Lecture 9:Introduction into database systems

Course instructors: Alexey Frolov and Yury Yanovich

Teaching assistant: Stanislav Kruglik

Technical assistants: Marina Dudina, Anton Glebov, Evgeny Marshakov

November 27, 2018

Database system

A **database** is an organized collection of data. A **relational database**, more restrictively, is a database with introduced relations over data.

A **database-management system (DBMS)** is a computer-software application that interacts with end-users, other applications, and the database itself to capture and analyze data. A general-purpose DBMS allows the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, PostgreSQL, EnterpriseDB, MongoDB, MariaDB, Microsoft SQL Server, Oracle, Sybase, SAP HANA, MemSQL, SQLite and IBM DB2.

Relation

Formally, given sets $D_1, D_2, \ldots D_n$ a relation r is a subset of $D_1 \times D_2 \times \cdots \times D_n$ Thus a relation is a set of n-tuples $(a_1, a_2 \ldots a_n)$ where each $a_i \in D_i$

Example

if
customer-name= {Jones, Smith, Curry, Lindsay}
customer-street= {Main, North, Park}
customer-city= {Harrison, Rye, Pittsfield}

Then r= { (Jones, Main, Harrison),
(Smith, North, Rye),
(Curry, North, Rye),
(Lindsay, Park, Pittsfield)}
is a relation over customer-name x customer-street x customer-city

Attribute types

Each attribute of a relation has a name

The set of allowed values for each attribute is called the domain of the attribute

Examples of simple domain types:

- Integer
- String
- Date

Relation schema

- $A_1, A_2, .. A_n$ are attributes
- $R = (A_1, A_2, \ldots A_n)$ is a relation schema

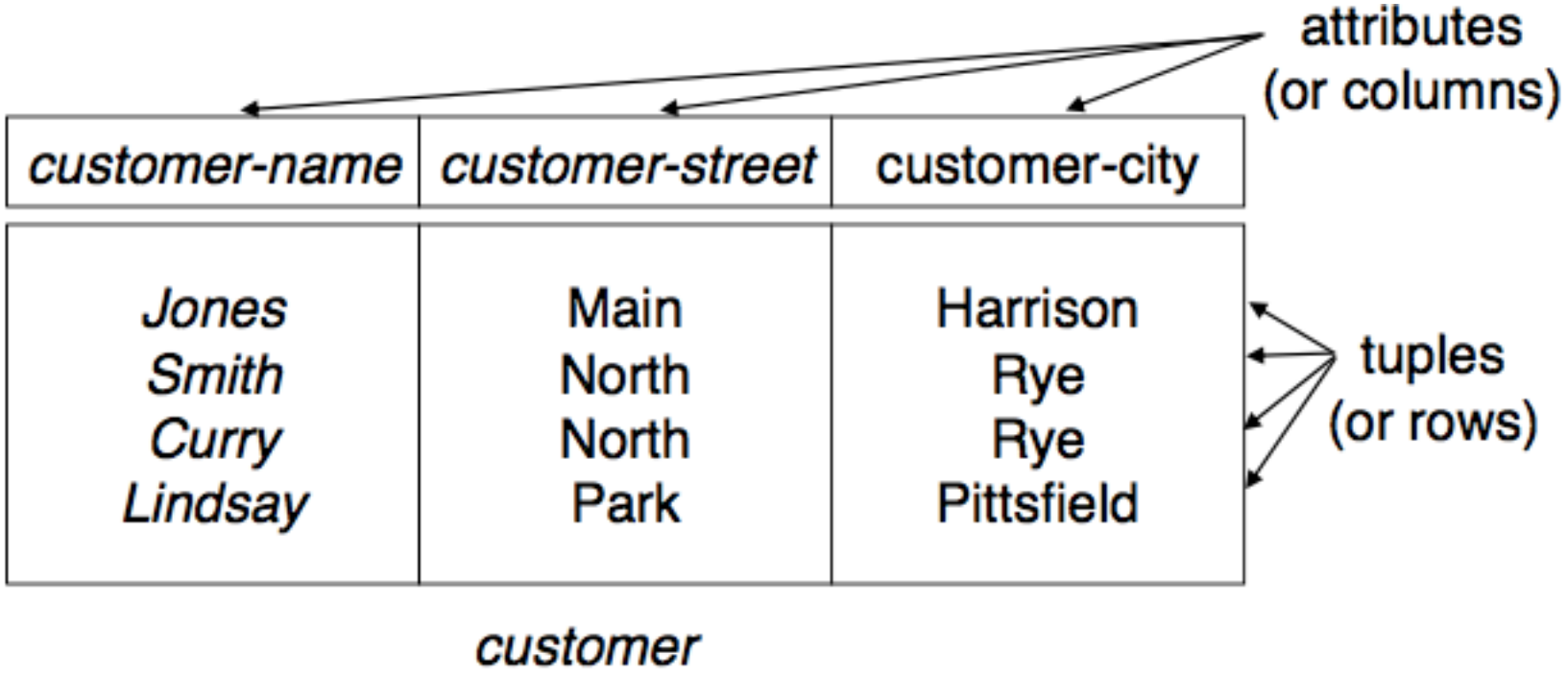
e.g. Account-schema = (account-number, branch-name, balance)

- r(R) is a relation on the relation schema R

e.g. customer (Customer-schema)

Relation instance

- The current values (relation instance) of a relation are specified by a table
- An element t of r is a tuple, represented by a row in a table



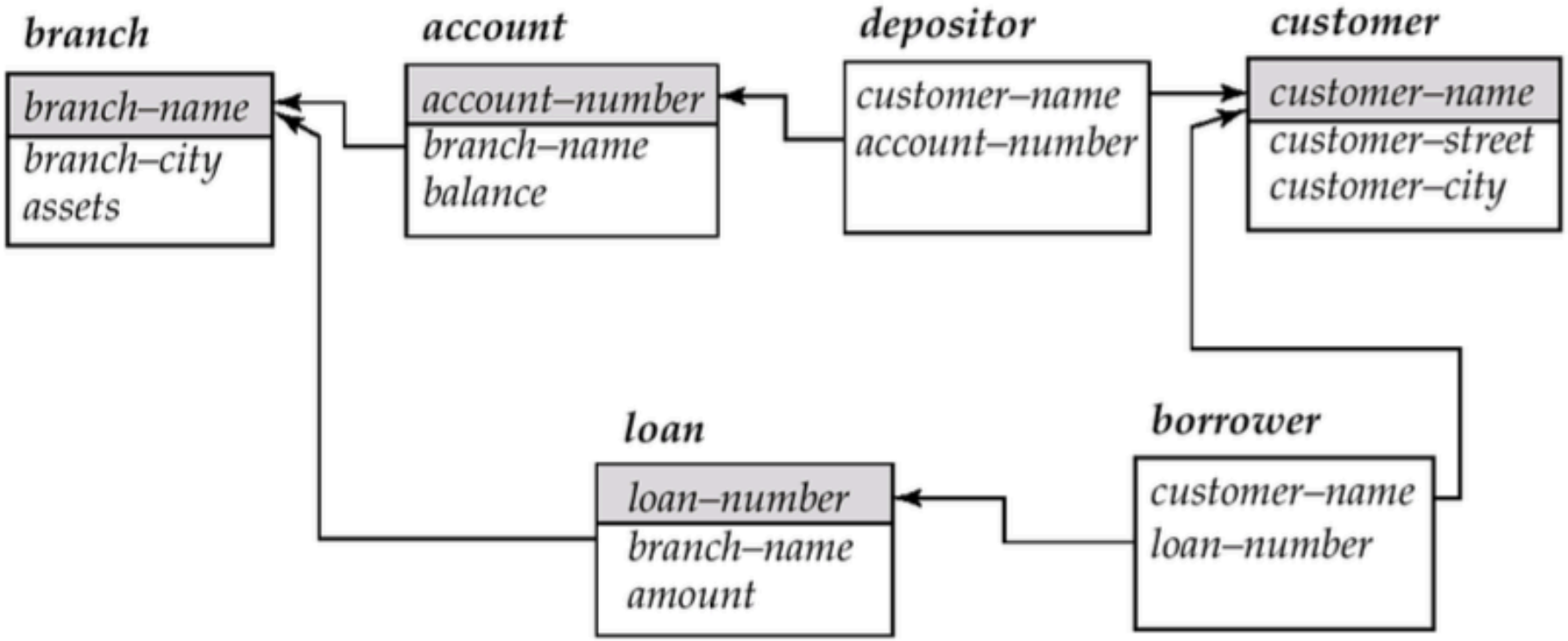
Example of relation instance

Account-schema = (account-number, branch-name, balance)

account-number	*branch-name*	*balance*
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

Relational database

Relational database consists of multiple relations which are inter-related



Query Languages

Language in which user requests information from the database.

Categories of languages

- procedural
- non-procedural

“Pure” languages:

- Relational Algebra
- Relational Calculus

Pure languages form underlying basis of query languages.

Relational algebra

Procedural language

Six basic operators: select, project, union, set difference, Cartesian product, rename

Additional operations: set intersection, natural join, θ-join, division, generalized projection, assignment, aggregation, outer joins, ...

The operators take one or more relations as inputs
and give a new relation as a result.

In Relational Algebra all inputs and outputs are relations. They are sets of tuples.

In SQL the output of an operator is a multiset of tuples

Select Operation

Notation: $\sigma_p(r)$

p is called the selection predicate

Defined as: $\sigma_p(r) = \{t | t \in r \; and \; p(t)\}$

Where p is a formula in propositional calculus consisting of terms connected by : and, or, not operations

Each term is one of:

<attribute> op <attribute> or <constant>

where op is one of: $=, \neq, >, \geq, <, \leq$

Select Operation – Example

Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

$\sigma_{A=B \wedge D>5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Project Operation

Notation: $\Pi_{A_1,A_2,\ldots A_k}(\mathrm{r})$
where A_1, A_2 … are attribute names and r is a relation name.

The result is defined as the relation of k columns obtained by erasing the columns that are not listed

Duplicate rows removed from result, since relations are sets

Union Operation

Notation: $r \cup s$

Defined as:
$r \cup s = \{t | t \in r \; or \; t \in s\}$

Union must be taken between compatible relations.

- r and s must have the same arity
- attribute domains of r and s must be compatible

Set-union Operation – Example

Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

r ∪ s:

A	B
α	1
α	2
β	1
β	3

Intersection Operation

Notation $r \cap s$

Defined as:

$r \cap s = \{t | t \in r \; and \; t \in s\}$

Intersection must be taken between compatible relations.

- r and s must have the same arity
- attribute domains of r and s must be compatible

Set-Intersection Operation - Example

Relation r, s:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

r ∩ s

A	B
α	2

Set Difference Operation

Notation r – s

Defined as:
$r - s = \{t | t \in r \; and \; t \notin s\}$

Set differences must be taken between compatible relations.

- r and s must have the same arity
- attribute domains of r and s must be compatible

Set Difference Operation – Example

Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

r – *s*:

A	B
α	1
β	1

Cartesian-Product Operation

Notation r x s

Defined as:
$r \times s = \{t\ q \mid t \in r \text{ and } q \in s\}$

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Composition of Operations

We can easily build expressions using multiple operations

$r \times s$

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

$\sigma_{A=C}(r \times s)$

A	B	C	D	E
α	1	α	10	a
β	2	β	20	a
β	2	β	20	b

Aggregate Functions and Operations

An aggregation function takes a collection of values and returns a single value as a result.

avg: average value
min: minimum value
max: maximum value
sum: sum of values
count: number of values

- Aggregate operation in relational algebra

$\mathrm{G1}, G2, \ldots Gn\ \mathbf{g}\ \mathrm{F1}(A1), F2(A2), \ldots Fn(An)\ (E)$

- E is any relational-algebra expression
- G1, G2 ..., Gn is a list of attributes on which to group (can be empty)
- Each Fi is an aggregate function
- Each Ai is an attribute name

Aggregate Operation – Example

Relation *r*:

A	*B*	*C*
α	α	7
α	β	7
β	β	3
β	β	10

$$g_{\text{sum(c)}}(\text{r})$$

sum-C
27

Relation *account* grouped by *branch-name*:

branch-name	*account-number*	*balance*
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

$$branch\text{-}name\ \mathcal{G}_{sum(balance)}\ (account)$$

branch-name	*balance*
Perryridge	1300
Brighton	1500
Redwood	700

SQL: Basic Structure

SQL is based on set and relational operations with certain modifications and enhancements

A typical SQL query has the form:

select $A_1, A_2, \ldots A_n$
from $r_1, r_2 \ldots r_m$
where P

$A_i\ represents\ attributes$
$r_i\ represents\ relations$
$P\ is\ a\ predicate$

This query is equivalent to the relational algebra expression
$\Pi_{A_1,A_2,\ldots A_n}(\sigma_P(r_1 \times r_2 \times \cdots r_m))$

The result of an SQL query is a multiset of tuples.

The select Clause

SQL allows duplicates in relations as well as in query results.

To force the elimination of duplicates, insert the keyword distinct After select.

Q1: Find the names of all branches in the loan relations, and remove duplicates

```
select distinct branch-name
from loan
```

Q2: If distinct is not used duplicates are not removed.

```
select branch-name
from loan
```

loan

loan-number	*branch-name*	*amount*
L-170	Downtown	3000
L-220	Downtown	4000
L-260	Perryridge	1700

Q2

branch-name
Downtown
Downtown
Perryridge

Q1

branch-name
Downtown
Perryridge

The where Clause

The where clause specifies conditions that the result must satisfy corresponds to the selection predicate of the relational algebra.

To find all loan number for loans made at the Perryridge branch with loan amounts greater than $1200.

```
select loan-number
from loan
where branch-name = ‘Perryridge’ and amount > 1200
```

Comparison results can be combined using the logical connectives and, or, and not.

Comparisons can be applied to results of arithmetic expressions.

The from Clause

The from clause lists the relations involved in the query corresponds to the Cartesian product operation of the relational algebra.

Find the Cartesian product borrower x loan

```
select *
from borrower, loan
```

Aggregate Functions – Group By

Find the number of depositors for each branch.

```
select branch-name, count (distinct customer-name)
        from depositor, account
        where depositor.account-number = account.account-number
        group by branch-name
```

Aggregate Functions – Having Clause

Find the names and average account balances of all branches where the average account balance is more than $1,200.

```
select branch-name, avg (balance)
from account
group by branch-name
having avg (balance) > 1200
```

Distributed database

A distributed database is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network.

Distributed DBMS is software system that permits the management of the distributed database and makes the distribution transparent to users

Factors Encouraging DDBMS

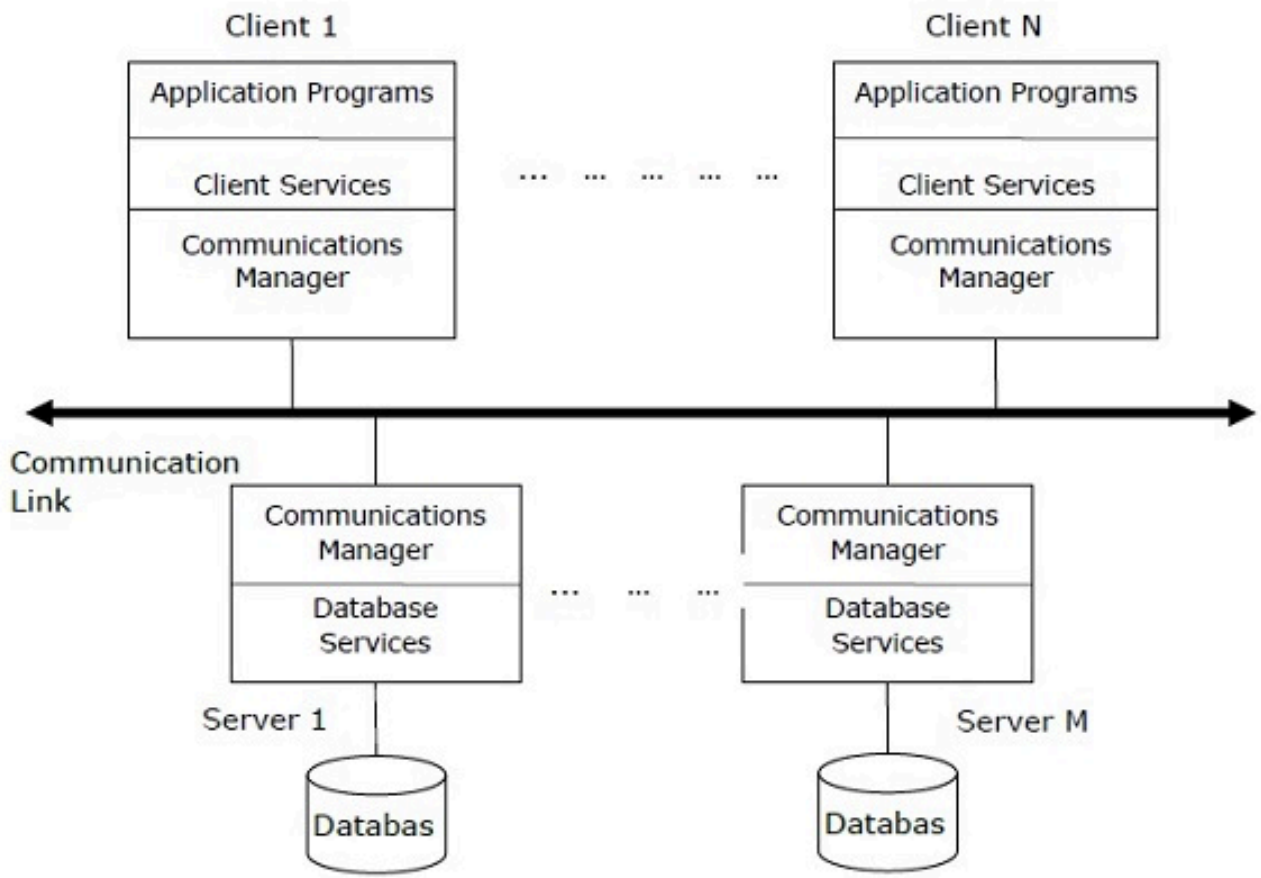
- Distributed Nature of Organizational Units
- Need for Sharing of Data
- Support for Both online transaction processing and analytical processing
- Database Recovery
- Support for Multiple Application Software

Advantages of Distributed Databases

- Modular Development
- More Reliable
- Better Response
- Lower Communication Cost

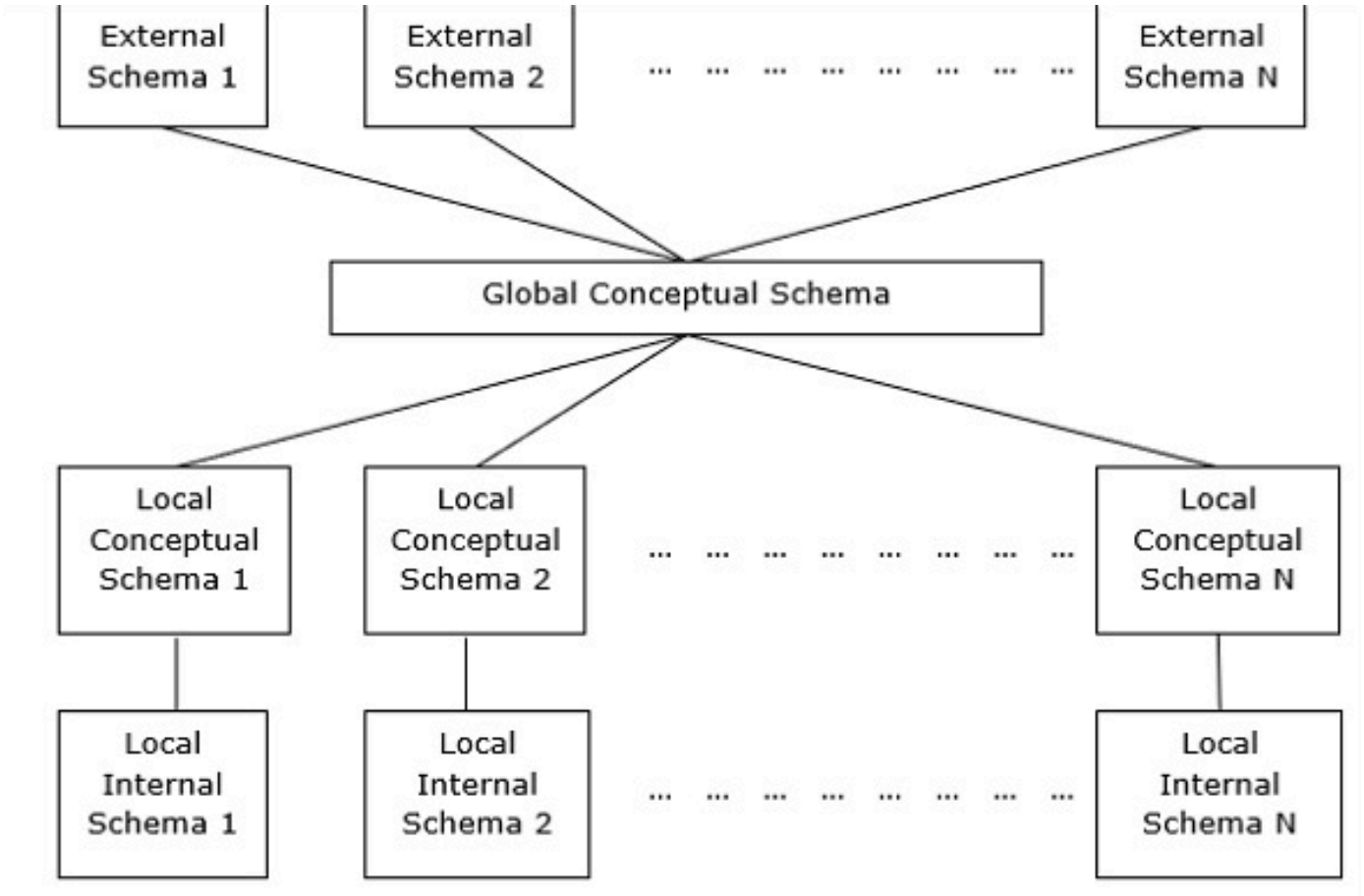
Client - Server Architecture for DDBMS

This is a two-level architecture where the functionality is divided into servers and clients. The server functions primarily encompass data management, query processing, optimization and transaction management. Client functions include mainly user interface. However, they have some functions like consistency checking and transaction management.


Client 1
Application Programs
Client Services
Communications Manager
Client N
Application Programs
Client Services
Communications Manager
Communication Link
Communications Manager
Database Services
Communications Manager
Database Services
Server 1
Server M
Databas
Databas

Peer- to-Peer Architecture for DDBMS

In these systems, each peer acts both as a client and a server for imparting database services. The peers share their resource with other peers and co-ordinate their activities.


External Schema 1
External Schema 2
External Schema N
Global Conceptual Schema
Local Conceptual Schema 1
Local Conceptual Schema 2
Local Conceptual Schema N
Local Internal Schema 1
Local Internal Schema 2
Local Internal Schema N

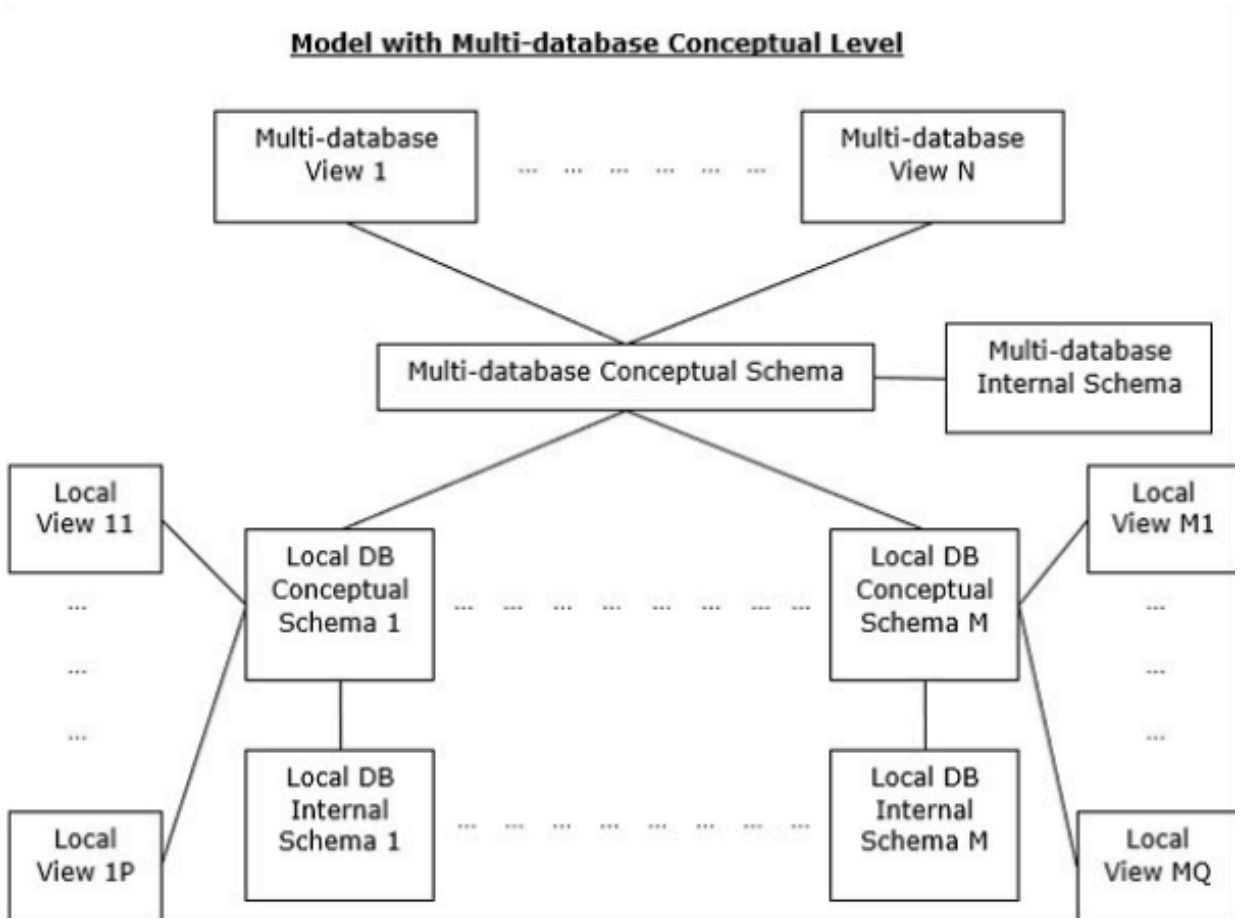
Multi - DBMS Architectures

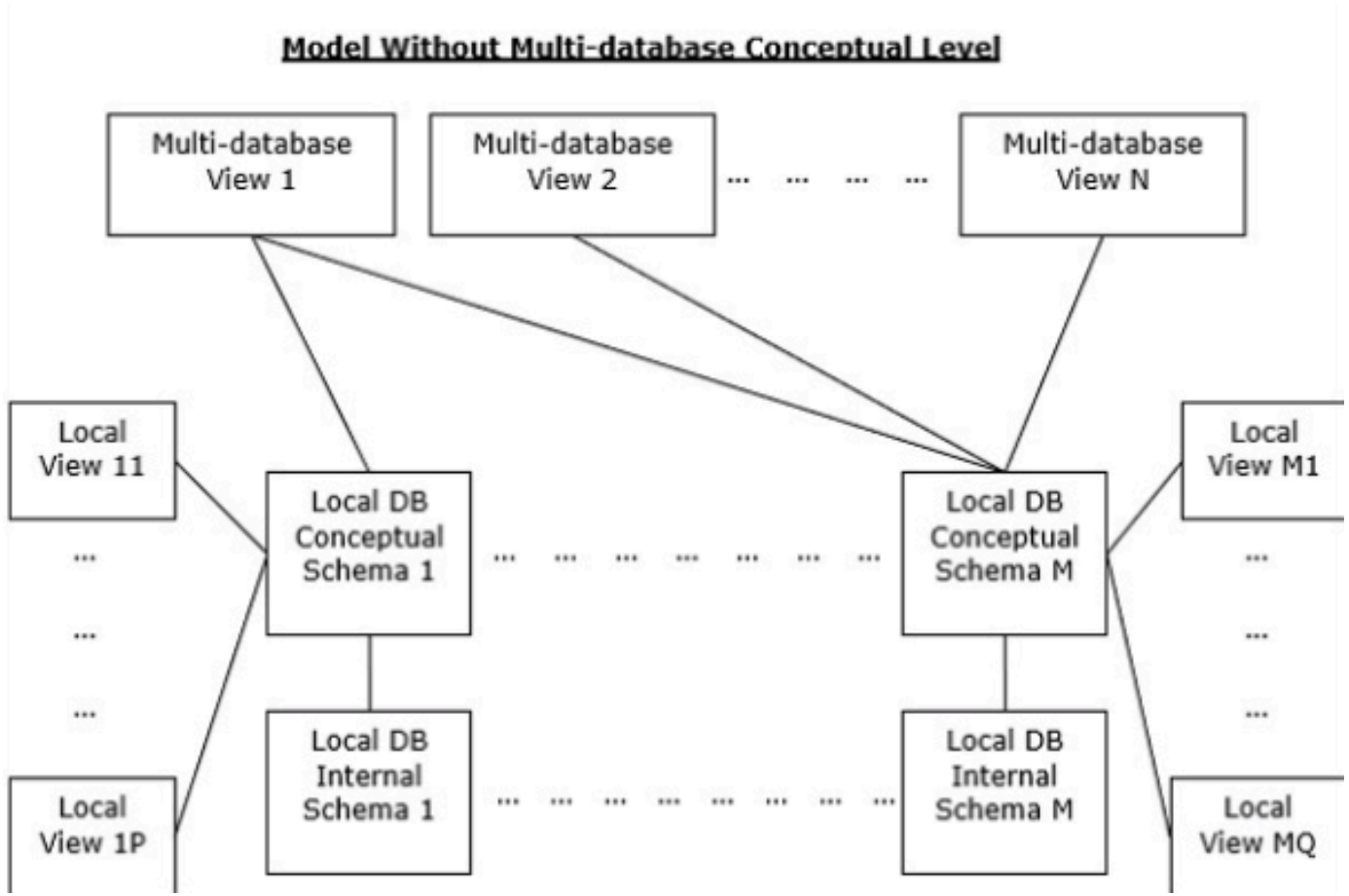
This is an integrated database system formed by a collection of two or more autonomous database systems.

There are two design alternatives for multi-DBMS

- Model with multi-database conceptual level
- Model without multi-database conceptual level

Model with Multi-database Conceptual Level




Model Without Multi-database Conceptual Level
Multi-database View 1
Multi-database View 2
Multi-database View N
Local View 11
Local View 1P
Local DB Conceptual Schema 1
Local DB Internal Schema 1
Local DB Conceptual Schema M
Local DB Internal Schema M
Local View M1
Local View MQ

Distributed Database Design

Three key issues:

- Fragmentation.

Relation may be divided into a number of sub- relations, which are then distributed.

- Allocation

Each fragment is stored at site with "optimal" distribution.

- Replication

Copy of fragment may be maintained at several sites.

Distribution transparency

Distribution transparency is the property of distributed databases by the virtue of which the internal details of the distribution are hidden from the users. The DDBMS designer may choose to fragment tables, replicate the fragments and store them at different sites. However, since users are oblivious of these details, they find the distributed database easy to use like any centralized database.

Database control

Database control refers to the task of enforcing regulations so as to provide correct data to authentic users and applications of a database. In order that correct data is available to users, all data should conform to the integrity constraints defined in the database. Besides, data should be screened away from unauthorized users so as to maintain security and privacy of the database. Database control is one of the primary tasks of the database administrator (DBA).

- Authentication

authentication is the process through which only legitimate users can gain access to the data resources

- Access rights

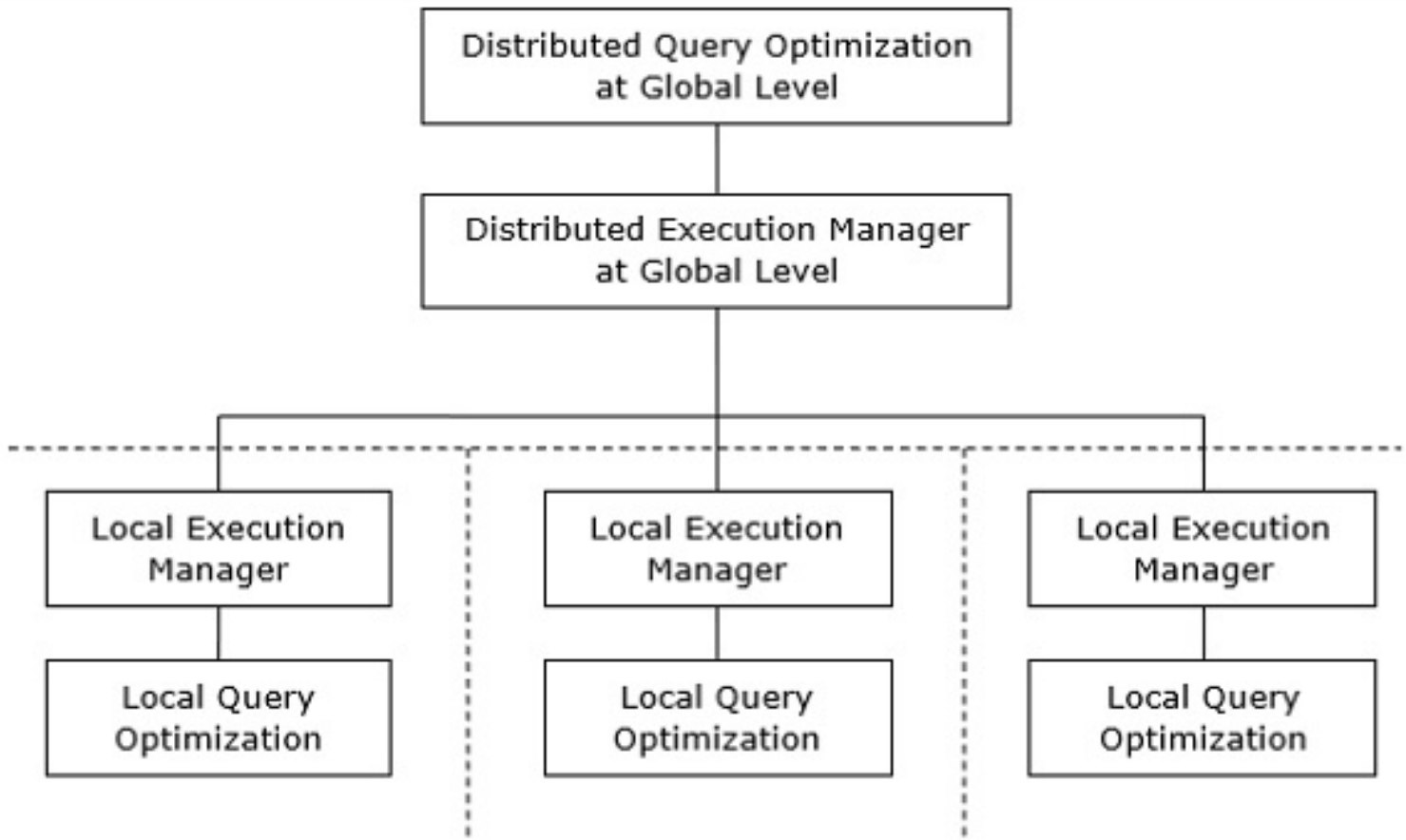
A user’s access rights refers to the privileges that the user is given regarding DBMS operations such as the rights to create a table, drop a table, add/delete/update tuples in a table or query upon the table

- Integrity constraints

Semantic integrity control defines and enforces the integrity constraints of the database system.

Query optimization

When a query is placed, it is at first scanned, parsed and validated. An internal representation of the query is then created such as a query tree or a query graph. Then alternative execution strategies are devised for retrieving results from the database tables. The process of choosing the most appropriate execution strategy for query processing is called query optimization.


Distributed Query Optimization at Global Level
Distributed Execution Manager at Global Level
Local Execution Manager
Local Execution Manager
Local Execution Manager
Local Query Optimization
Local Query Optimization
Local Query Optimization

Transaction

A transaction is a program including a collection of database operations, executed as a logical unit of data processing. The operations performed in a transaction include one or more of database operations like insert, delete, update or retrieve data. It is an atomic process that is either performed into completion entirely or is not performed at all. A transaction involving only data retrieval without any data update is called read-only transaction.

Transaction operations

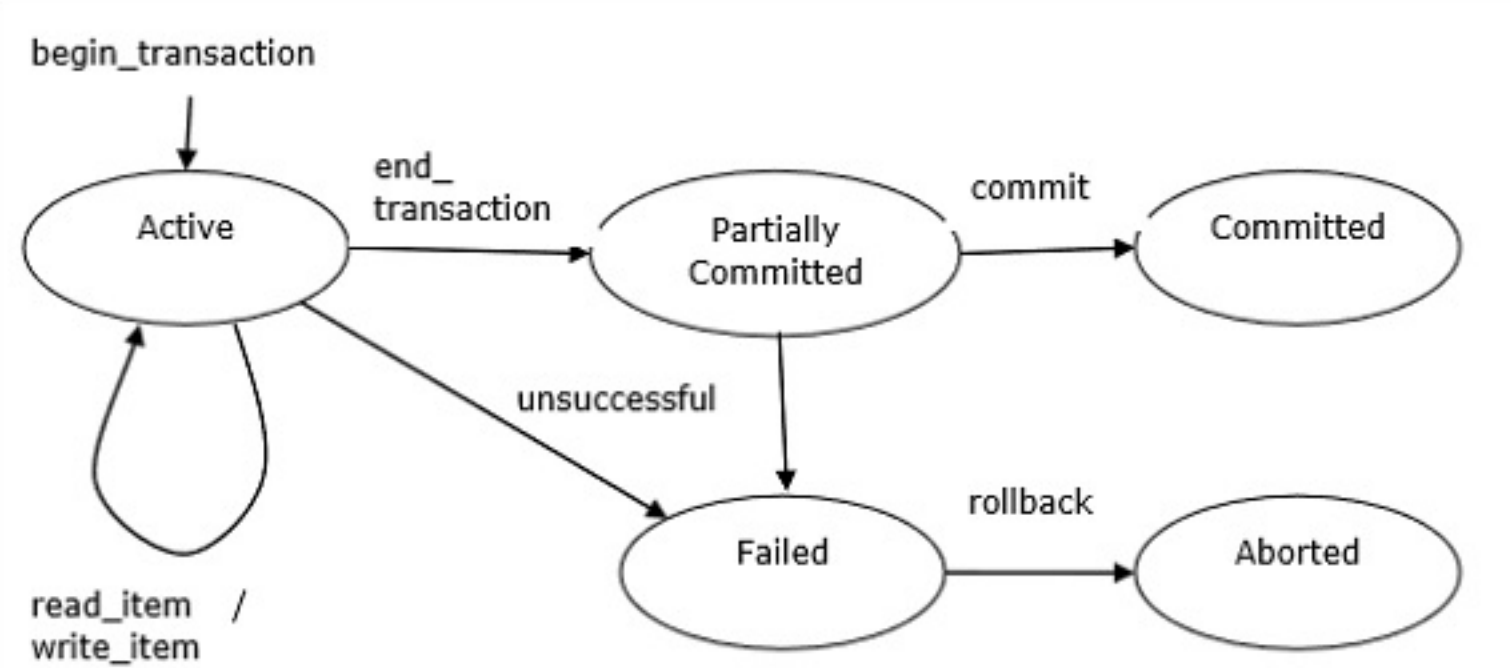
- begin_transaction − A marker that specifies start of transaction execution
- read_item or write_item − Database operations that may be interleaved with main memory operations as a part of transaction
- end_transaction − A marker that specifies end of transaction

- commit − A signal to specify that the transaction has been successfully completed in its entirety and will not be undone
- rollback − A signal to specify that the transaction has been unsuccessful and so all temporary changes in the database are undone. A committed transaction cannot be rolled back.

Transaction States

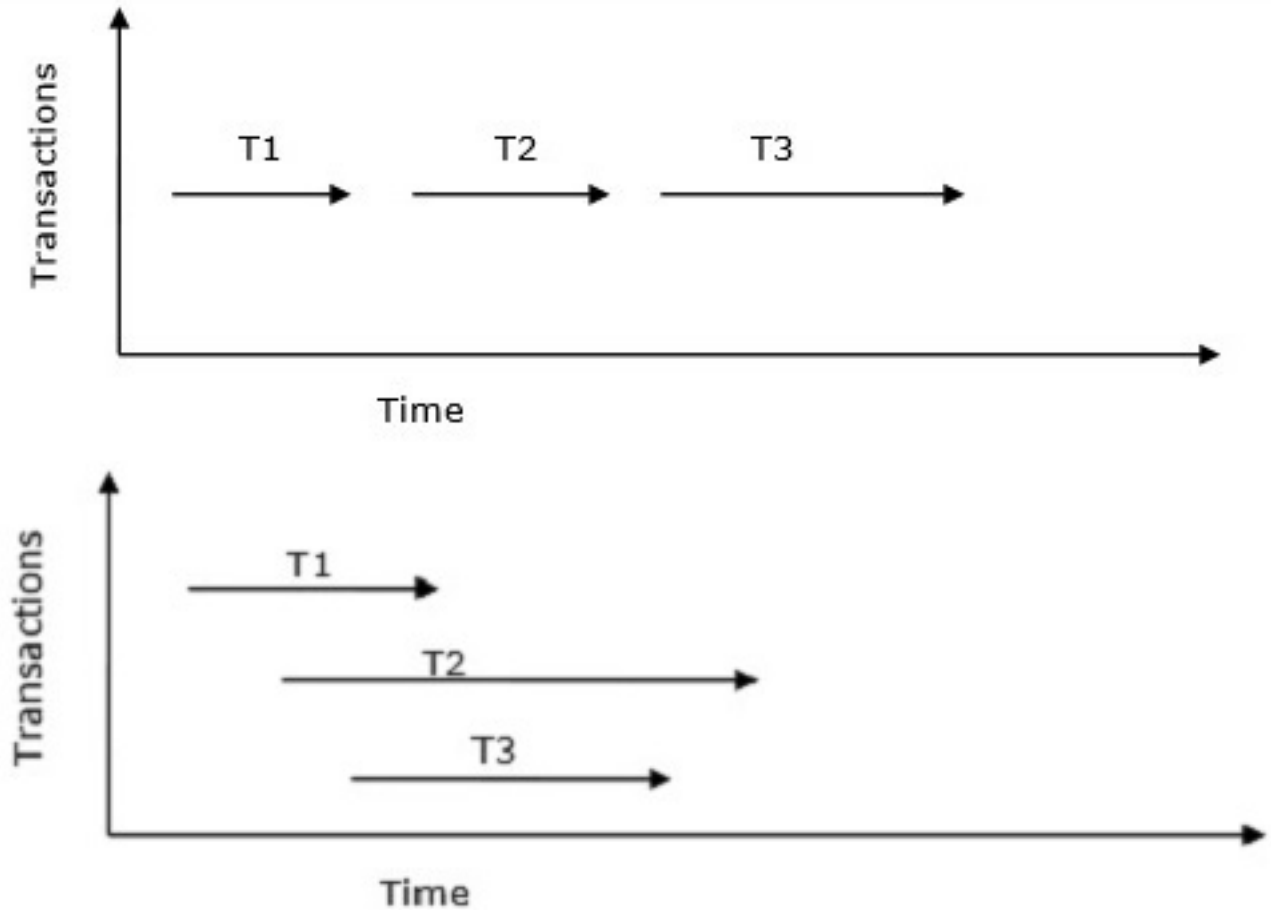
- Active − The initial state where the transaction enters is the active state. The transaction remains in this state while it is executing read, write or other operations.
- Partially Committed − The transaction enters this state after the last statement of the transaction has been executed.
- Committed − The transaction enters this state after successful completion of the transaction and system checks have issued commit signal.

- Failed − The transaction goes from partially committed state or active state to failed state when it is discovered that normal execution can no longer proceed or system checks fail.
- Aborted − This is the state after the transaction has been rolled back after failure and the database has been restored to its state that was before the transaction began.


begin_transaction
Active
end_
transaction
Partially
Committed
commit
Committed
unsuccessful
Failed
rollback
Aborted
read_item /
write_item

Schedules and Conflicts

In a system with a number of simultaneous transactions, a schedule is the total order of execution of operations. Given a schedule S comprising of n transactions, say T1, T2, T3…….Tn; for any transaction Ti, the operations in Ti must execute as laid down in the schedule S.


Transactions
T1
T2
T3
Time
Transactions
T1
T2
T3
Time

In a schedule comprising of multiple transactions, a conflict occurs when two active transactions perform non-compatible operations. Two operations are said to be in conflict, when all of the following three conditions exists simultaneously

- The two operations are parts of different transactions.
- Both the operations access the same data item.
- At least one of the operations is a write_item() operation, i.e. it tries to modify the data item.

Desirable Properties of Transactions

- Atomicity

An atomic transaction is an indivisible and irreducible series of database operations such that either all occur, or nothing occurs.

- Consistency

Any given database transaction must change affected data only in allowed ways

- Isolation

Isolation determines how transaction integrity is visible to other users and systems.

- Durability

Transactions that have committed will survive permanently.

Serializability

A serializable schedule of ‘n’ transactions is a parallel schedule which is equivalent to a serial schedule comprising of the same ‘n’ transactions. A serializable schedule contains the correctness of serial schedule while ascertaining better CPU utilization of parallel schedule.

Controlling Concurrency

Concurrency controlling techniques ensure that multiple transactions are executed simultaneously while maintaining the ACID properties of the transactions and serializability in the schedules.

Thank you for your attention!!!