Distributed Databases

Database System Concepts, 6th Ed.

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

Homogeneous Distributed Databases

■ 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 Replication

- A relation or fragment of a relation is **replicated** if it is stored redundantly in two or more sites.

- Full replication of a relation is the case where the relation is stored at all sites.

- Fully redundant databases are those in which every site contains a copy of the entire database.

Data Replication (Cont.)

- Advantages of Replication

  - **Availability**: failure of site containing relation r does not result in unavailability of r is replicas exist.

  - **Parallelism**: queries on r may be processed by several nodes in parallel.

  - **Reduced data transfer**: relation r is available locally at each site containing a replica of r.

- Disadvantages of Replication

  - Increased cost of updates: each replica of relation r must be updated.

  - Increased complexity of concurrency control: concurrent updates to distinct replicas may lead to inconsistent data unless special concurrency control mechanisms are implemented.

    - One solution: choose one copy as **primary copy** and apply concurrency control operations on primary copy

Data Fragmentation

- Division of relation r into fragments r_1, r_2, …, 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.

Horizontal Fragmentation of *account* Relation

branch_name	account_number	balance
Hillside	A-305	500
Hillside	A-226	336
Hillside	A-155	62

$$account_1 = \sigma_{branch\_name=\text{"Hillside"}} (account)$$

branch_name	account_number	balance
Valleyview	A-177	205
Valleyview	A-402	10000
Valleyview	A-408	1123
Valleyview	A-639	750

$$account_2 = \sigma_{branch\_name=\text{"Valleyview"}} (account)$$

Vertical Fragmentation of *employee_info* Relation

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

$deposit_1 = \Pi_{branch\_name,\ customer\_name,\ tuple\_id}\ (employee\_info\ )$

account_number	balance	tuple_id
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

$deposit_2 = \Pi_{account\_number,\ balance,\ tuple\_id}\ (employee\_info\ )$

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.