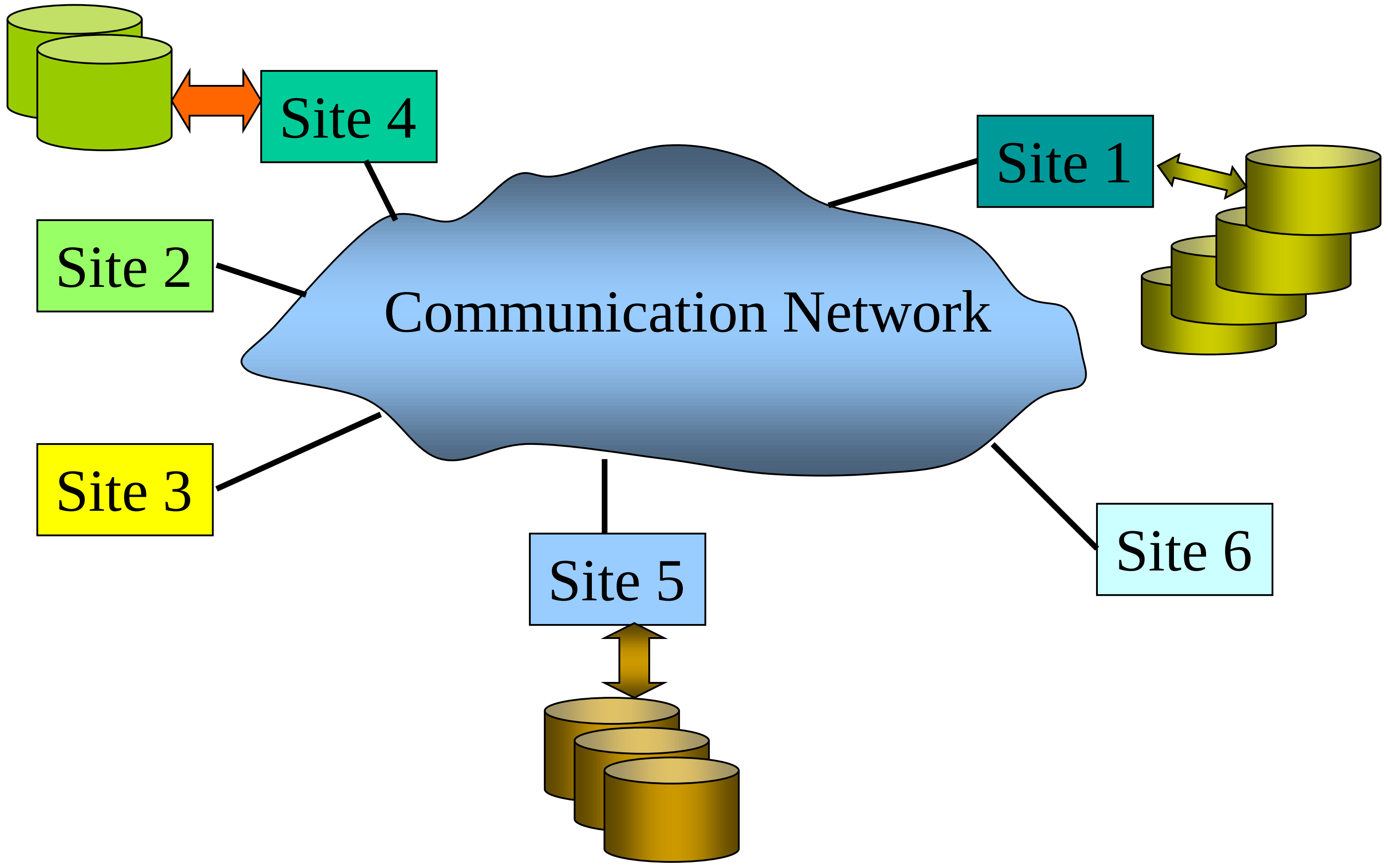
- A timesharing computing system
- A loosely or tightly coupled multiprocessor system
- A database system which resides at one of the nodes of a network of computers
  - This is a centralized database on a network node

Centralized DBMS on a Network



- Data resides only at one node.
- Database management is the same as in a centralized DBMS.
- Remote processing, single-server multiple-clients

Distributed DBMS Environment

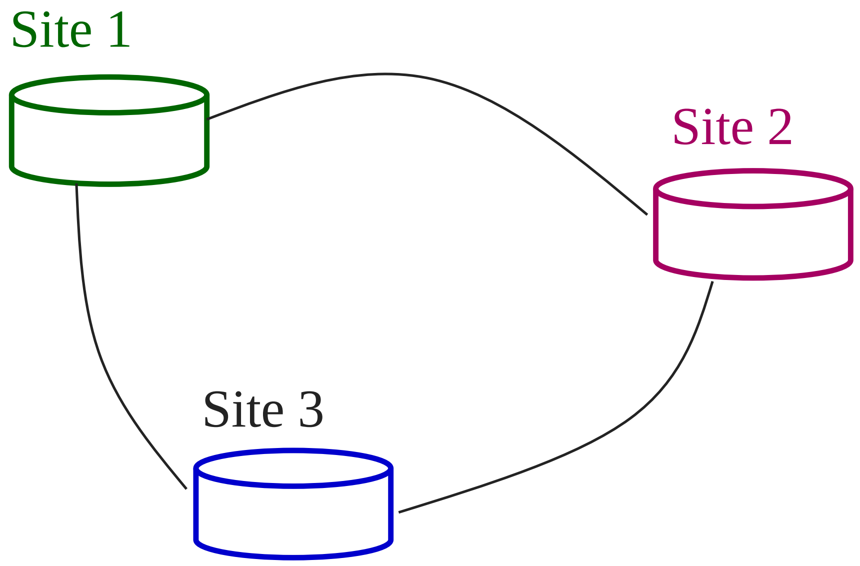


Applications of DDBMS

- Multi-plant manufacturing
- Military command and control
- Airlines
- Hotel chains
- Any organization which has a decentralized organization structure
- Big data era

Why DDBS?

- Several factors leading to the development of DDBS
  - Distributed nature of some database applications
  - Increased reliability and availability
  - Allowing data sharing while maintaining some measure of local control
  - Improved performance
  - Achieve scalability



Implicit Assumptions

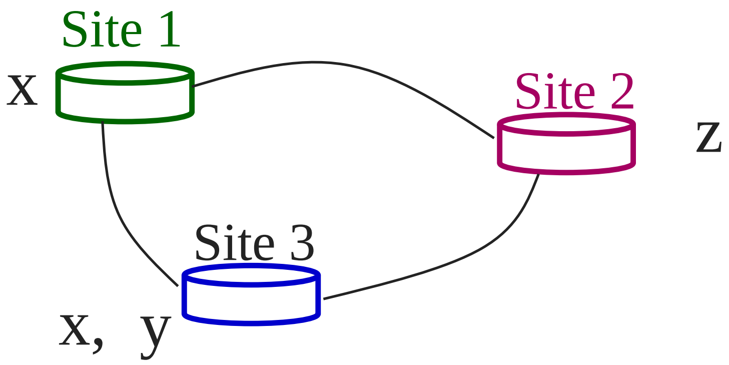
- Data stored at a number of sites
  - Each site has processing power
- Processors at different sites are interconnected by a computer network
- Distributed database is a database, not a collection of files
  - Data logically related as exhibited in the users' access patterns (e.g., relational data model)
- DDBMS is a fully-fledged DBMS
  - Not remote file systems

Design Issues of Distributed Systems

- Transparent data distribution
- Reliable
  - Design should not require the simultaneous functioning of a substantial number of critical components
  - More redundancy, greater availability, and greater consistency
  - Fault tolerance, the ability to mask failures from the user
- Good performance
  - Important (the rest are useless without this)
  - Balance number of messages and grain size of distributed computations
- Scalable
  - A maximum for developing distributed systems
  - Avoid centralized components, tables, and algorithms
  - Only decentralized algorithms should be used

Promises of DDBSs

- Transparent management of distributed, partitioned/fragmented, and replicated data
- Improved reliability and availability through distributed transactions
- Improved performance
- Easier and more economical system expansion



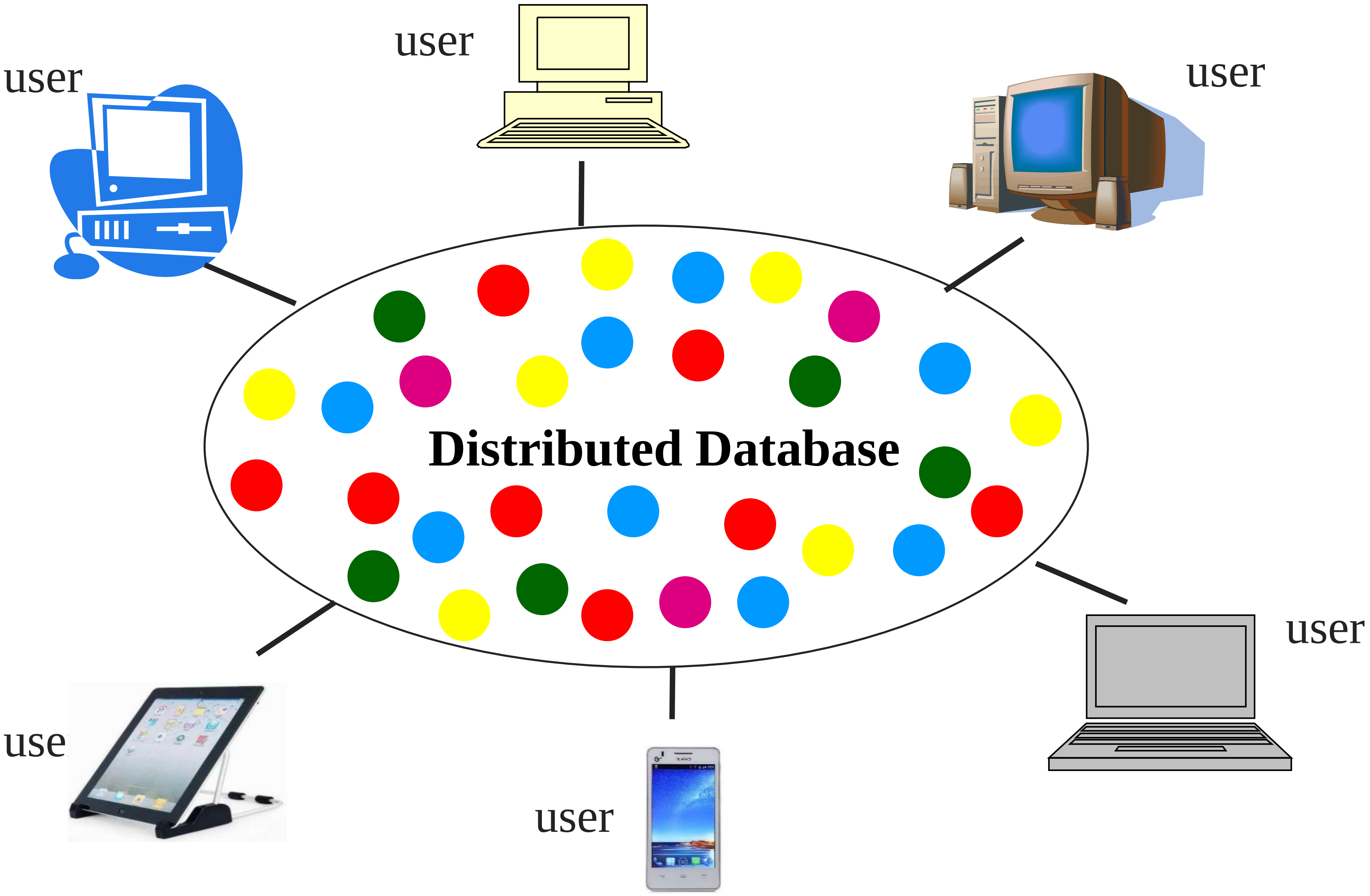
Transparency

- Transparency refers to separation of the high-level semantics of a system from low-level implementation details.
- Fundamental issue is to provide data independence in the distributed environment
  - Network (distribution) transparency
  - Replication transparency
  - Fragmentation transparency
    - Horizontal fragmentation: selection
    - Vertical fragmentation: projection
    - hybrid

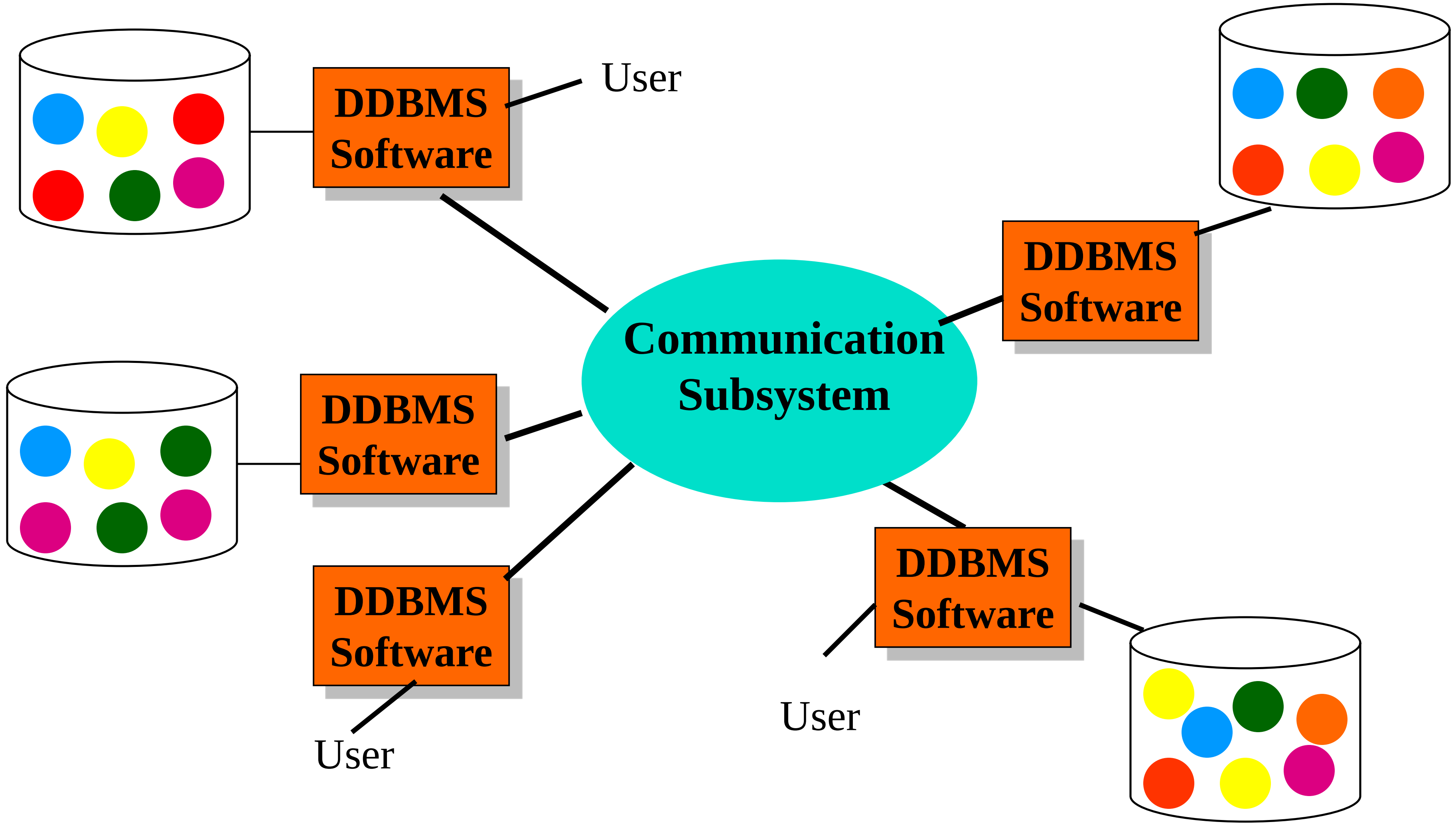
EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

Distributed Database – User View



Distributed DBMS – Reality



Example Relations

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Programmer	48
E4	P2	Manager	18
E5	P2	Manager	24
E6	P4	Engineer	48
E7	P3	Engineer	36
E7	P5	Engineer	23
E8	P3	Manager	40

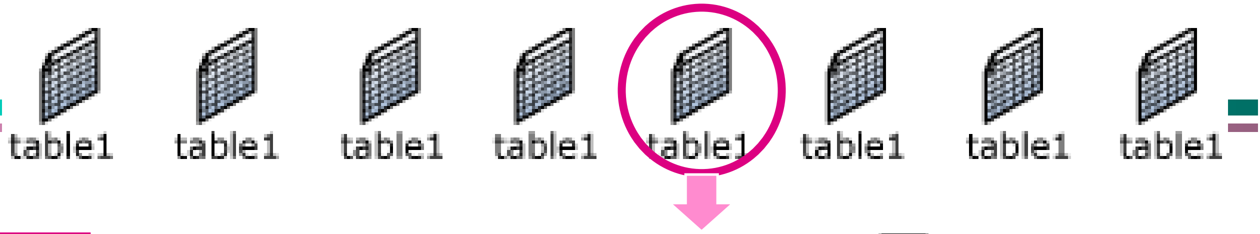
PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY

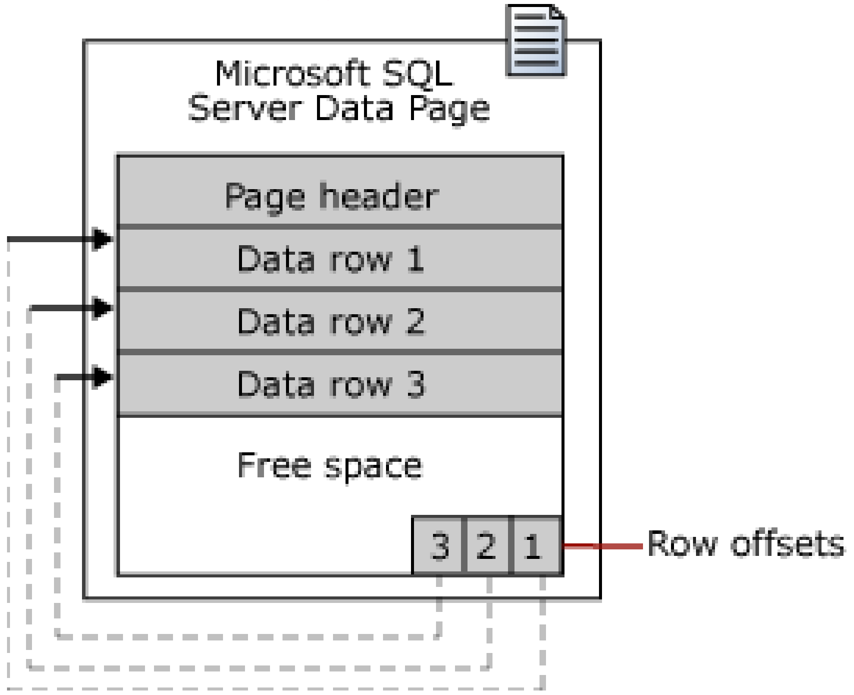
TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

Review: Physical Storage of a Table



EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

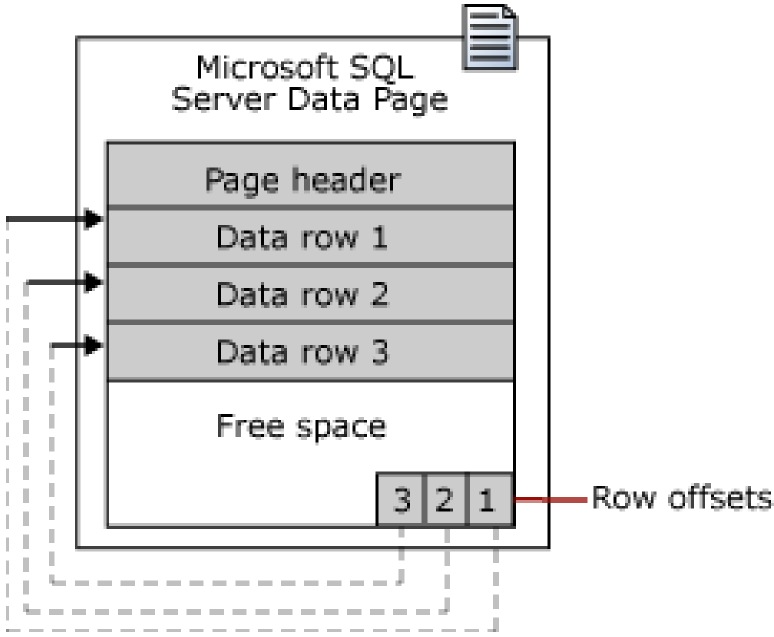


1 Page （8K = 8192 Bytes**)**
1 MB needs 128 data pages

- Page is the most basic unit of data storage. Disk I/O operations for DB read/write are performed in units of pages

A DB page contains three parts

- Page Hearder (96 bytes)
  - Records page ID, page type, and available storage, etc.
- Data Rows (8060 bytes)
- Row Offsets (36 bytes)
  - Record the distance between the first byte of each row and the top of the page (i.e. 96 bytes)

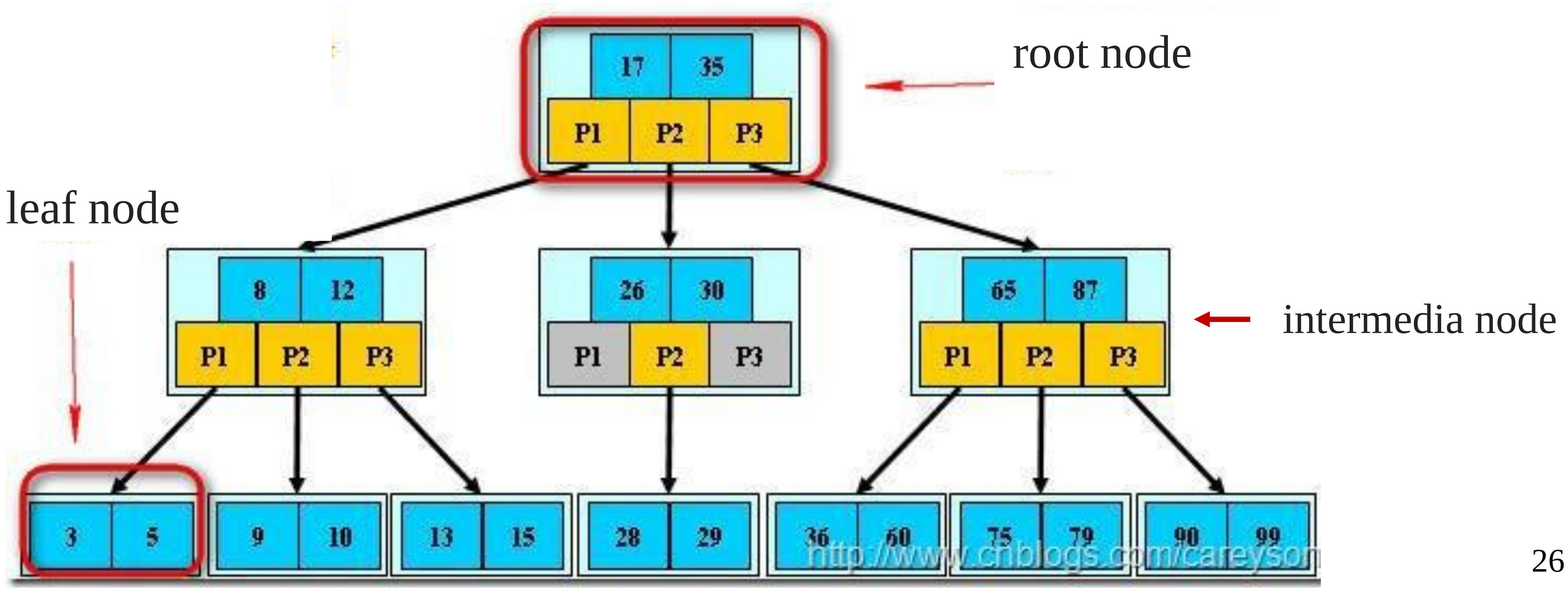


Page Types

- Data Type： Data Page
  - Not including text、ntext、image、nvarchar(max)、varchar(max)、varbinary(max), XML data
- Text/Image Type： Big Data Page
  - text、ntext、image、nvarchar(max)、varchar(max)、varbinary(max), XMLdata
  - When data size is over 8 KB, data type is of variable length
- Index Type： Index Page
- …

Index Page

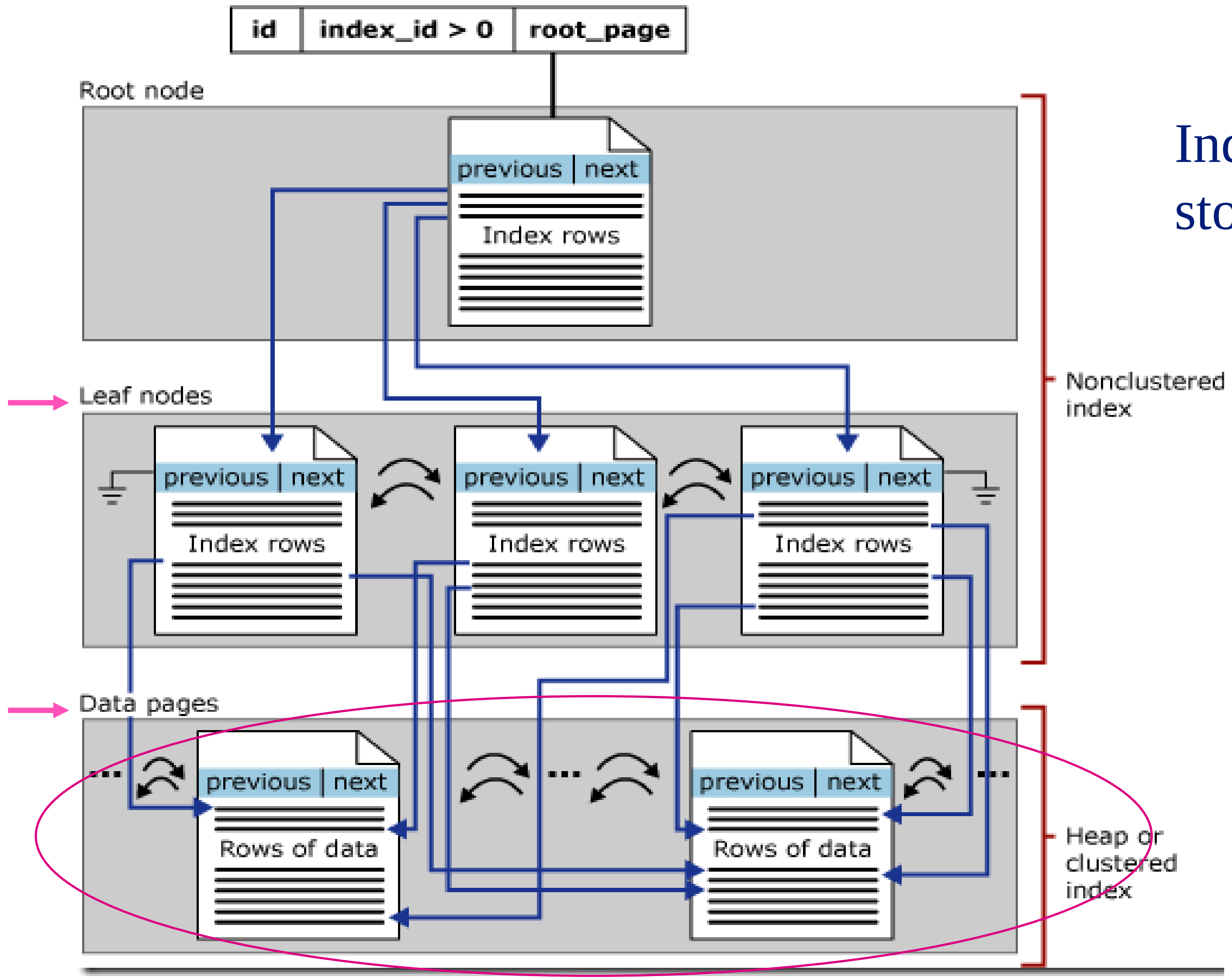
- An index is a structure that sorts the values of one or more columns in a database table
- Use indexes to quickly access specific information in database tables
- B-tree Index Structure



Two Index Types

- Clustered Index
  - Leaf nodes of B tress directly store cluster-indexed data rows (i.e., Data Page)
  - Each DB table can have only one clustered index.
- Non-Clustered Index
  - Leaf nodes of B tress store the data reference, which can be either the clustered index page ID or row ID (if no clustered index exists for the data)

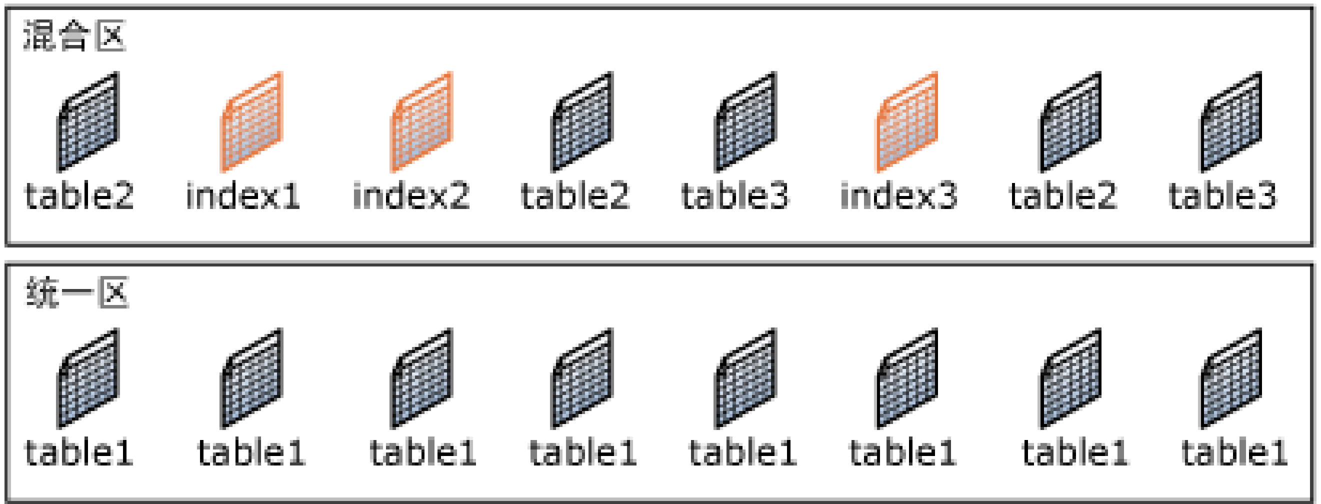
Non-Clustered and Clustered Index



Indexes are also stored in pages.

Extent

- Extent (basic unit in space management)
  - An extent contains 8 physically continuous pags（i.e., 8 KB * 8 = 64 KB）
  - Uniform Extent (统一区)：used by one object
  - Mixed Extent (混合区)：used by maximally 8 different objects

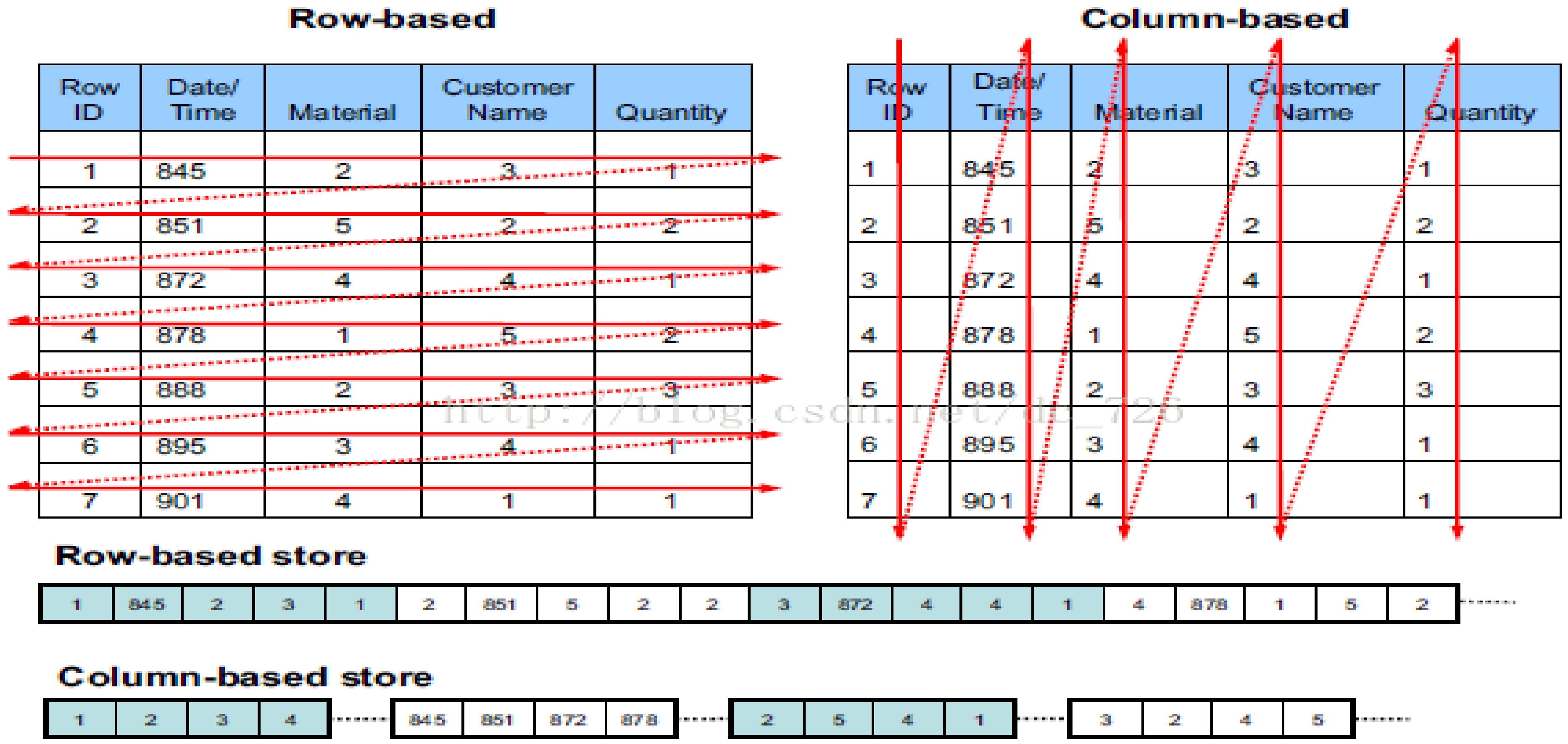


Row vs. Column Storage

- Row storage
  - store a table in a sequence of rows（e.g., Oracle、DB2、MySQL、SQL Server, etc.）

- Column storage
  - store a table in a sequence of columns（e.g., HBase、HP Vertica、EMC Green plum, etc.）

Row vs. Column Storage



Row-based

Row ID	Date/ Time	Material	Customer Name	Quantity
1	845	2	3	1
2	851	5	2	2
3	872	4	4	1
4	878	1	5	2
5	888	2	3	3
6	895	3	4	1
7	901	4	1	1

Column-based

Row ID	Date/ Time	Material	Customer Name	Quantity
1	845	2	3	1
2	851	5	2	2
3	872	4	4	1
4	878	1	5	2
5	888	2	3	3
6	895	3	4	1
7	901	4	1	1

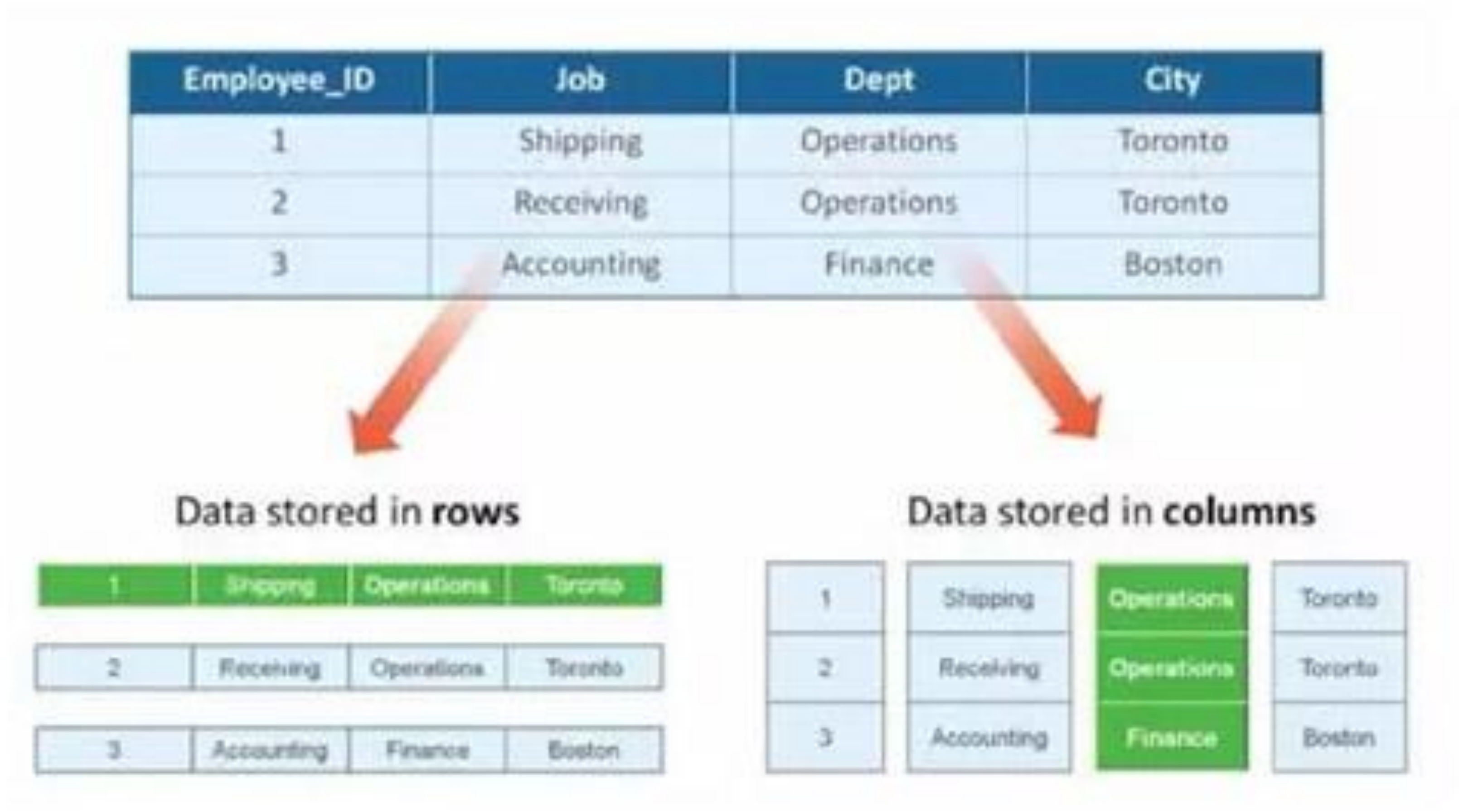
Row-based store

1	845	2	3	1	2	851	5	2	2	3	872	4	4	1	4	878	1	5	2

Column-based store

1	2	3	4	845	851	872	878	2	5	4	1	3	2	4	5

Row vs. Column Storage



Employee_ID	Job	Dept	City
1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston

Data stored in **rows**

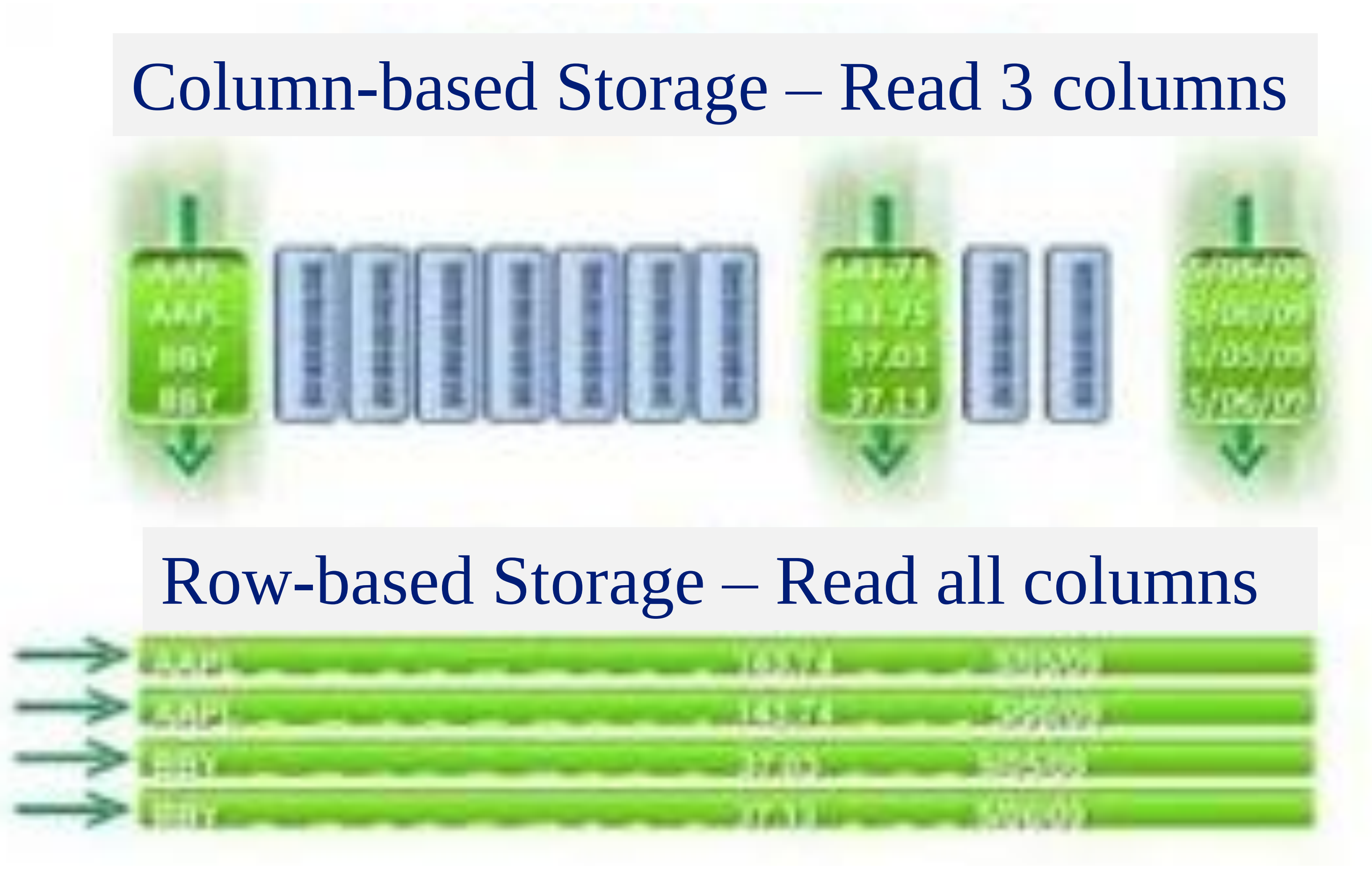
1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston

Data stored in **columns**

1	Shipping	Operations	Toronto
2	Receiving	Operations	Toronto
3	Accounting	Finance	Boston

Operations on Row vs. Column Storage

Column-based Storage – Read 3 columns



Row-based Storage – Read all columns

About Row Storage

- Advantages
  - Write can be completed at one time, consuming less time than column storage
  - Ensure data integrity
- Disadvantages
  - Reading brings many irrelevant data to the memory.
    If there is only a small amount of data, the impact can be ignored; the large amount of data will negatively affect the efficiency of data processing

About Column Storage

- Disadvantages
  - Writing efficiency and ensuring data integrity are not as good as row storage
- Advantages
  - Reading brings relevant data to the memory, which is quite important for big data processing fields with low data integrity requirements, such as the Internet.
  - Read operation on each column can be parallel executed, and the complete record set can be aggregated in memory to reduce the query response time;

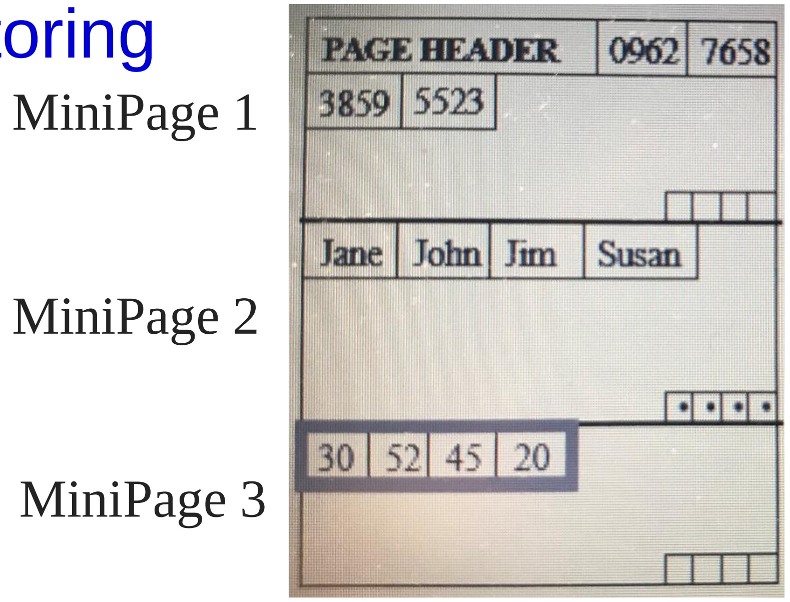
About Column Storage (cont.)

- Because each column is stored independently and the data type is known, the compression algorithm can be dynamically selected according to the data type and data size of the column to improve the utilization of physical storage;
- If a certain column has empty data, there is no need to make space for this column, saving more space than row storage.

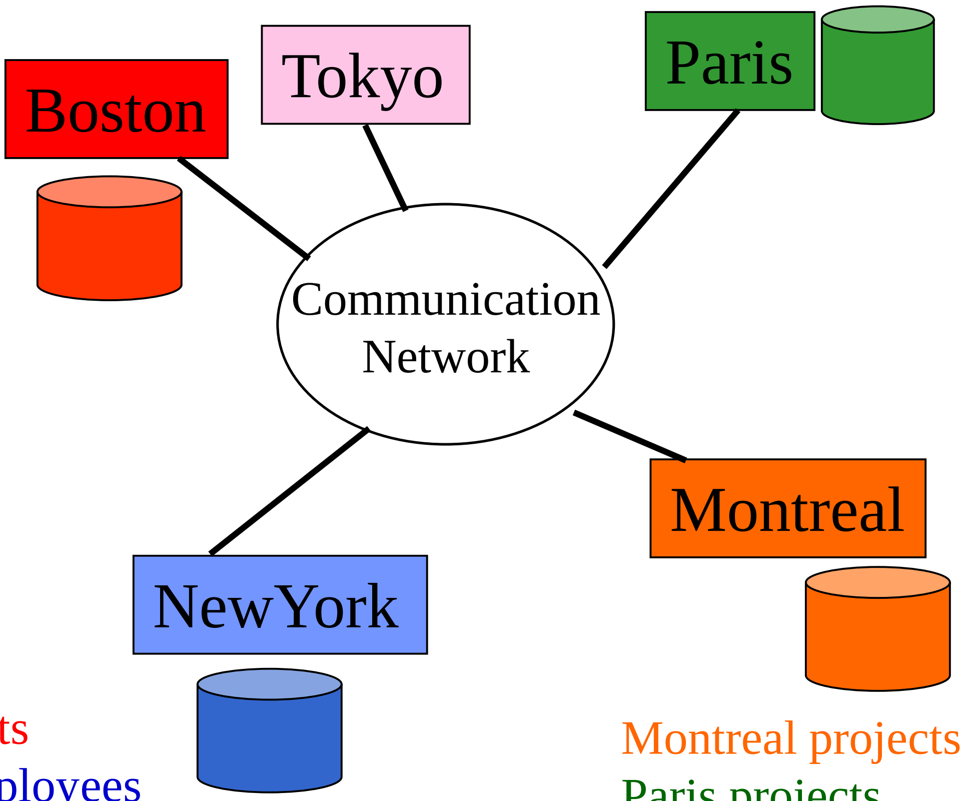
Mixture of Row and Column Storage

- Overcoming the limitations of row and column storage
- All attribute values of the same tuple are stored in one page
- Tuples are vertically divided in the page.
- M attributes divide a page into M MiniPage, each of which corresponds to an attribute, storing all its values.

SID	NAME	Age
0962	Jane	30
7658	Jone	52
3859	Jim	45
5523	Susan	20



Transparent Access Example



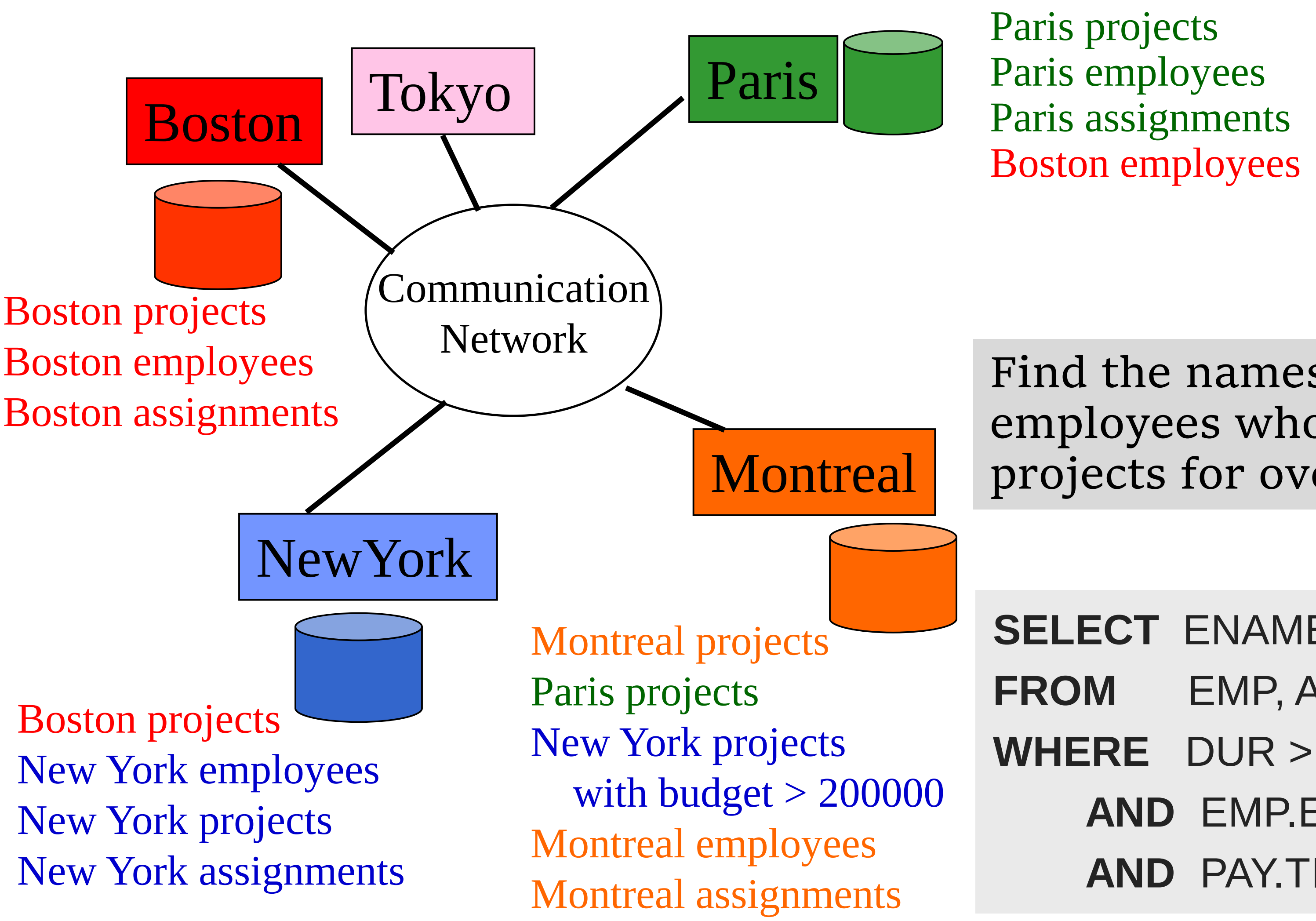
Boston projects
Boston employees
Boston assignments

Paris projects
Paris employees
Paris assignments
Boston employees

Boston projects
New York employees
New York projects
New York assignments

Montreal projects
Paris projects
New York projects with budget > 200000
Montreal employees
Montreal assignments

Transparent Access Example



Find the names and salaries of employees who are assigned to projects for over 12 weeks.

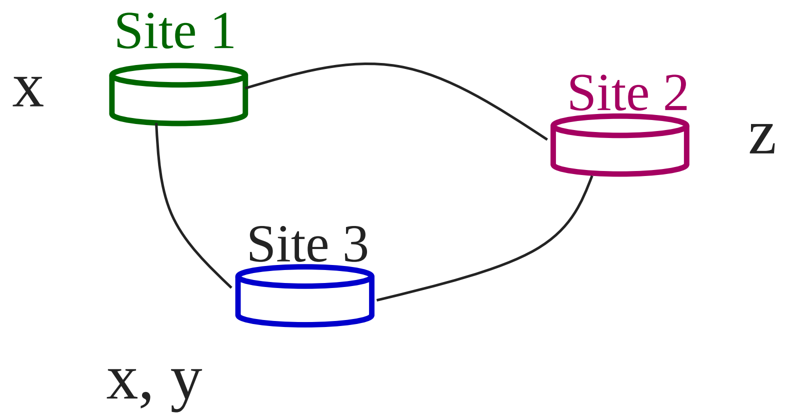
```
SELECT  ENAME, SAL
FROM    EMP, ASG, PAY
WHERE   DUR > 12
    AND  EMP.ENO = ASG.ENO
    AND  PAY.TITLE = EMP.TITLE
```

Improved Performance

- Parallelism in execution
  - inter-query parallelism
  - intra-query parallelism
- Since each site handles only a portion of a database, the contention for CPU and I/O resources is not that severe.
- Data localization reduces communication overheads.
  - Proximity of data to its points of use requires some support from fragmentation and replication

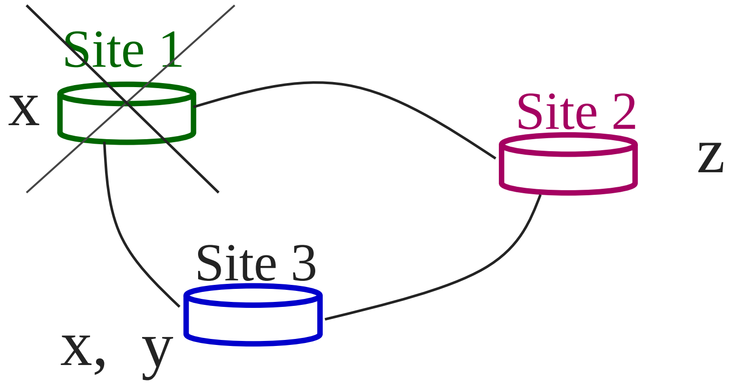
Parallelism Requirements

- Have as much of the data required by *each* application at the site where the application executes
  - Full replication
- How about updates?
  - Updates to replicated data requires implementation of distributed concurrency control and commit protocol

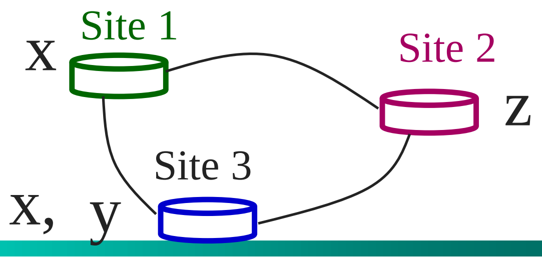


Improved Reliability

- Distributed DBMS can use replicated components to eliminate single point failure.
- Users can still access part of the distributed database with “proper care” even though some of the data is unreachable.
- Distributed transactions facilitate maintenance of consistent database state even when failures occur.

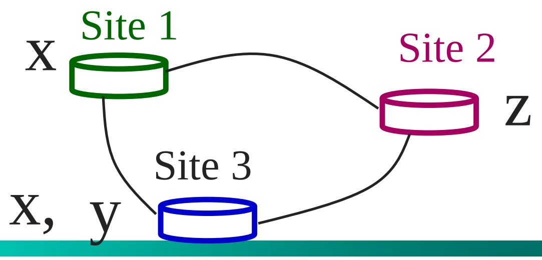


Easier System Expansion



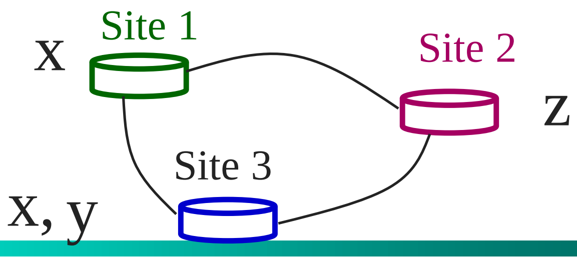
- Ability to add new sites, data, and users over time without major restructuring.
- Huge centralized database systems (mainframes) are history (almost!).
- Cloud computing will lead to natural distributed processing.
- Some applications (such as large enterprise, social network data) are naturally distributed - centralized systems will just not work.

Disadvantages of DDBSs



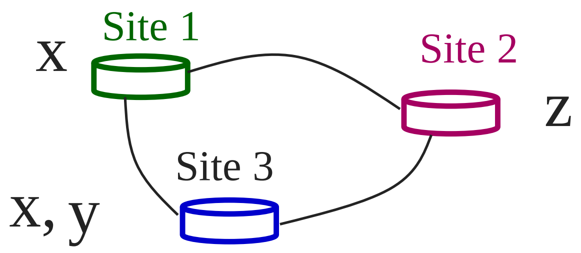
- Complexity
  - DDBS problems are inherently more complex than centralized DBMS ones
- Cost
  - More hardware, software, and people costs
- Distribution of control
  - Problems of synchronization and coordination to maintain data consistency
- Security
  - Database security + network security
- Difficult to convert
  - No tools to convert centralized DBMSs to DDBSs

Complicating Factors



- Data may be replicated in a distributed environment, consequently the DDBS is responsible for
    - choosing one of the stored copies of the requested data for access in case of retrievals
    - making sure that the effect of an update is reflected on each and every copy of that data item
- If there is a site/link failure while an update is being executed, the DDBS must make sure that the effect will be reflected on the data residing at the failing or unreachable sites as soon as the system recovers from the failure.

Complicating Factors



- Maintaining consistency of distributed/replicated data.
- Since each site cannot have instantaneous information on the actions currently carried out in other sites, the synchronization of transactions at multiple sites is harder than a centralized system.

Distributed vs. Centralized DBS

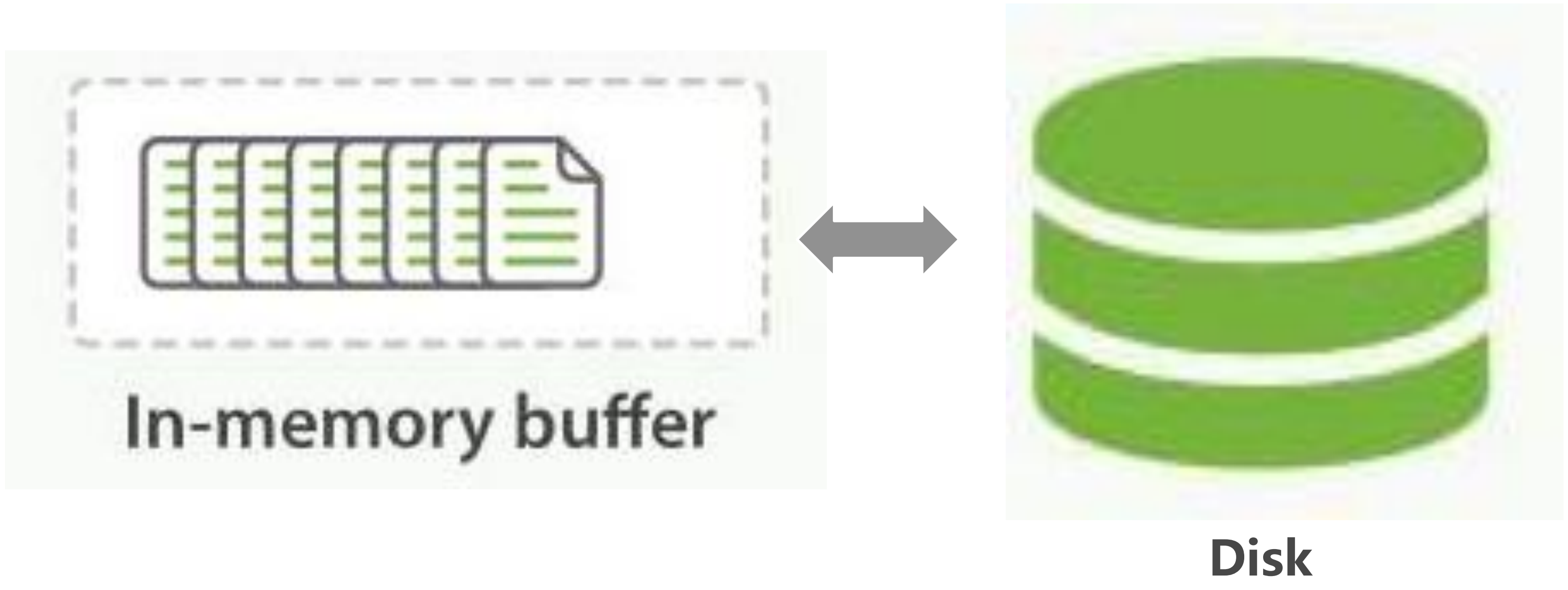
- DDBS must be able to provide additional functions to those of a centralized DBMS.

- Distribution leads to increased complexity in the system design and implementation.

Review: Querying in Centralized DBS

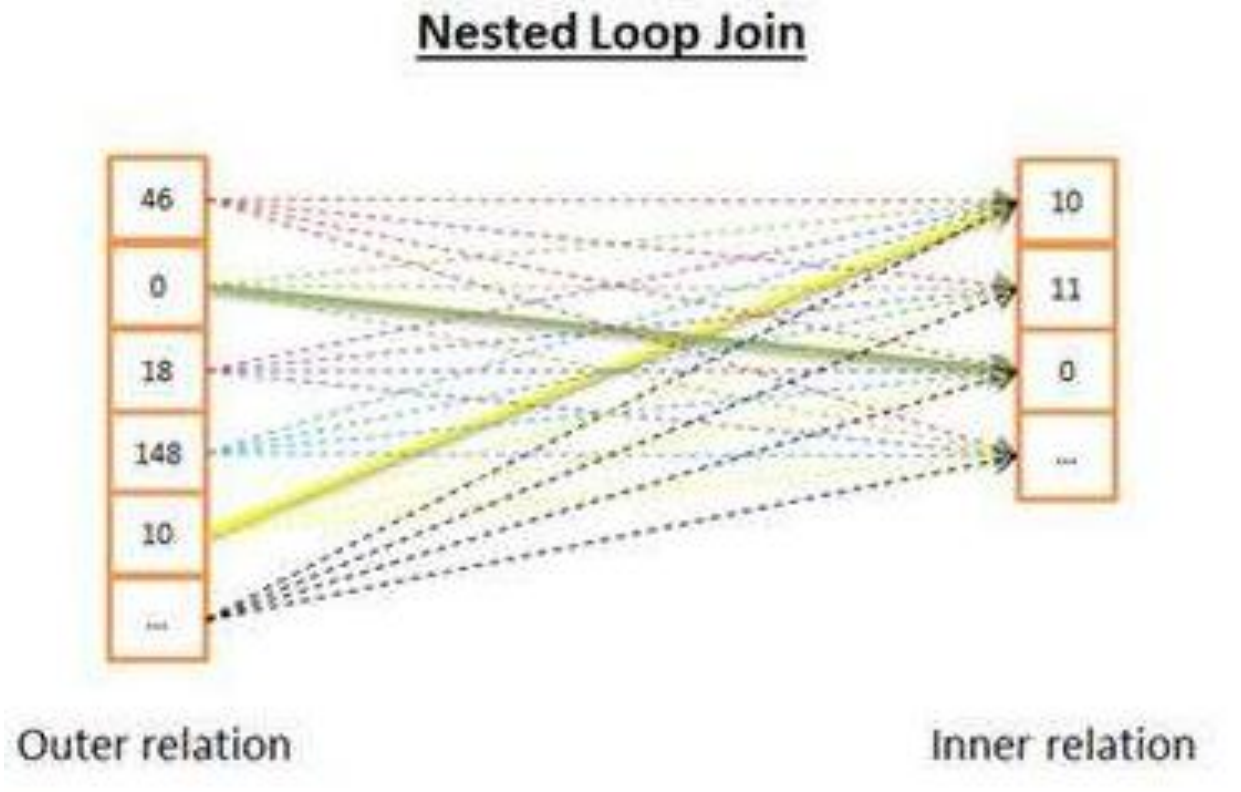
- Centralized relational DBS
  - data modeling (relation/table)
  - Physical storage of data and index

  - Query processing and optimization
    - Reduce disk I/O (DB pages in disk vs. read/write in memory)

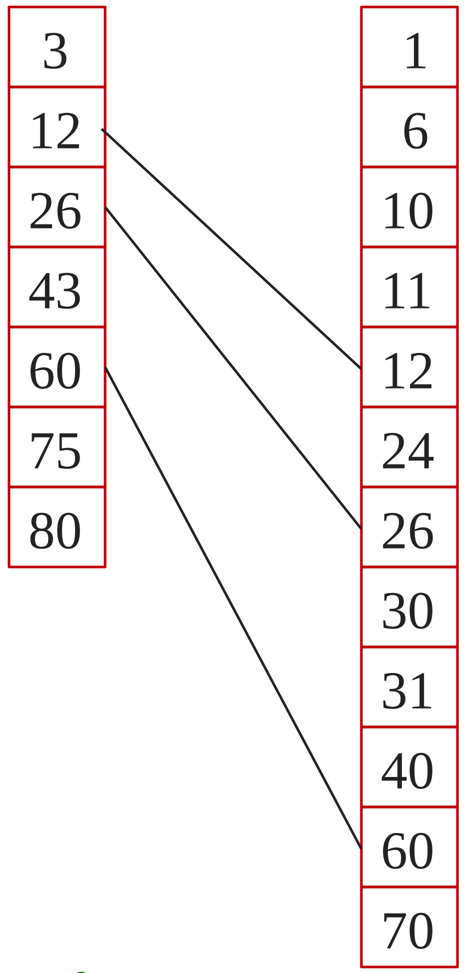


Review: Querying in Centralized DBS

- Centralized relational DBS
  - Query processing and optimization
    - reduce disk I/O (DB pages in disk vs. read/write in memory)
  - Reduce expensive join costs



Review: Querying in Centralized DBS

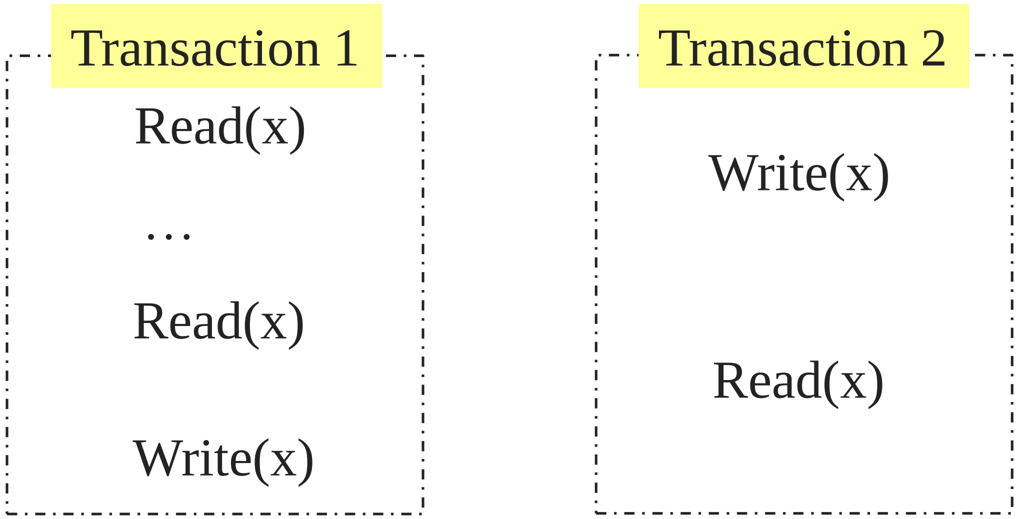


After sorting

Merge Join

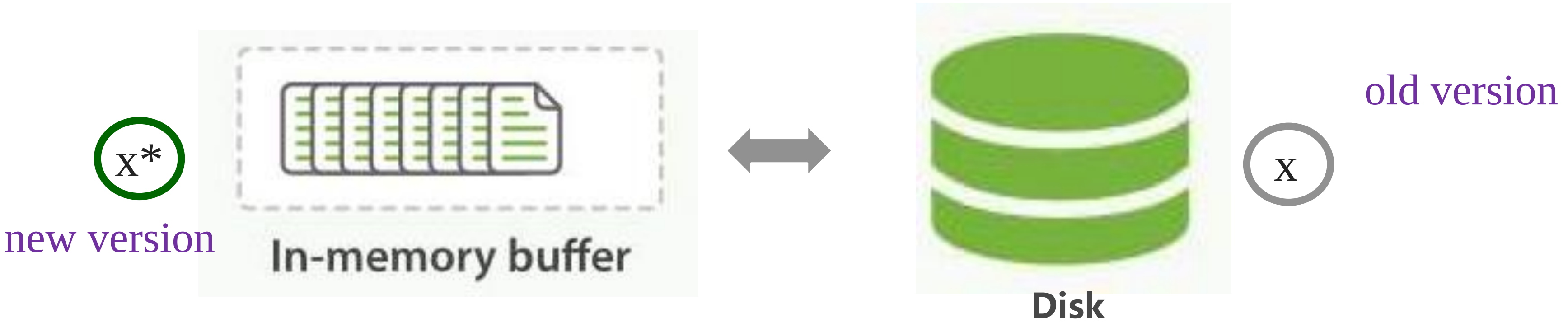
Review: Transaction in Centralized DBS

- Centralized DBS
  - Data modeling
  - Physical storage of data and index
  - Query processing and optimization
    - reduce disk I/O (DB pages in disk vs. read/write in memory)
    - reduce join costs
  - Transaction management and concurrency control
    - ACID (Atomicity, Consistency, Isolation, Durability)



Review: Reliability in Centralized DBS

- Centralized DBS
  - Data modeling
  - Physical storage of data and index
  - Query processing and optimization
    - reduce disk I/O (DB pages in disk vs. read/write in memory)
    - reduce join costs
  - Transaction management and concurrency control
    - ACID (Atomicity, Consistency, Isolation, Durability)
  - Reliability and failure recovery
    - strong consistency (between memory and disk)



Review: Security in Centralized DBS

- Centralized DBS
  - Data modeling
  - Physical storage of data and index
  - Query processing and optimization
    - reduce disk I/O (DB pages in disk vs. read/write in memory)
    - reduce join costs
  - Transaction management and concurrency control
    - ACID (Atomicity, Consistency, Isolation, Durability)
  - Reliability and failure recovery
    - strong consistency (between memory and disk)
  - Security

Review: Secure Data Management in Centralized DBS

- **User identification**
  - Each time the user requests to use the system, the system will check and provide the right to use the system after passing the identification.
- **Access Control**
  - Only legitimate users can access the database, and all unauthorized personnel cannot.
- **Audit**
  - Automatically record all users' operations on the database into the audit log.
  - DBA can use the audit tracking information to reproduce a series of DB events that lead to the current situation of the database, and find out the person, time and content of illegal access to data.

Review: Secure Data Management in Centralized DBS

- **Data Encryption**
  - Encrypt the stored and transmitted data so that those who don't know the decryption algorithm can't know the content of the data (talk later)
- **View Mechanism**
  - 为不同用户定义不同视图，通过视图机制把要保密的数据对无权存取的用户隐藏起来，从而自动地对数据提供一定程度的安全保护
  - Define different views for different users to protect the data from being accessed by the users who have no access right.

View in secure data management

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

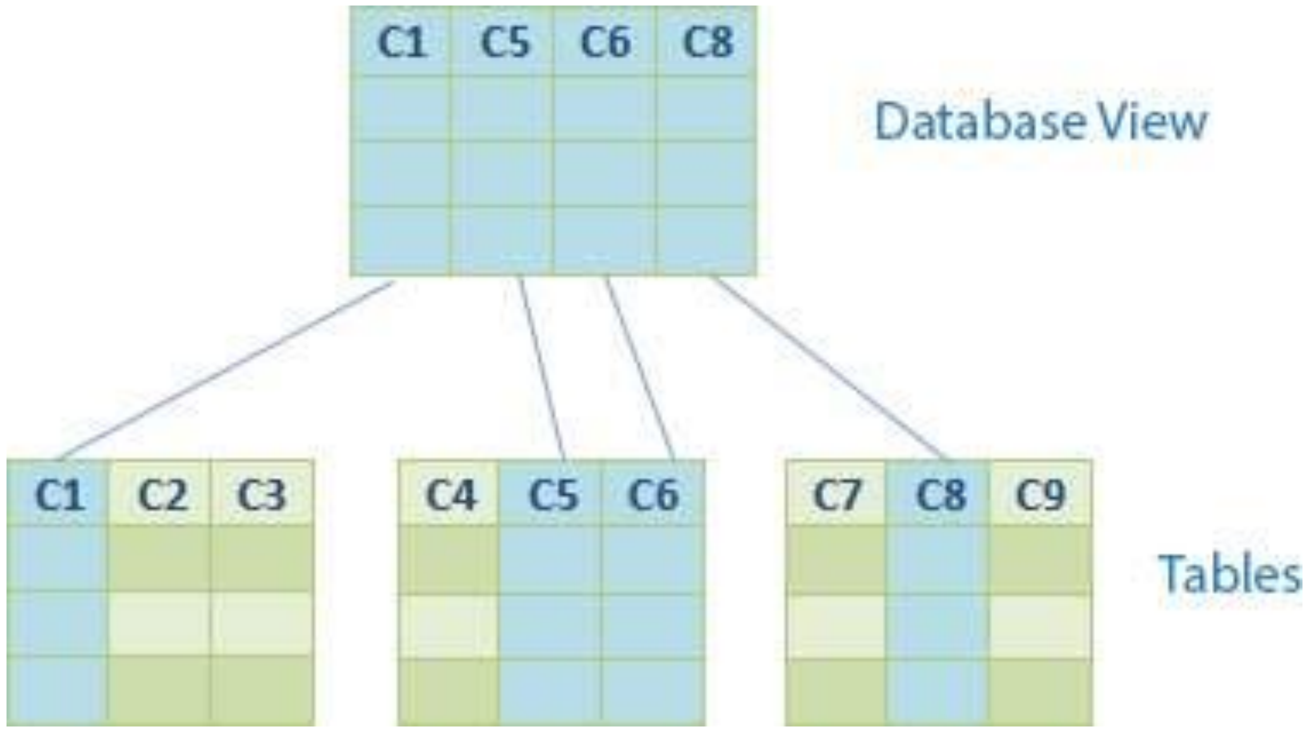
BUDGET

PNAME	BUDGET
Instrumentation	150000
Database Develop	135000
CAD/CAM	250000
Maintenance	310000

Create a BUDGET view from the PROJ relation

```
CREAT  VIEW     BUDGET(PNAME, BUD)
AS     SELECT   PNAME, BUDGET
       FROM     PROJ
```

View Example



Ordinary View vs. Materialized View

- Ordinary view
  - Virtual table
  - Named select statement

- Materialized view
  - A database object that stores the results of a query
  - Cache expensive queries
    - Expensive in terms of time or memory
    - E.g., sum, average, other calculations on large amounts of data

Advantages and Disadvantages of Materialized View

- Advantages
  - reduce system CPU/IO resource requirements by pre-calculating and storing results of intensive queries
  - Useful for summarizing, pre-computing, replicating and distributing data
  - Faster access for expensive and complex joins
  - Transparent to end-users
    - Materialized views can be added/dropped without invalidating coded SQL
- Disadvantages
  - Performance costs of maintaining the views
  - Storage costs of maintaining the views

Refresh Modes of Materialized View

- Can be refreshed on demand or on a schedule
- Refresh Modes
  - **ON COMMIT** – refreshes occur whenever a commit is performed on one of the view's underlying detail table(s)
    - Available only with single table aggregate or join-based views
    - Keep view data transactionally accurate
    - Need to check alert log for view creation errors
  - **ON DEMAND** – refreshes are initiated manually using one of the procedures in the DBMS_MVIEW package
    - Can be used with all types of materialized views
  - **START WITH [NEXT] <date>** - refreshes start at a specified date/time and continue at regular intervals

Refresh Options of Materialized View

- Refresh Options
  - **COMPLETE** – totally refreshes the view
    - – Can be done at any time; time-consuming
  - **FAST** – incrementally applies data changes
    - – A materialized view (MV) log is required on each detail table
    - – Data changes are recorded in MV logs or direct loader logs
  - **FORCE –** Try a FAST refresh, (if not possible) make COMPLETE
    - – The default refresh option

Additional Functions of DDBSs

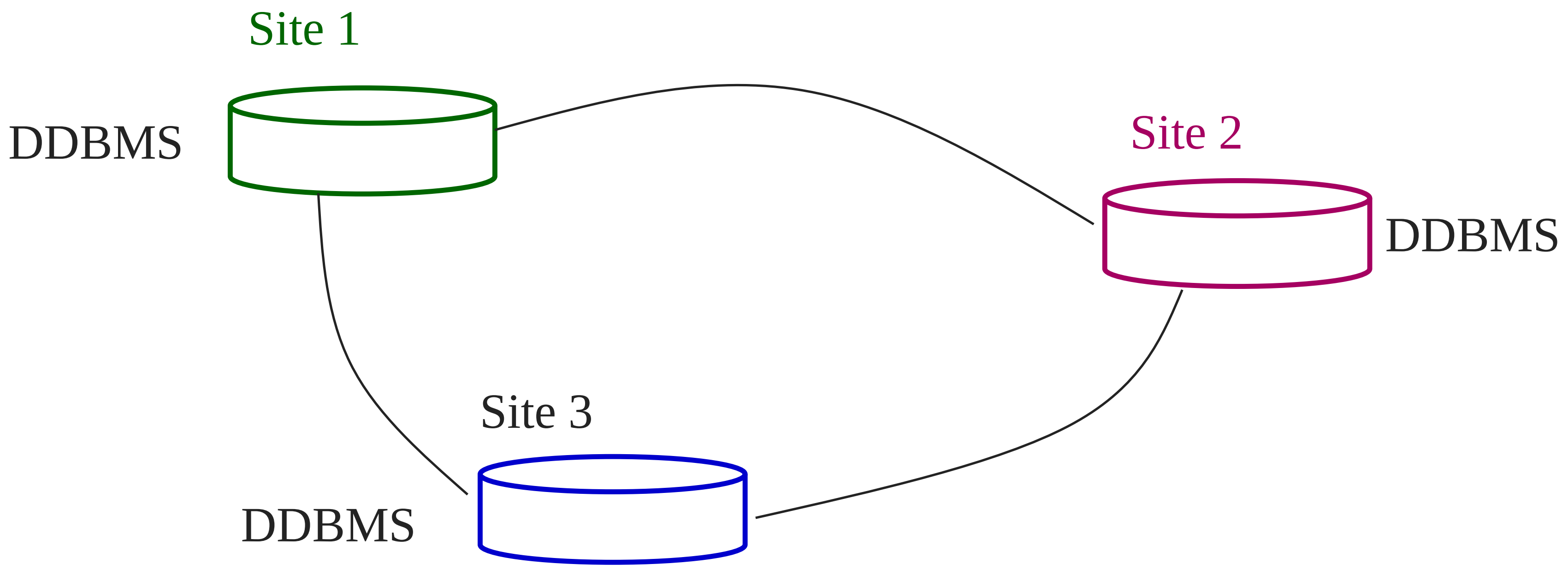
- To access remote sites and transmit queries and data among various sites via a communication network
- To keep track of data distribution and replication in the DDBMS catalog
- To devise execution strategies for queries and transactions that access data from more than one site
- To decide on which copy of a replicated data item to access
- To maintain the consistency of copies of a replicated data item
- To maintain the global conceptual schema of the distributed database
- To recover from individual site crashes and from new types of failures such as failure of a communication link

Distributed DBMS Issues

- Distributed Database System Architecture
- Distributed Database Design (fragmentation and allocation)
- Distributed Query Processing
- Distributed Transaction management and Concurrency Control
- Reliability of Distributed Databases

DDBS Architecture

- Client-Server
- Peer-to-Peer
- Master-Slave



Distributed Database Design

- The problem is how the database and the applications that run against it should be placed across the sites.
- Two fundamental design issues:
  - fragmentation (separation of the database into partitions called fragments)
  - allocation (distribution)
    - The optimum distribution of fragments: an NP-hard problem
    - Replicated & non-replicated allocation

Distributed Query Processing

- Query processing deals with designing algorithms that analyze queries and convert them into a series of data manipulation operations.

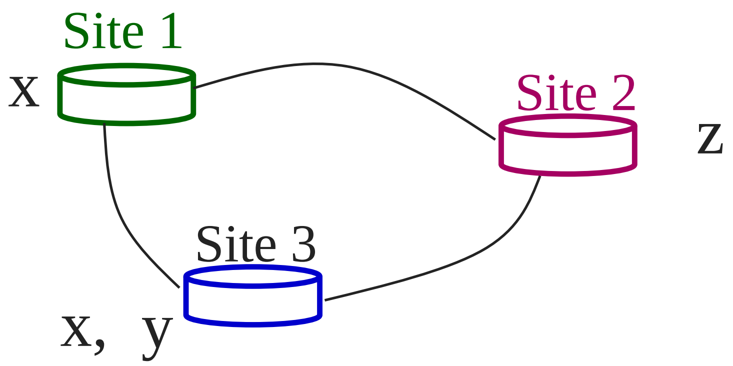
- The problem is to decide on strategy for executing each query <u>over the network</u> in the most cost effective way.
  The objective is to optimize where the inherent parallelism is used to improve the performance of executing the transaction.
  - min {cost = data transmission + local processing + … }

Distributed Directory Management

- A directory contains information (such as descriptions and locations) about data items in the database.
- A directory may be global to the entire DDBS, or local to each site, distributed, multiple copies, etc.

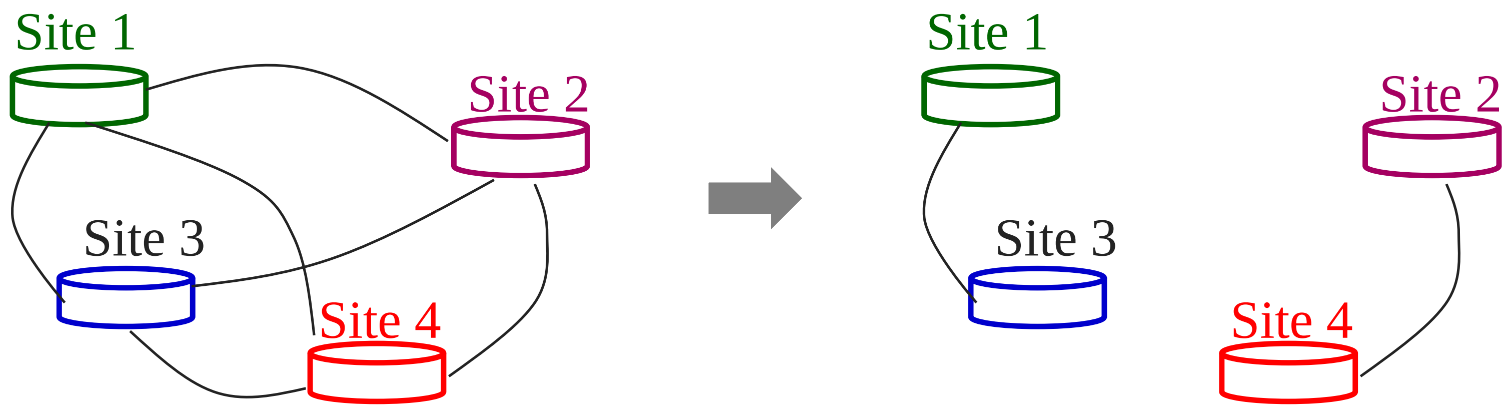
Distributed Concurrency Control

- Concurrency control involves
  - synchronization of accesses to the distributed database, such that the integrity of the database is maintained.
  - Consistency of multiple copies of the database (mutual consistency) and isolation of transactions' effects.
- Deadlock management



Reliability of Distributed DBMS

- It is important that mechanisms be provided to ensure the consistency of the database as well as to detect failures and recover from them.
- This may be extremely difficult in case of network partitioning, where the sites are divided into two or more groups with no communication among them.



Date’s 12 Rules for DDBSs

To the user, a distributed system should look exactly like a non-distributed system.

1. Local autonomy
2. No reliance on a central site
3. Continuous operation
4. Location independence
5. Fragmentation independence
6. Replication independence
7. Distributed query processing
8. Distributed transaction management
9. Hardware independence
10. Operating system independence
11. Network independence
12. DBMS independence

Rule 1: Local Autonomy

Sites should be autonomous to the maximum extent possible.

- Local data is locally owned and managed with local accountability
  - security consideration
  - integrity consideration
- Local operations remain purely local
- All operations at a given site are controlled by that site
  - No site X should depend on some other site Y for its successful functioning

Rule 1: Local Autonomy (cont.)

- In some situations, some slight loss of autonomy is inevitable.
  - fragmentation problem - Rule 5
  - replication problem - Rule 6
  - update of replicated relation - Rule 6
  - multiple-site integrity constraint problem - Rule 7
  - a problem of participation in a two-phase commit process - Rule 8

Rule 2: No Reliance on a Central Site

There must not be any reliance on a central "master" site for some central service, such as centralized query processing or centralized transaction management, such that the entire system is dependent on that central site.

- Reliance on a central site would be undesirable for at least the following two reasons:
  - that central site might be a bottleneck
  - the system would be vulnerable

Rule 2: No Reliance on a Central Site (cont.)

- In a distributed system, the following functions (among others) must therefore all be distributed:
  - Dictionary management
  - Query processing
  - Concurrency control
  - Recovery control

Rule 3: Continuous Operation

There should ideally never be any need for a planned entire system shut down.

- Incorporating a new site X into an existing distributed system D should not bring the entire system to a halt.
- Incorporating a new site X into an existing distributed system D should not require any changes to existing user programs or terminal activities.

Rule 3: Continuous Operation (cont.)

- Removing an existing site X from the distributed system should not cause any unnecessary interruptions in service.
- Within the distributed system, it should be possible to create and destroy fragments and replicas of fragments dynamically.
- It should be possible to upgrade the DBMS at any given component site to a newer release without taking the entire system down.

Rule 4: Location Independence (Transparency)

Users should not have to know where data is physically stored, but rather should be able to behave - at least from a logical standpoint - as if the data were all stored at their own local site.

- Simplify user programs and terminal activities.
- Allow data to migrate from site to site.
- It is easier to provide location independence for simple retrieval operations than it is for update operations.
- Distributed data naming scheme and corresponding support from the dictionary subsystem.

Rule 4: Location Independence (Transparency) (cont.)

- User naming scheme
  - User U has to have a valid logon ID at each of multiple sites to operate
  - User profile for each valid logon ID in the dictionary
  - Granting of access privileges at each component site

Rule 5: Fragmentation

- A distributed system supports data fragmentation if a given relation can be divided into pieces or "fragments" for physical storage purposes.
- Fragmentation is desirable for performance reason.
  - **Horizontal** and/or **Vertical** fragmentation

Employee

EMP#	DEPT#	SALARY
E1	DX	45K
E2	DY	40K
E3	DZ	50K
E4	DY	63K
E5	DZ	40K

New York Segment

EMP#	DEPT#	SALARY
E1	DX	45K
E3	DZ	50K
E5	DZ	40K

Physical storage
New York

London Segment

EMP#	DEPT#	SALARY
E2	DY	40K
E4	DY	63K

Physical storage
London

Rule 5: Fragmentation Independence (Transparency)

Users should be able to behave (at least from a logical standpoint) as if the data were in fact not fragmented at all.

- A system that supports data fragmentation should also support fragmentation independence (also known as fragmentation transparency).
- Fragmentation independence (like location independence) is desirable because it simplifies user programs and terminal activities.

Rule 5: Fragmentation Independence (Transparency) (cont.)

- Fragmentation independence implies that users should normally be presented with a view of the data in which the fragments are logically combined together by means of suitable joins and unions.

Rule 6: Replication Independence (Transparency)

- A distributed system supports data replication if a given relation (more generally, a given fragment of a relation) can be represented at the physical level by many distinct stored copies or replicas, at many distinct sites.
- Replication is desirable for at least two reasons
  - Performance
  - Availability

Employee

EMP#	DEPT#	SALARY
E1	DX	45K
E2	DY	40K
E3	DZ	50K
E4	DY	63K
E5	DZ	40K

NewYork Segment

EMP#	DEPT#	SALARY
E1	DX	45K
E3	DZ	50K
E5	DZ	40K

Replica of London Segment

EMP#	DEPT#	SALARY
E2	DY	40K
E4	DY	63K

London Segment

EMP#	DEPT#	SALARY
E2	DY	40K
E4	DY	63K

Replica of New York Segment

EMP#	DEPT#	SALARY
E1	DX	45K
E3	DZ	50K
E5	DZ	40K

Rule 6: Replication Independence (Transparency) (cont.)

Users should be able to behave as if the data were in fact not replicated at all.

- User should be able to behave as if the data were in fact not replicated at all.
- Replication, like fragmentation, should be “transparent to the user”.
- Update propagation problem
- Replication independence (like location and fragmentation independence) is desirable because it simplifies user programs and terminal activities.

Rule 7: Distributed Query Processing

It is crucially important for distributed database systems to choose a good strategy for distributed query processing.

- Query processing in a distributed system involves
  - local CPU and I/O activities at several distinct sites
  - some amount of data communication among those sites
  - energy, cost, …
- Amount of data communication is a major performance factor.
- Query compilation ahead of time

Rule 8: Distributed Transaction Management

Two major aspects of transaction management, concurrency control and recovery control, require extended treatment in a distributed environment.

- In a distributed system, a single transaction can involve the execution of code at multiple sites and can thus involve updates at multiple sites.
- Each transaction is therefore said to consist of multiple "agents," where an agent is the process performed on behalf of a given transaction at a given site.
- Deadlock problem may be incurred.

Rule 9: Hardware Independence (Transparency)

User should be presented with a "single-system image" regardless of any particular hardware platform.

- It is desirable to be able to run the same DBMS on different hardware systems.
- It is desirable to have those different hardware systems all participate as equal partners (where appropriate) in a distributed system.
- It is assumed that the same DBMS is running on all those different hardware systems.

Rule 10: Operating System Independence

It is obviously desirable, not only to be able to run the same DBMS on different hardware systems, but also to be able to run it on different operating systems - even different operating systems on the same hardware.

- From a commercial point of view, the most important operating system environments, and hence the ones that (at a minimum) the DBMS should support, are probably MVS/XA, MVS/ESA, VM/CMS, VAX/VMS, UNIX (various flavors), OS/2, MS/DOS, Windows,…

Rule 11: Network Independence

It is obviously desirable to be able to support a variety of disparate communication networks.

- From the viewpoint of the distributed DBMS, the network is merely the provider of a reliable message transmission service.
- “Reliable" here means that, if the network accepts a message from site X for delivery to site Y, then it will eventually deliver that message to site Y.
- Messages will not be delivered more than once, and will be delivered in the order sent.
- The network should also be responsible for site authentication.
- Ideally the system should support both local and wide area networks.
- A distributed system should support a variety of different network architectures.

Rule 12: DBMS Independence

Ideally a distributed system should provide DBMS independence (or transparency).

Question and Answer