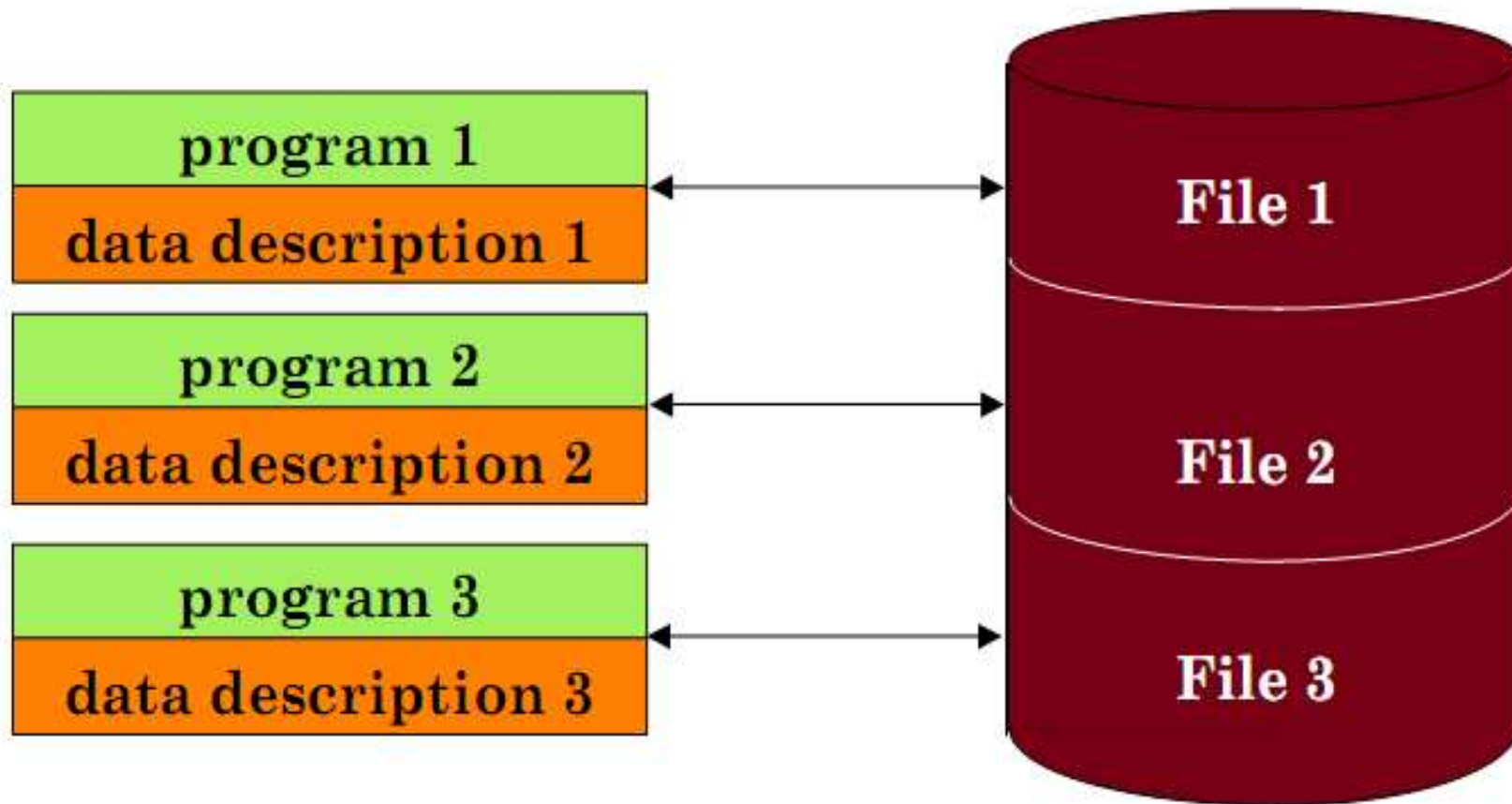


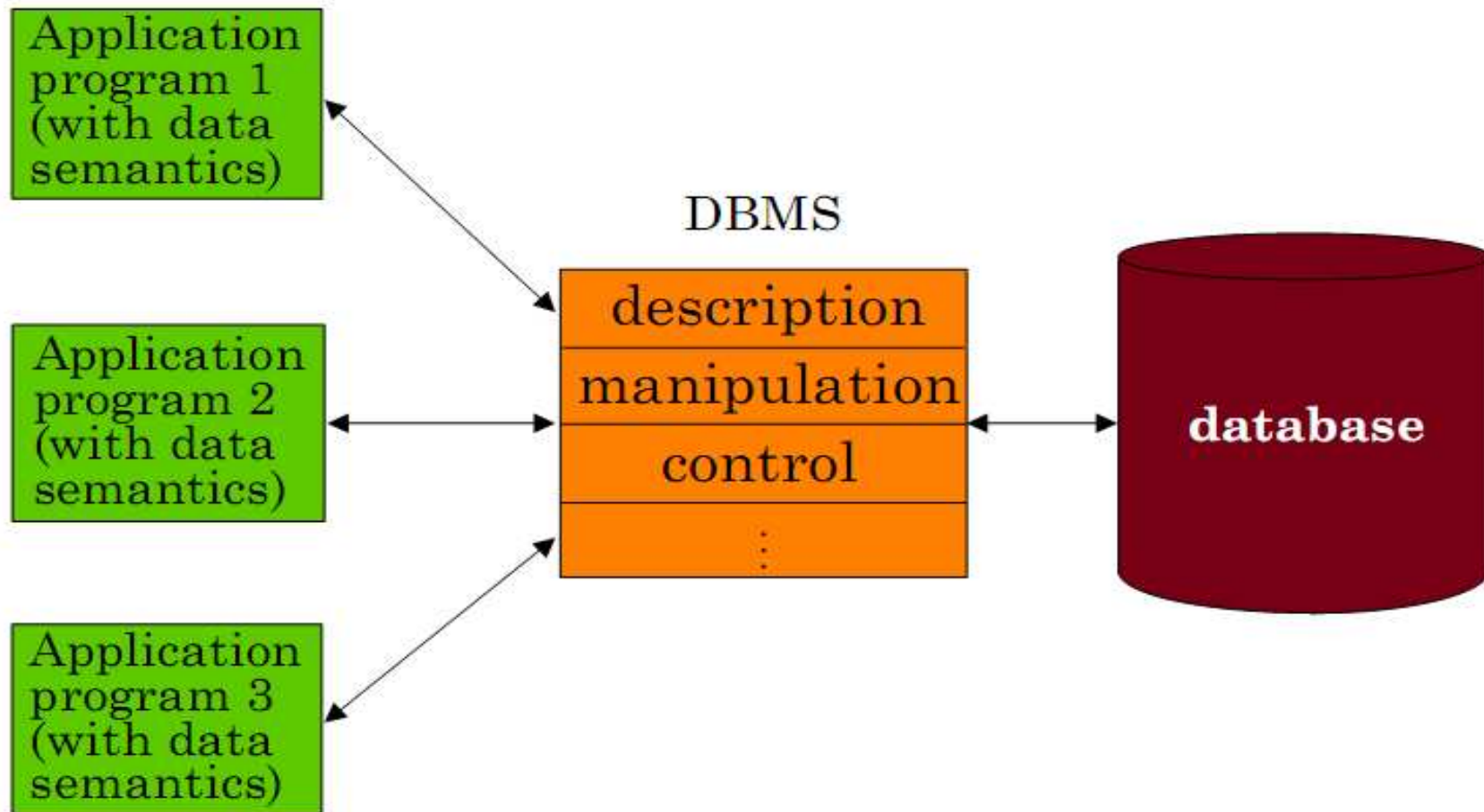
Sistem Terdistribusi

Sistem Basis Data Terdistribusi

File system



Database system



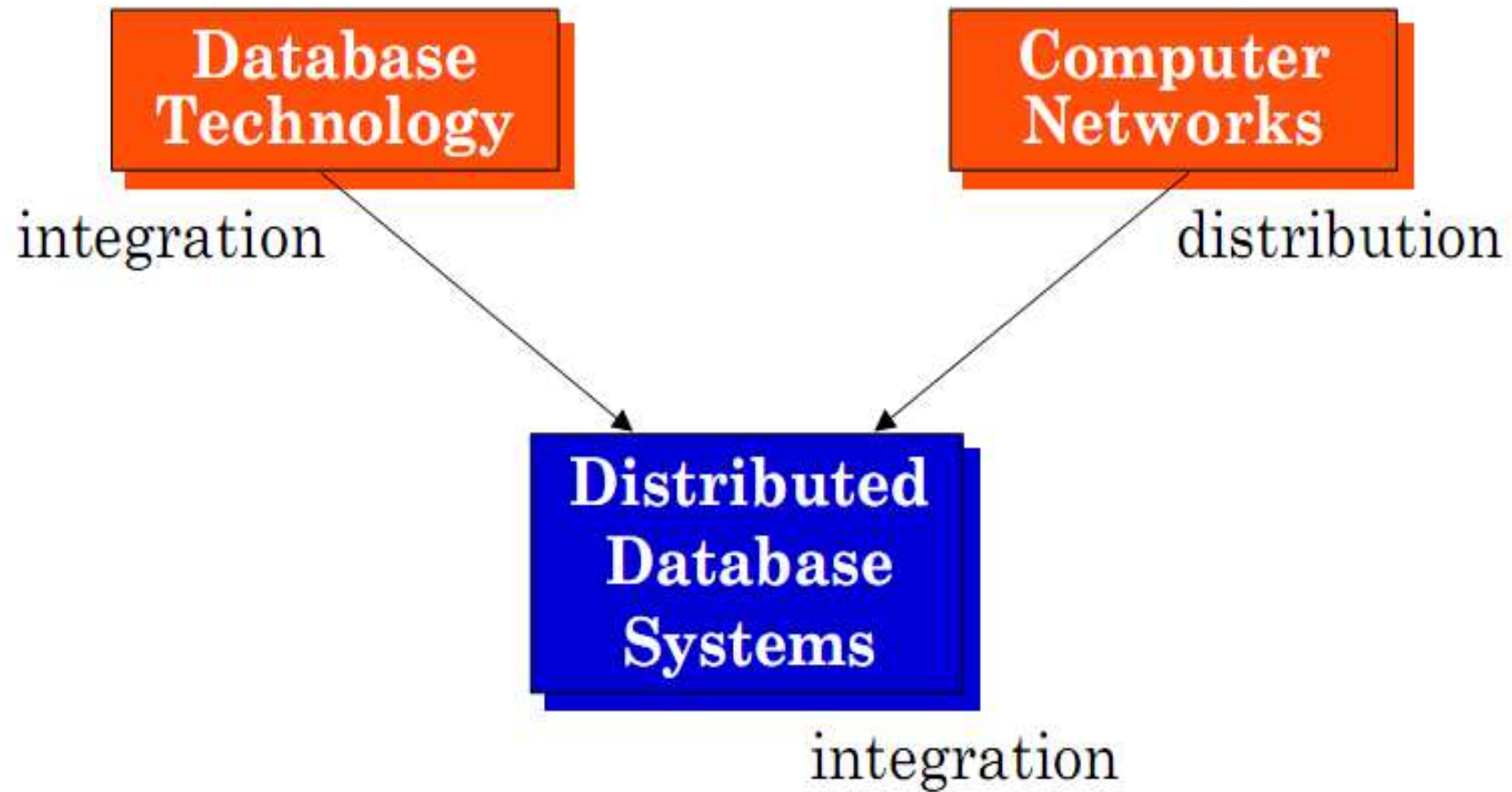
Database Management System (DBMS)

- DBMS contains information 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: all transactions
 - Airlines: reservations, schedules
 - Universities: registration, grades
 - Sales: customers, products, purchases
 - Online retailers: order tracking, customized recommendations
 - Manufacturing: production, inventory, orders, supply chain
 - Human resources: employee records, salaries, tax deductions

Tujuan DBMS

- Protect from redundancy and inconsistency
 - Multiple file formats, duplication of information in different files
- Protect from difficulty in accessing data
 - Need to write a new program to carry out each new task
- Make data isolation — multiple files and formats
- Protect from integrity problems
 - Integrity constraints (e.g. account balance > 0) become “buried” in program code rather than being stated explicitly
 - Hard to add new constraints or change existing ones
- Make atomicity of updates
 - Failures may leave database in an inconsistent state with partial updates carried out
 - Example: Transfer of funds from one account to another should either complete or not happen at all
- Protect from concurrent access by multiple users
 - Concurrent accessed needed for performance
 - Uncontrolled concurrent accesses can lead to inconsistencies
 - Example: Two people reading a balance and updating it at the same time
- Protect from security problems
 - Hard to provide user access to some, but not all, data

Motivasi Distributed Database



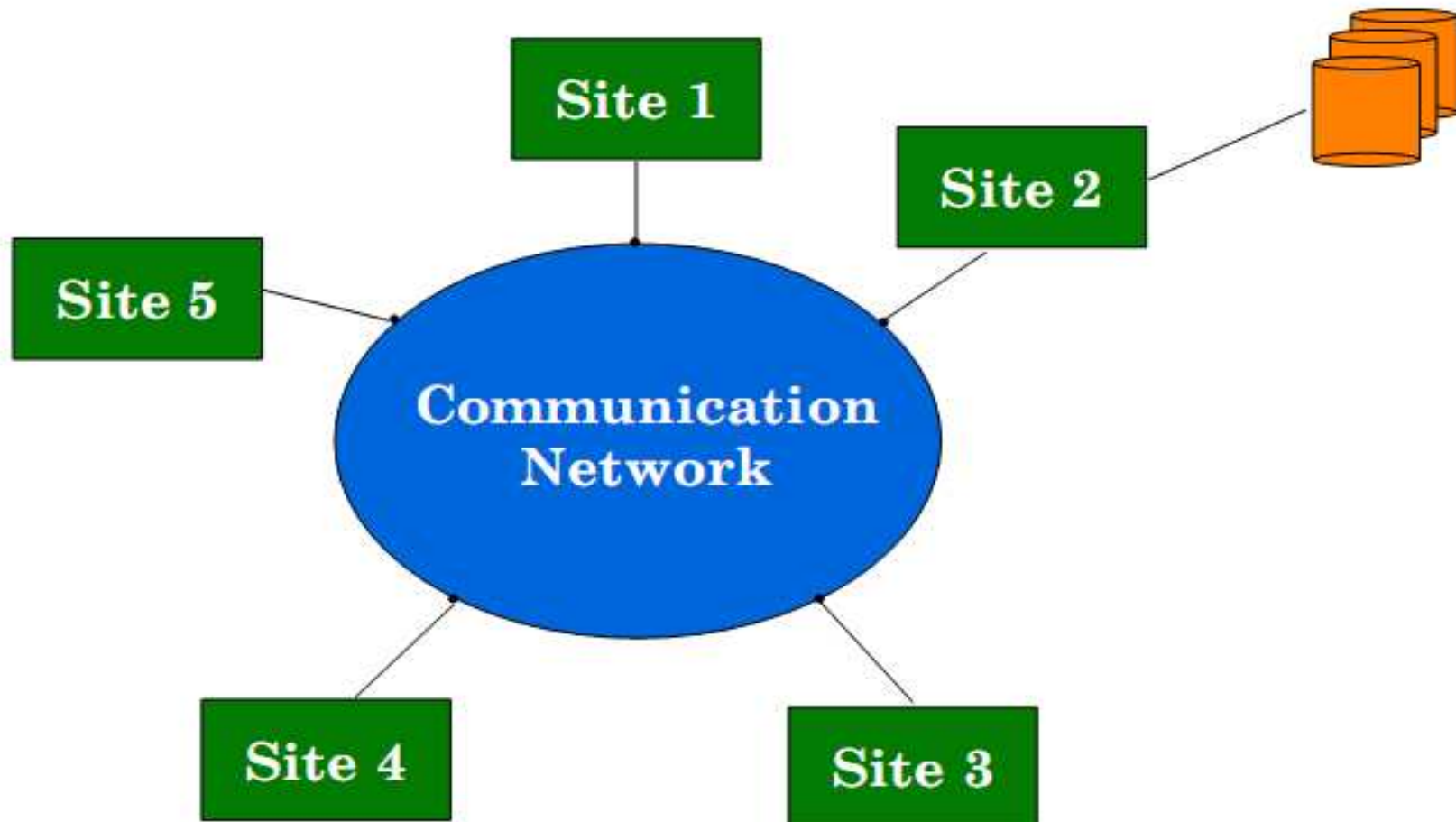
Distributed Database System

- A distributed database (DDB) is a collection of multiple, **logically interrelated** databases distributed over a **computer network**.
- A distributed database management system (D-DBMS) is the software that manages the DDB and provides an access mechanism that makes this distribution **transparent** to the users.
- Distributed database system (DDBS) = DDB + D-DBMS

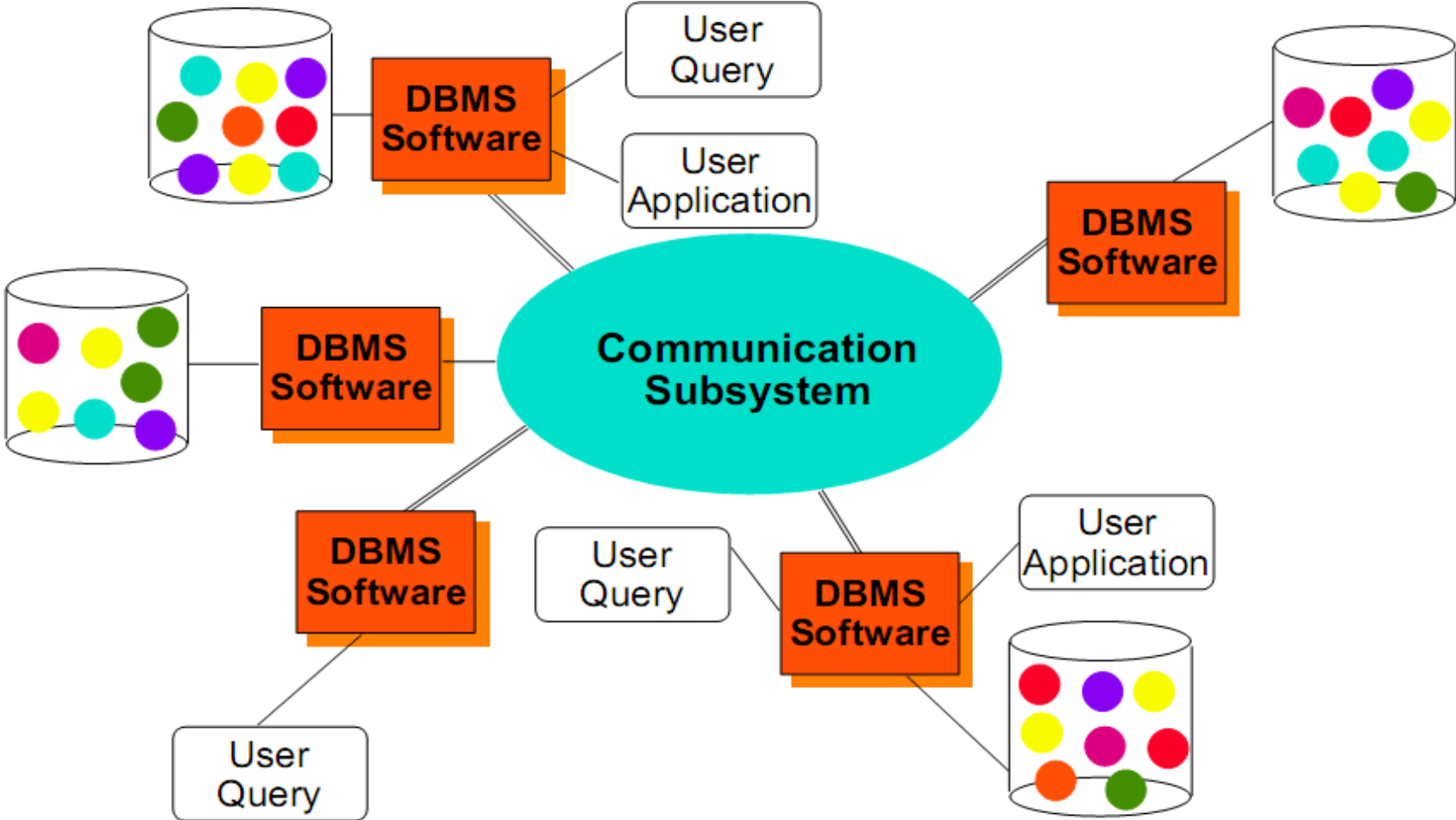
Distributed Database System

- A distributed database system consists of **loosely coupled** sites that **share no physical component**
- Database systems that run on each site are **independent of each other**
- Transactions may access data at **one or more sites**

Not Distributed Db, but Centralized Db



Distributed Db



Ciri-ciri Distributed Db

- Data disimpan pada sejumlah tempat
 - Pada setiap tempat prosesornya tunggal
- Prosesor berada ditempat berbeda, dihubungkan dgn jaringan
- Distributed Db bukan file, melainkan database
- Setiap tempat dapat memproses permintaan secara mandiri dan jg dapat memproses permintaan dari tempat lain.

KELEBIHAN BASIS DATA TERDISTRIBUSI

- **Otonomi Lokal:** Karena data terdistribusi, kelompok user yang bisa menggunakan data tersebut dapat menyimpannya di site dimana dia bekerja, sehingga masing-masing site mempunyai kontrol lokal.
- **Performansi Tinggi:** Karena data yang digunakan umumnya lebih dekat dengan user, maka performansi akses ke database dapat ditingkatkan.
- **Kepercayaan:** Karena ada replikasi data di lebih dari satu tempat, maka kemungkinan crashnya satu node/site, atau hilangnya aksesibilitas karena kegagalan link komunikasi, tidak membuat data untuk tidak dapat diakses.

Kelebihan Basis Data Terdistribusi

- **Ekonomis:**
 - Dari sisi communication cost: Lebih ekonomis dengan membagi aplikasi dan menjalankannya di beberapa situs lokal, dari pada satu aplikasi dipaksa untuk akses ke database yang tersebar.
 - Dari sisi biaya normal: Lebih ekonomis membeli komputer kecil-kecil yang kemampuannya sama dengan satu komputer besar.
- **Ekspansinya Mudah:** Dapat disesuaikan dengan mudah seiring dengan berkembangnya ukuran database. Paling sekitar penambahan pengolahan dan kemampuan penyimpanan di jaringan. Jelas, bahwa kemampuannya tidak akan bertambah secara linier, tapi paling tidak improvisasi dapat dimungkinkan.
- **Shareability:** Untuk merespon trend organisasi ke arah distributed fashion

KELEMAHAN BASIS DATA TERDISTRIBUSI

- **Kurangnya Pengalaman**: Minimnya percobaan di lingkup kerja nyata.
- **Kompleksitas**: Menambah masalah baru dari masalah yang ada pada DBS.
- **Biaya**: Menambah biaya hardware (hardware u/ mekanisme komunikasi).
 - Memang H/W sekarang murah shg biaya tdk terlalu signifikan, tapi biaya pengembangan teknologi/software dalam masalah teknis (distributed debuggers dll) sangat diperlukan.
 - Selain itu adalah biaya untuk replikasi usaha/tenaga (manpower), yang dibutuhkan di beberapa site → Analisa trade-off antara keuntungan dari efisiensi/efektivitas penggunaan informasi dengan biaya penambahan personel.

KELEMAHAN BASIS DATA TERDISTRIBUSI

- **Kontrol yang Terdistribusi**: Ini kelebihan yang sudah disebut, tapi sayangnya ini dapat menimbulkan masalah sinkronisasi dan koordinasi
- **Keamanan**: Sekuriti jaringan lebih kompleks dari sekuriti akses database yang tersentralisasi.
- **Migrasi yang susah**: Belum ada tool atau metodologi yang dapat membantu untuk konversi dari DBS ke DDBS.

Homogeneous & Heterogeneous Distributed Db

- In a homogeneous distributed database
 - All sites have identical software
 - Are aware of each other and agree to cooperate in processing user requests.
 - Each site surrenders part of its autonomy in terms of right to change schemas or software
 - Appears to user as a single system
- In a heterogeneous distributed database
 - Different sites may use different schemas and software
 - Difference in schema is a major problem for query processing
 - Difference in software is a major problem for transaction processing
 - Sites may not be aware of each other and may provide only limited facilities for cooperation in transaction processing

Distributed Data Storage

- Assume relational data model
- Replication
 - System maintains multiple copies of data, stored in different sites, for faster retrieval and fault tolerance.
- Fragmentation
 - Relation is partitioned into several fragments stored in distinct sites
- Replication and fragmentation can be combined
 - Relation is partitioned into several fragments: system maintains several identical replicas of each such fragment.

Data Fragmentation

- Division of relation r into fragments r_1, r_2, \dots, r_n which contain sufficient information to reconstruct relation r .
- **Horizontal fragmentation:** each tuple of r is assigned to one or more fragments
- **Vertical fragmentation:** the schema for relation r is split into several smaller schemas
 - All schemas must contain a common candidate key (or superkey) to ensure lossless join property.
 - A special attribute, the tuple-id attribute may be added to each schema to serve as a candidate key.
- Example :
Account = (branch_name, account_number, balance)

Horizontal Fragmentation of *account* Relation

<i>branch_name</i>	<i>account_number</i>	<i>balance</i>
Hillside	A-305	500
Hillside	A-226	336
Hillside	A-155	62

$$account_1 = \sigma_{branch_name="Hillside"}(account)$$

<i>branch_name</i>	<i>account_number</i>	<i>balance</i>
Valleyview	A-177	205
Valleyview	A-402	10000
Valleyview	A-408	1123
Valleyview	A-639	750

$$account_2 = \sigma_{branch_name="Valleyview"}(account)$$

Fragmentasi Vertikal

Deposit' = Deposit-scheme U tuple-id

Branch-name	Account-number	Customer-name	balance	Tuple-id
Hillside	305	Lowman	500	1
Hillside	226	Camp	336	2
Valleyview	117	Camp	205	3
Valleyview	402	Kahn	10000	4
Hillside	155	Kahn	62	5
Valleyview	408	Kahn	1123	6
Valleyview	639	Green	750	7

Deposit-scheme-3 = (branch-name, customer-name, tuple-id)

Deposit-scheme-4 = (account-number, balance, tuple-id)

Vertical Fragmentation

<i>branch_name</i>	<i>customer_name</i>	<i>tuple_id</i>
Hillside	Lowman	1
Hillside	Camp	2
Valleyview	Camp	3
Valleyview	Kahn	4
Hillside	Kahn	5
Valleyview	Kahn	6
Valleyview	Green	7

<i>account_number</i>	<i>balance</i>	<i>tuple_id</i>
A-305	500	1
A-226	336	2
A-177	205	3
A-402	10000	4
A-155	62	5
A-408	1123	6
A-639	750	7

Advantages of Fragmentation

- Horizontal:
 - allows parallel processing on fragments of a relation
 - allows a relation to be split so that tuples are located where they are most frequently accessed
- Vertical:
 - allows tuples to be split so that each part of the tuple is stored where it is most frequently accessed
 - tuple-id attribute allows efficient joining of vertical fragments
 - allows parallel processing on a relation
- Vertical and horizontal fragmentation can be mixed.
 - Fragments may be successively fragmented to an arbitrary depth.

Contoh lain

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	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P5	Engineer	23
E8	P3	Manager	40

PROJ

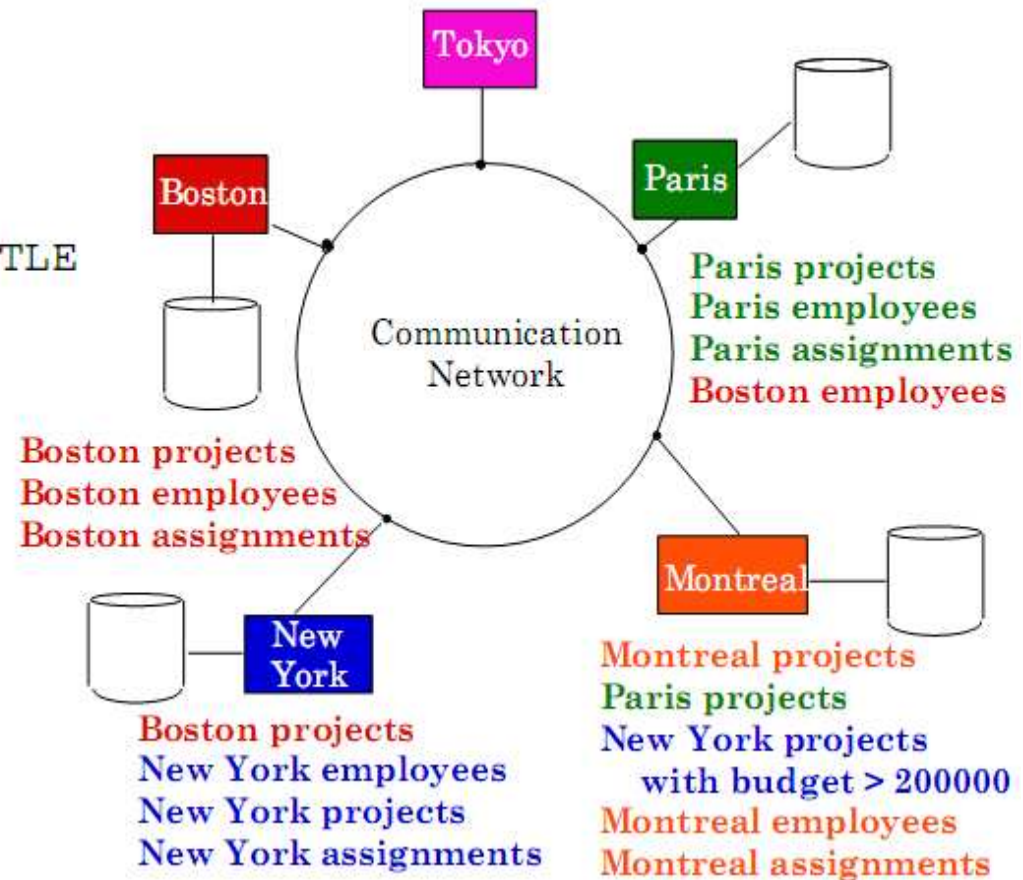
PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY

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

Distributed Db

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



Distributed Transactions

- Transaction may access data at several sites.
- Each site has a local transaction manager responsible for:
 - Maintaining a log for recovery purposes
 - Participating in coordinating the concurrent execution of the transactions executing at that site.
- Each site has a transaction coordinator, which is responsible for:
 - Starting the execution of transactions that originate at the site.
 - Distributing subtransactions at appropriate sites for execution.
 - Coordinating the termination of each transaction that originates at the site, which may result in the transaction being committed at all sites or aborted at all sites.

System Failure Modes

- Failures unique to distributed systems:
 - Failure of a site.
 - Loss of messages
 - Handled by network transmission control protocols such as TCP-IP
 - Failure of a communication link
 - Handled by network protocols, by routing messages via alternative links
 - **Network partition**
 - A network is said to be **partitioned** when it has been split into two or more subsystems that lack any connection between them
 - Note: a subsystem may consist of a single node
- Network partitioning and site failures are generally indistinguishable.

Distributed Query Processing

- For centralized systems, the primary criterion for measuring the cost of a particular strategy is the number of disk accesses.
- In a distributed system, other issues must be taken into account:
 - The cost of a data transmission over the network.
 - The potential gain in performance from having several sites process parts of the query in parallel.

Query Transformation

- Translating algebraic queries on fragments.
 - It must be possible to construct relation r from its fragments
 - Replace relation r by the expression to construct relation r from its fragments

- Consider the horizontal fragmentation of the *account* relation into

$$account_1 = \sigma_{branch_name = \text{“Hillside”}}(account)$$

$$account_2 = \sigma_{branch_name = \text{“Valleyview”}}(account)$$

- The query $\sigma_{branch_name = \text{“Hillside”}}(account)$ becomes

$$\sigma_{branch_name = \text{“Hillside”}}(account_1 \cup account_2)$$

which is optimized into

$$\sigma_{branch_name = \text{“Hillside”}}(account_1) \cup \sigma_{branch_name = \text{“Hillside”}}(account_2)$$

Example Query (Cont.)

- Since $account_1$ has only tuples pertaining to the Hillside branch, we can eliminate the selection operation.
- Apply the definition of $account_2$ to obtain $\sigma_{branch_name = \text{“Hillside”}} (\sigma_{branch_name = \text{“Valleyview”}} (account))$
- This expression is the empty set regardless of the contents of the $account$ relation.
- Final strategy is for the Hillside site to return $account_1$ as the result of the query.

Directory Systems

- Typical kinds of directory information
 - Employee information such as name, id, email, phone, office addr, ..
 - Even personal information to be accessed from multiple places
 - e.g. Web browser bookmarks
- White pages
 - Entries organized by name or identifier
 - Meant for forward lookup to find more about an entry
- Yellow pages
 - Entries organized by properties
 - For reverse lookup to find entries matching specific requirements
- When directories are to be accessed across an organization
 - Alternative 1: Web interface. Not great for programs
 - Alternative 2: Specialized **directory access protocols**
 - Coupled with specialized user interfaces

Directory Access Protocols

- Most commonly used directory access protocol:
 - LDAP (Lightweight Directory Access Protocol)
 - Simplified from earlier X.500 protocol
- Question: Why not use database protocols like ODBC/JDBC?
- Answer:
 - Simplified protocols for a limited type of data access, evolved parallel to ODBC/JDBC
 - Provide a nice hierarchical naming mechanism similar to file system directories
 - Data can be partitioned amongst multiple servers for different parts of the hierarchy, yet give a single view to user
 - E.g. different servers for Bell Labs Murray Hill and Bell Labs Bangalore
 - Directories may use databases as storage mechanism

NEXT