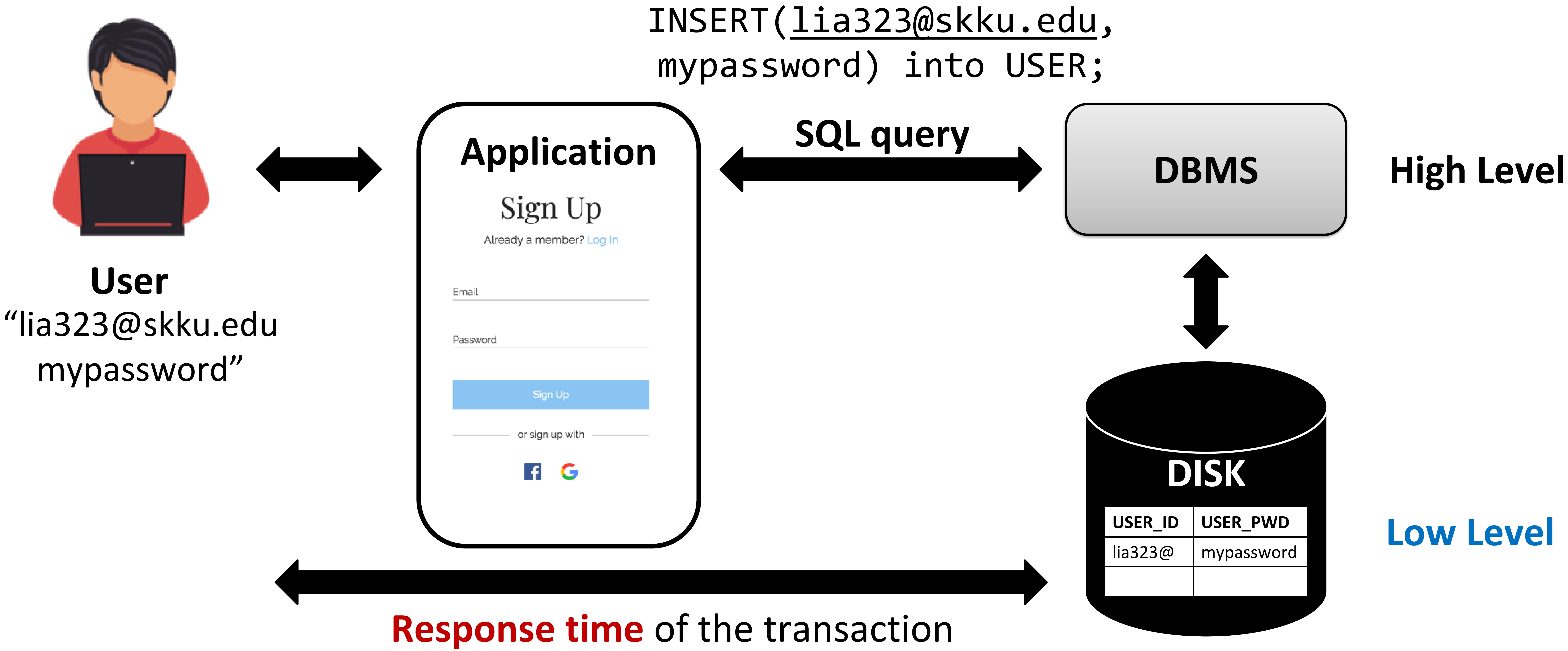
SWE3033:
Database Project

Run TPC-C and Analyze the Results

Bo-Hyun Lee
lia323@skku.edu

DBMS Performance



User
“lia323@skku.edu
mypassword”

Application

```
INSERT(lia323@skku.edu,
mypassword) into USER;
```

SQL query

DBMS

High Level

DISK

USER_ID	USER_PWD
lia323@	mypassword

Low Level

Response time of the transaction

Benchmarking: What and Why

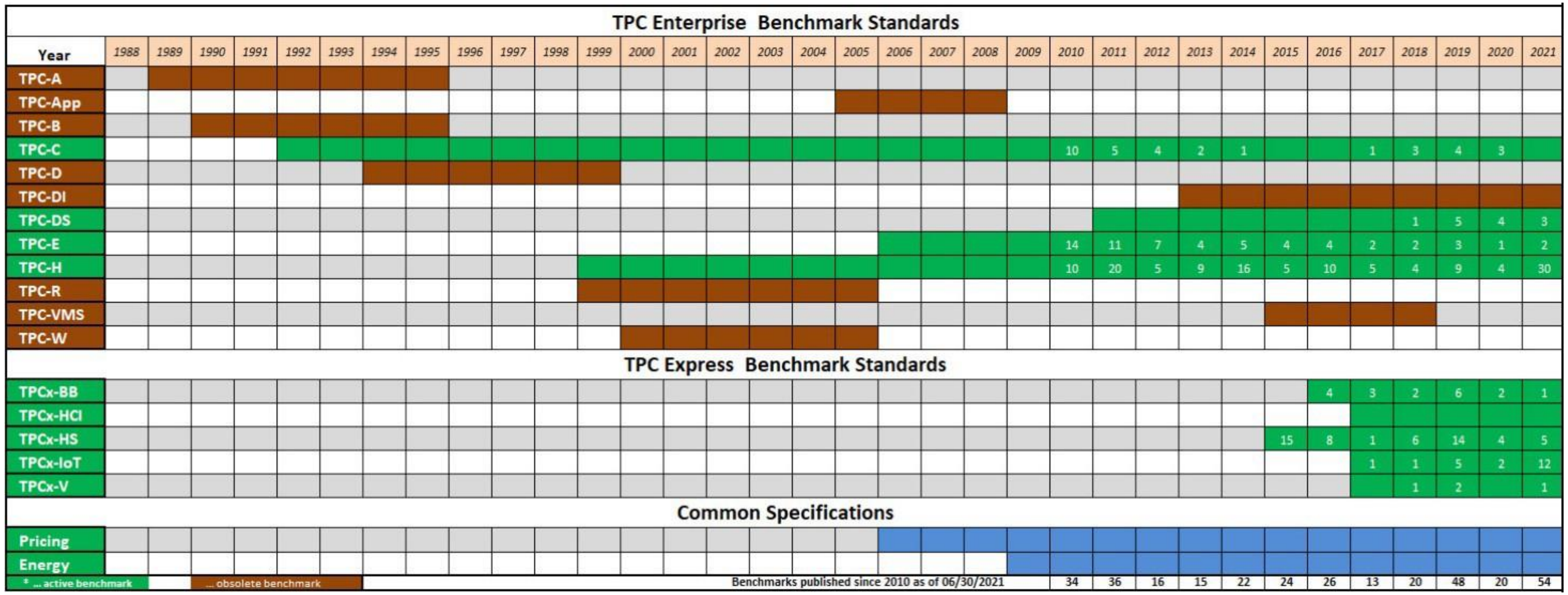
- Benchmarking is a process of **measuring** the performance of a given application and **comparing** it with other similar workloads
  - to discover if there is a performance gap
  - to improve its performance

- Benchmark is *domain-specific*:
  - The more general the benchmark, the less useful it is
  - A benchmark is a distillation of the essential attributes of a particular workload

- Desirable attributes:
  - Relevant ➔ It should be meaningful within the target domain
  - Understandable/acceptable ➔ Vendors and users embrace it
  - Scalable/coverable ➔ It should not oversimplify the typical environment

What is the TPC?

- TPC = Transaction Processing Performance Council
  - Founded in Aug 1988 by Omri Serlin and 8 vendors
- De facto industry-standards body for enterprise benchmark



TPC Enterprise Benchmark Standards

Year	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
TPC-A		🟫	🟫	🟫	🟫	🟫	🟫	🟫																										
TPC-App																		🟫	🟫	🟫	🟫													
TPC-B			🟫	🟫	🟫	🟫	🟫	🟫																										
TPC-C					🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	10	5	4	2	1	🟩	🟩	1	3	4	3	🟩
TPC-D							🟫	🟫	🟫	🟫	🟫	🟫																						
TPC-DI																										🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫
TPC-DS																								🟩	🟩	🟩	🟩	🟩	🟩	🟩	1	5	4	3
TPC-E																			🟩	🟩	🟩	🟩	14	11	7	4	5	4	4	2	2	3	1	2
TPC-H												🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	10	20	5	9	16	5	10	5	4	9	4	30
TPC-R												🟫	🟫	🟫	🟫	🟫	🟫	🟫																
TPC-VMS																												🟫	🟫	🟫	🟫			
TPC-W													🟫	🟫	🟫	🟫	🟫	🟫																
TPC Express Benchmark Standards																																		
TPCx-BB																													4	3	2	6	2	1
TPCx-HCI																														🟩	🟩	🟩	🟩	🟩
TPCx-HS																												15	8	1	6	14	4	5
TPCx-IoT																														1	1	5	2	12
TPCx-V																														🟩	1	2	🟩	1
Common Specifications																																		
Pricing																			🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Energy																						🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
🟩 * ... active benchmark			🟫 ... obsolete benchmark													Benchmarks published since 2010 as of 06/30/2021							34	36	16	15	22	24	26	13	20	48	20	54

TPC-C

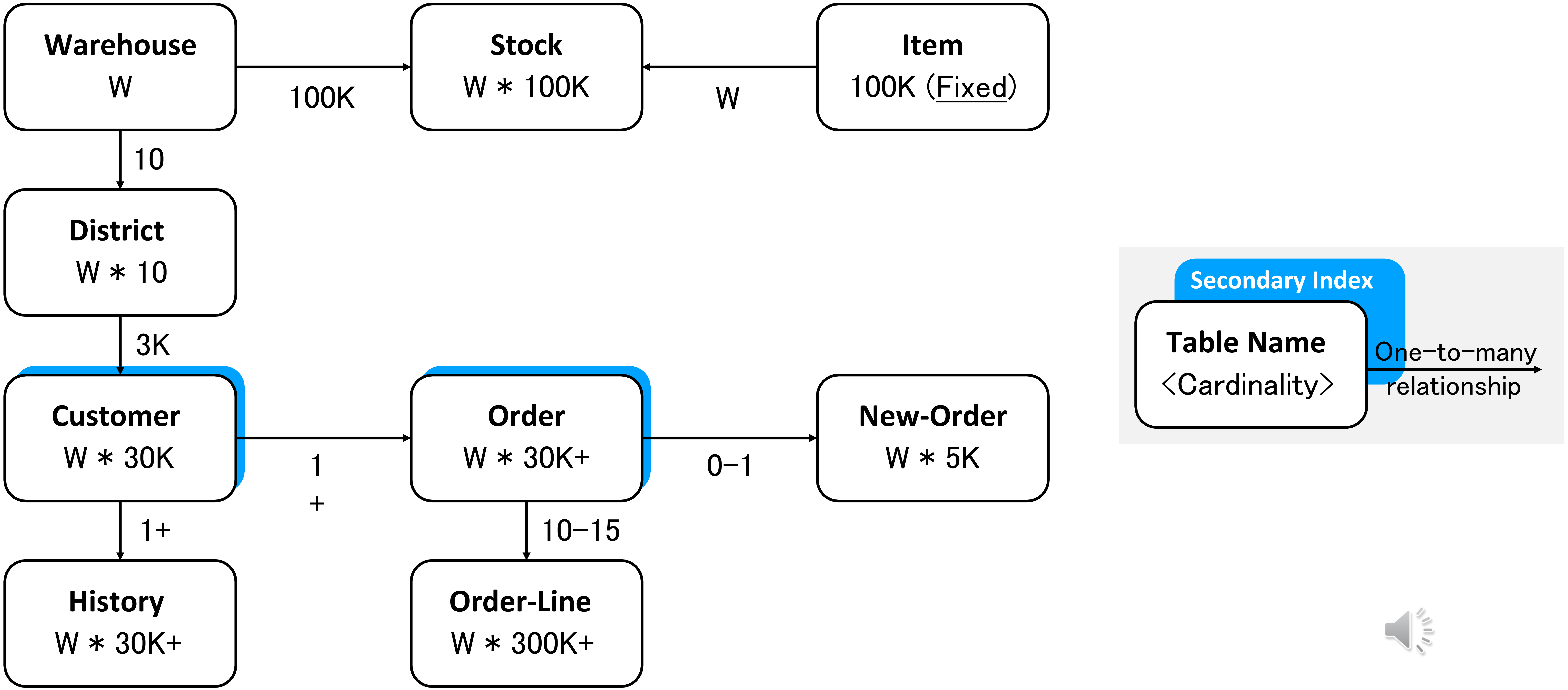
- A 29-year-old **industry-standard OLTP benchmark** **used to measure the performance of databases**
  - It simulates an *e-commerce* or *retail* company

- **5** types of well-defined transactions:
  - New-Order (Read/Write), Payment (Read/Write), Delivery (Read/Write), Order-Status (Read Only), Stock-Level (Read Only)

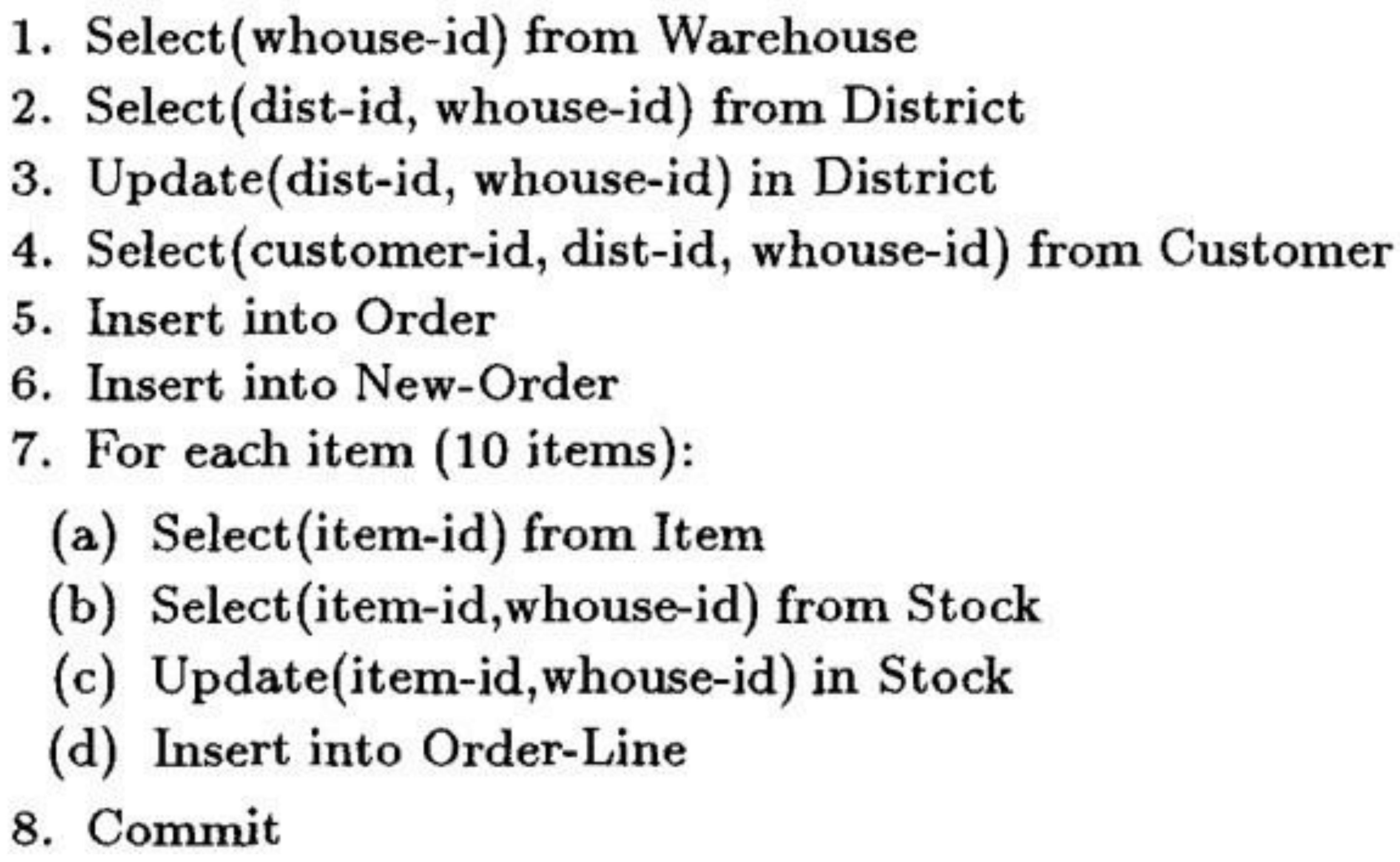
- **Throughput of TPC-C** = The number of **New-Order** transactions executed per minute
  - Transactions per minute Count (TpmC)

- **Random I/O intensive** workload
  - 65% Reads, 35% writes

TPC-C Entity/Relationship Diagram



New Order Transaction



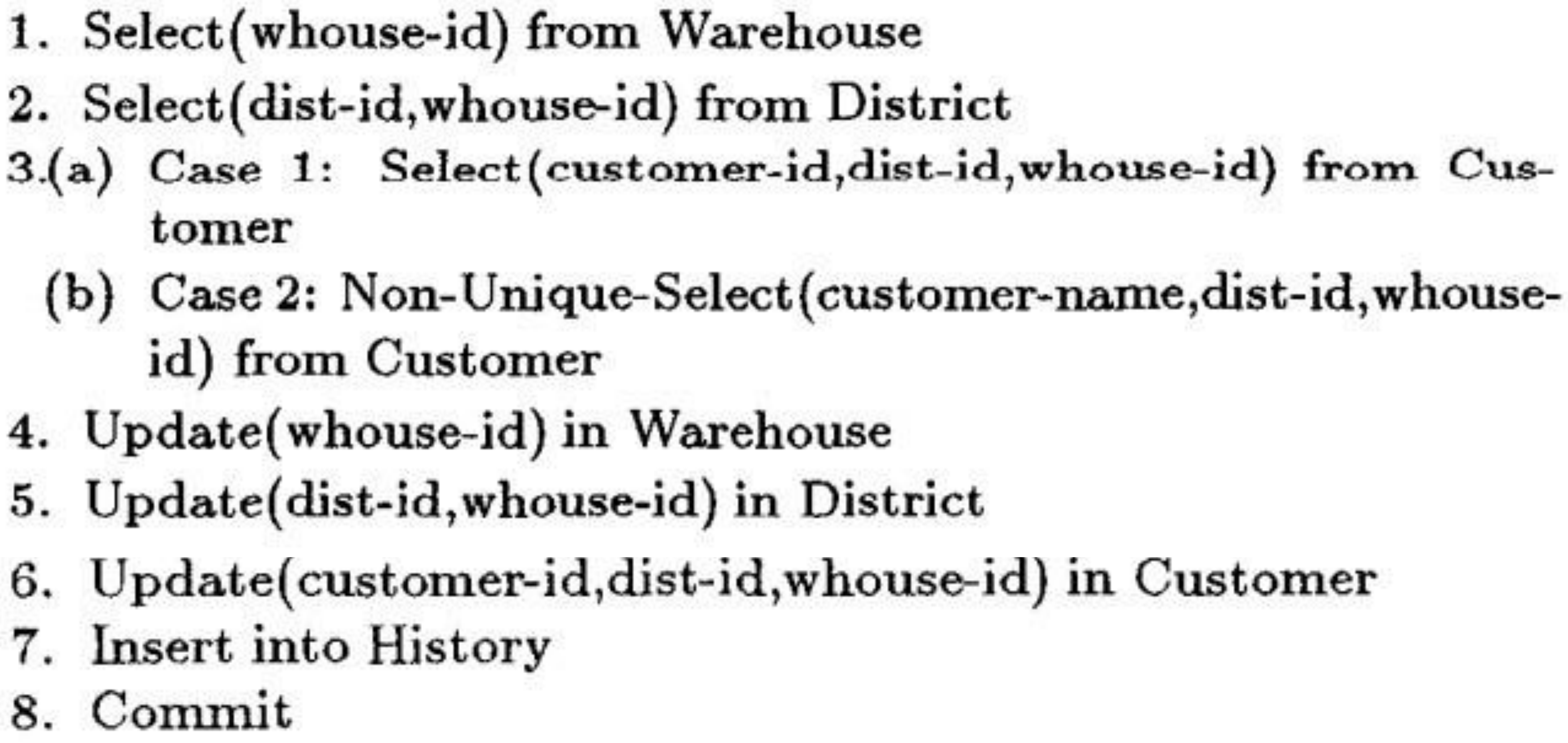
```
1. Select(whouse-id) from Warehouse
2. Select(dist-id, whouse-id) from District
3. Update(dist-id, whouse-id) in District
4. Select(customer-id, dist-id, whouse-id) from Customer
5. Insert into Order
6. Insert into New-Order
7. For each item (10 items):
   (a) Select(item-id) from Item
   (b) Select(item-id,whouse-id) from Stock
   (c) Update(item-id,whouse-id) in Stock
   (d) Insert into Order-Line
8. Commit
```

Place an order for on average 10 items from a warehouse

Insert the order

Update the corresponding stock level for each item

Payment Transaction

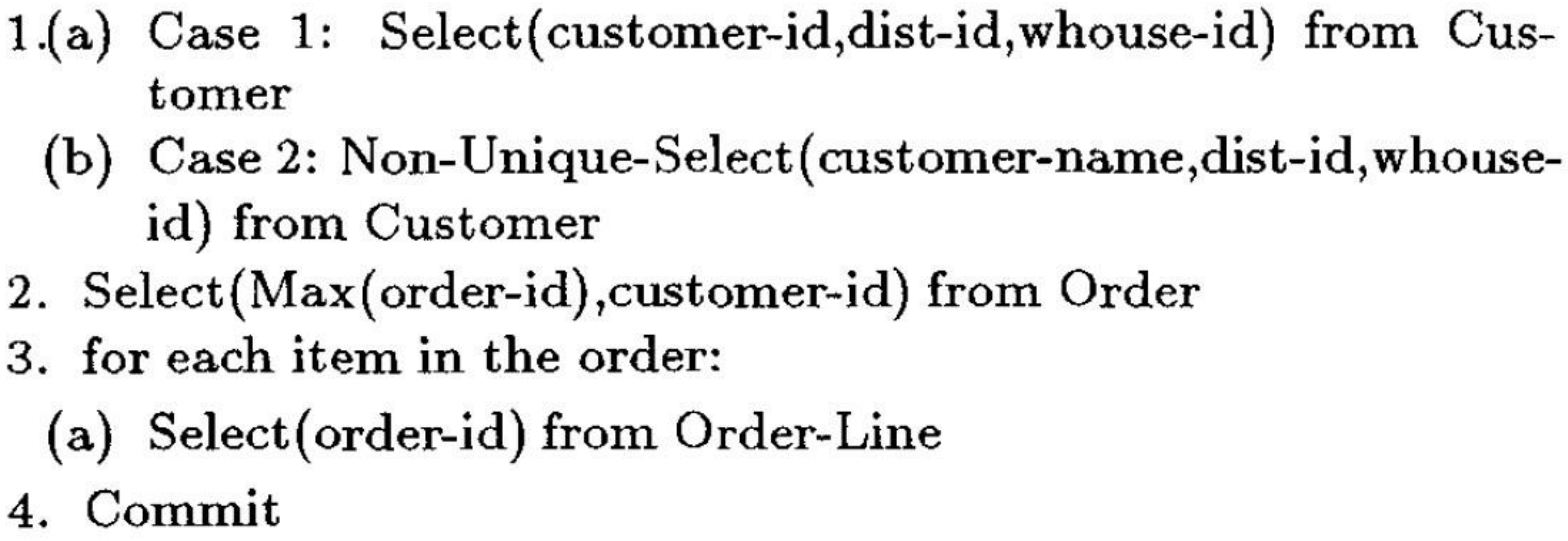


1. Select(whouse-id) from Warehouse
2. Select(dist-id,whouse-id) from District
3. (a) Case 1: Select(customer-id,dist-id,whouse-id) from Customer
   (b) Case 2: Non-Unique-Select(customer-name,dist-id,whouse-id) from Customer
4. Update(whouse-id) in Warehouse
5. Update(dist-id,whouse-id) in District
6. Update(customer-id,dist-id,whouse-id) in Customer
7. Insert into History
8. Commit

Process a payment for a customer

Update balances and other data

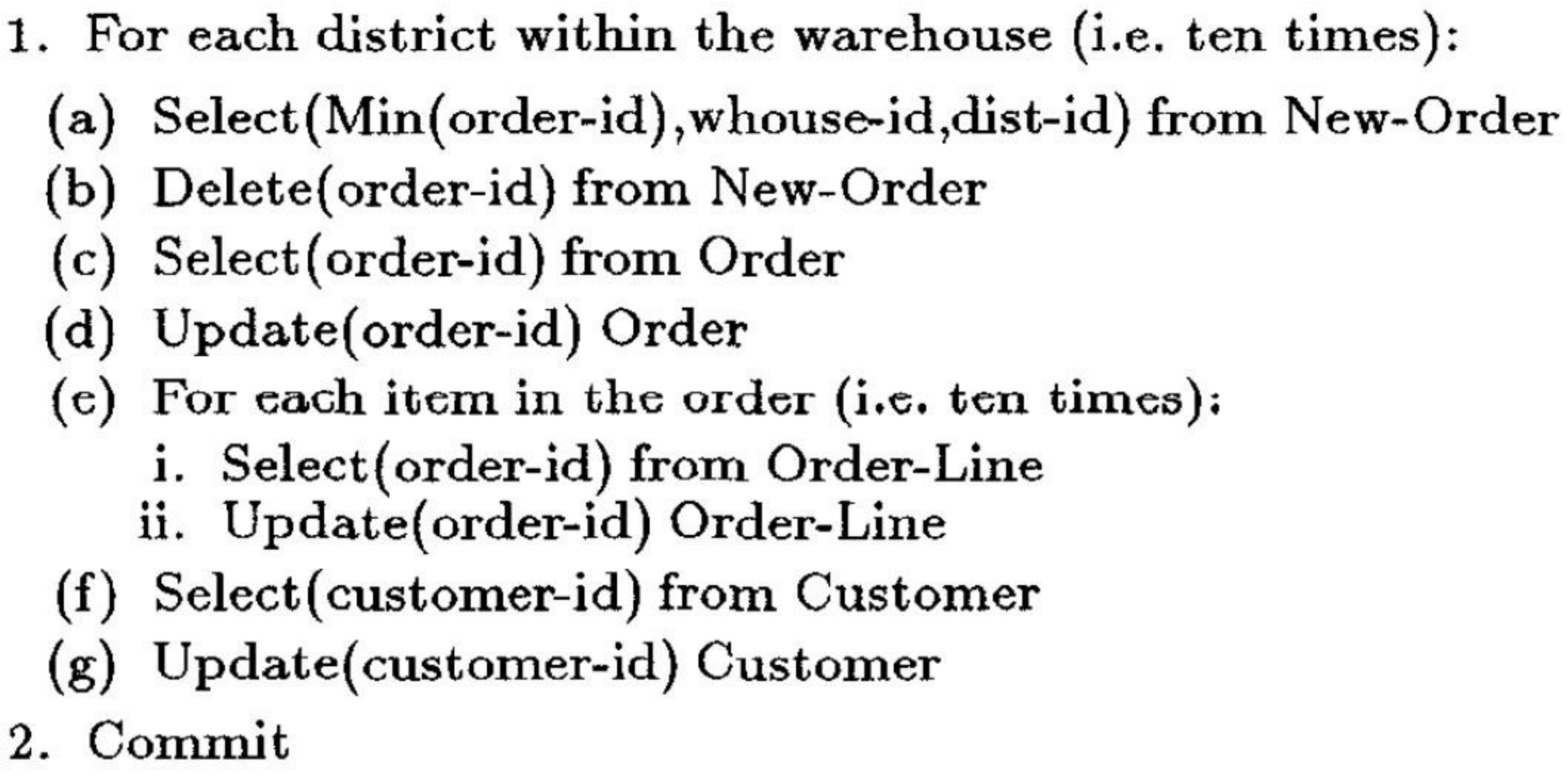
Order Status Transaction



```
1.(a) Case 1: Select(customer-id,dist-id,whouse-id) from Customer
  (b) Case 2: Non-Unique-Select(customer-name,dist-id,whouse-id) from Customer
2. Select(Max(order-id),customer-id) from Order
3. for each item in the order:
  (a) Select(order-id) from Order-Line
4. Commit
```

Return the status of a customer’s last order

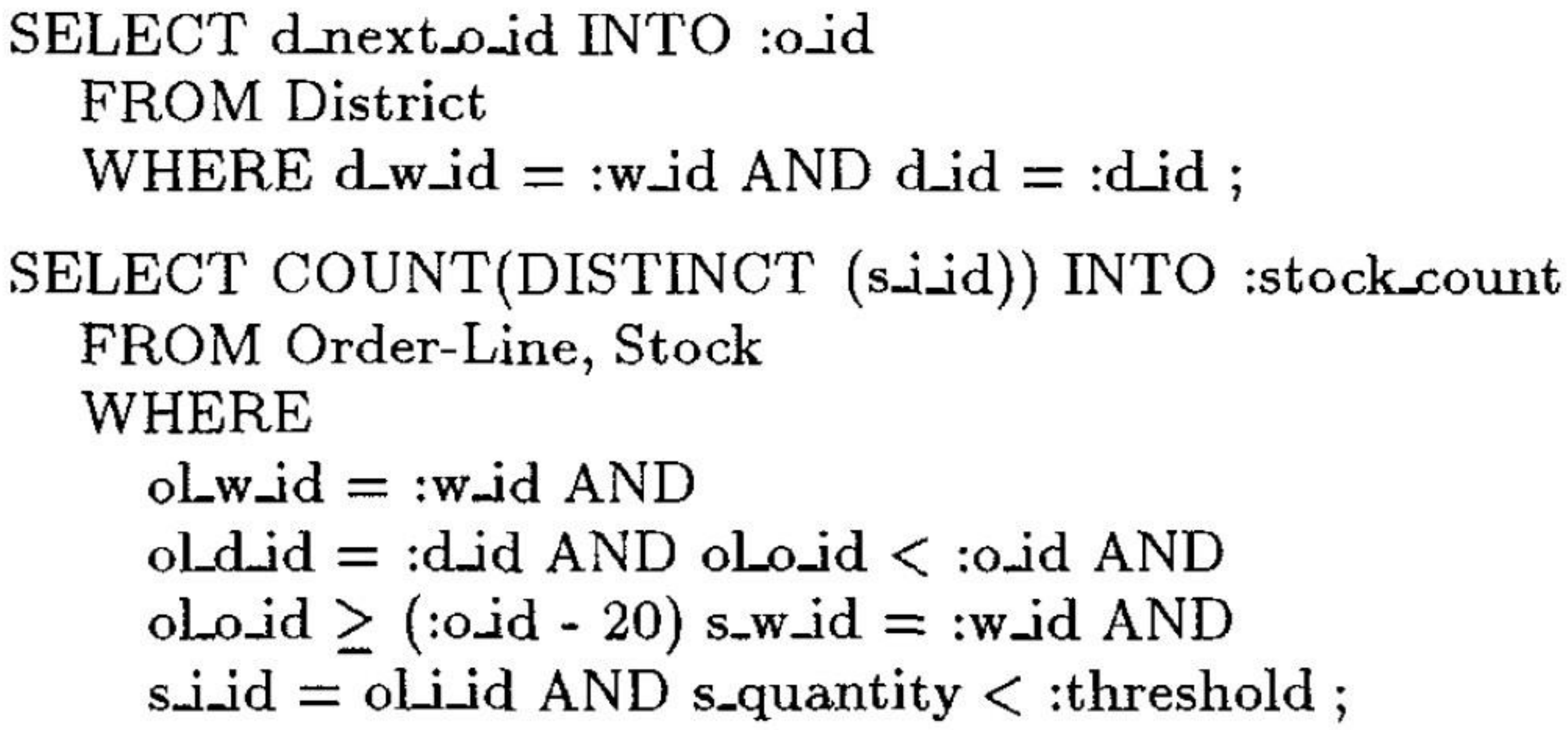
Delivery Transaction



```
1. For each district within the warehouse (i.e. ten times):
  (a) Select(Min(order-id),whouse-id,dist-id) from New-Order
  (b) Delete(order-id) from New-Order
  (c) Select(order-id) from Order
  (d) Update(order-id) Order
  (e) For each item in the order (i.e. ten times):
       i. Select(order-id) from Order-Line
      ii. Update(order-id) Order-Line
  (f) Select(customer-id) from Customer
  (g) Update(customer-id) Customer
2. Commit
```

Process orders corresponding to 10 pending orders, one for each district, with 10 items per order

Stock Level Transaction



```
SELECT d_next_o_id INTO :o_id
   FROM District
   WHERE d_w_id = :w_id AND d_id = :d_id ;

SELECT COUNT(DISTINCT (s_i_id)) INTO :stock_count
   FROM Order-Line, Stock
   WHERE
      ol_w_id = :w_id AND
      ol_d_id = :d_id AND ol_o_id < :o_id AND
      ol_o_id ≥ (:o_id - 20) s_w_id = :w_id AND
      s_i_id = ol_i_id AND s_quantity < :threshold ;
```

Examine the quantity of stock for the items ordered by each of the last 20 orders in a district

Relation Access Pattern: S/I/U/D

Relation Name	New Order	Payment	Order Status	Delivery	Stock Level	Comment

Warehouse	Select(1)	Select(1)
		Update(1)
						Small Table

District	Select(1)
	Update(1)
		Select(1)
		Update(1)
					Select(1)	**Small Table**

Customer	Select(1)	Select(1)
		Update(1)
			Select(2,2)	Select(10)
				Update(10)
						Skewed Update

Stock	Select(10)
	Update(10)
					Select(200)	**Skewed Update**

Item	Select(10)					**Skewed RD-Only**

Order	Insert(1)		Select(1)	Select(10)
				Update(10)
						Growing

New-order	Insert(1)			Select(10)
				Delete(10)
						Cyclic Reuse

Order-line	Insert(10)		Select(10)	Select(100)
				Update(100)
					Select(200)	**Growing**

History		Insert(1)				**Growing**

OLTP vs. OLAP

- **OLTP (On-Line Transaction Processing):**
  - Handle a transactional system with operational data using a lot of short transactions (i.e., based on SELECT, INSERT, UPDATE, DELETE)
  - Mixed read/write workloads
  - Examples: ATM machines, online banking/booking/shopping, etc.

- **OLAP (On-Line Analytical Processing):**
  - Handle an analytical system with historical data using complex queries (i.e., based on SELECT)
  - Heavy read workloads (for large data)
  - Examples: Sales analysis, market research, forecasting, etc.

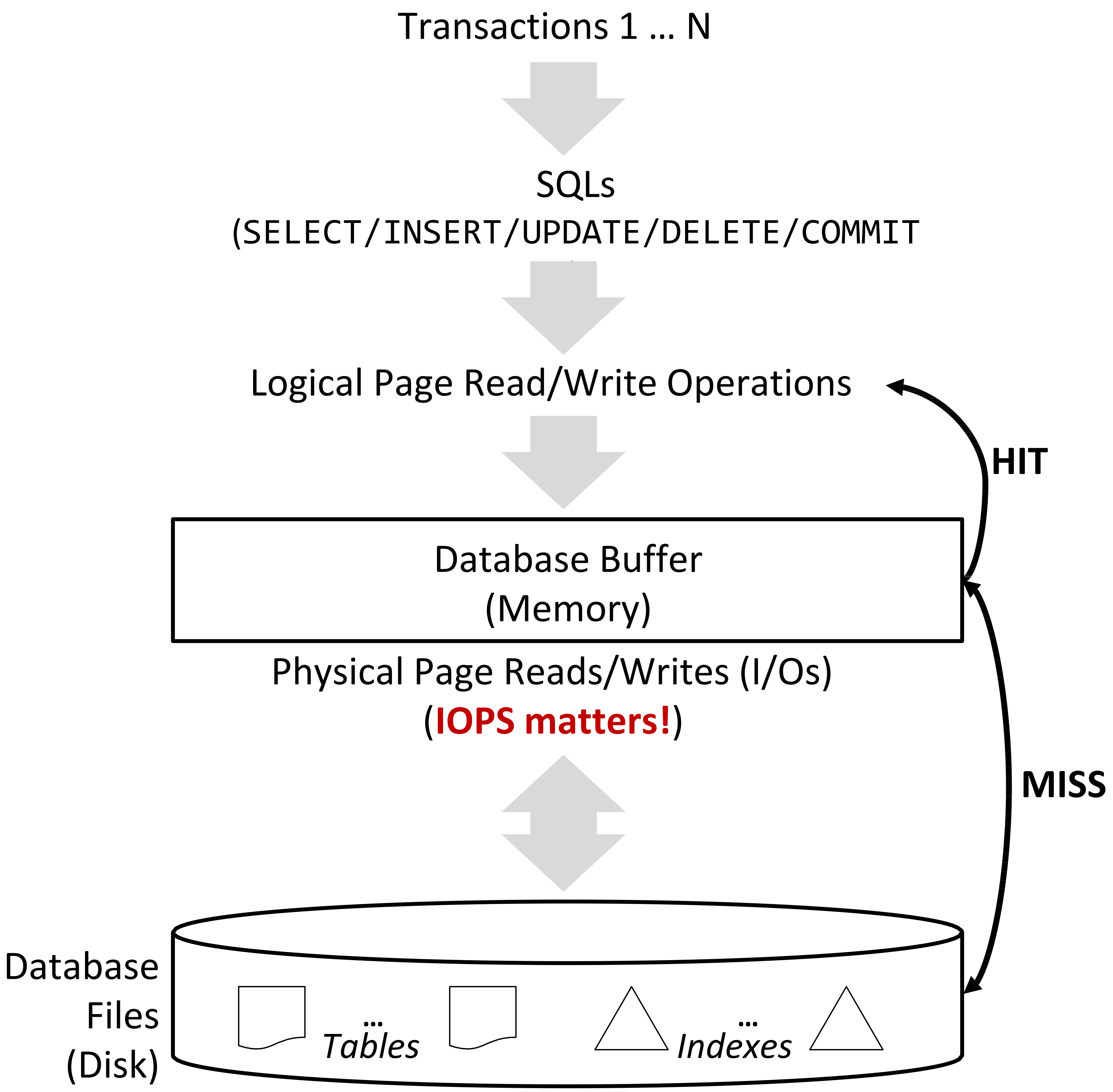
OLTP vs. OLAP: Access Patterns

- Both OLTP and OLAP uses *indexes*, but they access and handle data differently

- OLTP is a **random read/write** workload
  - It consists of random accesses with typically 2KB~16KB request sizes
  - It uses *index scans*

- OLAP is a **heavy-read** workload
  - Typically, it consists of sequential accesses to a large amount of data
  - It mainly uses *full table scans*

Database I/O Architecture



Transactions 1 … N

SQLs
(SELECT/INSERT/UPDATE/DELETE/COMMIT

Logical Page Read/Write Operations

HIT

Database Buffer
(Memory)

Physical Page Reads/Writes (I/Os)
(**IOPS matters!**)

MISS

Database
Files
(Disk)

… *Tables*

… *Indexes*

- A transaction
  = A sequence of SQL statements
  = A sequence of Reads and Writes

- SELECT reads tuples from page(s)
- INSERT/DELETE/UPDATE change records in page(s)
  - Thus, they access one or more pages

- When page(s) is in buffer (i.e., **HIT**): DRAM operation
- Otherwise (i.e., **MISS**): Disk I/Os
  - In case of dirty victim, write the page to storage
  - Read page(s) from storage

Run the TPC-C Benchmark and Analyze the Results

- This week, you will learn to monitor the system performance while running the TPC-C benchmark on MySQL

- You will also learn what those performance metrics mean

- Refer to week 2 contents in https://github.com/LeeBohyun/SWE3033-S20223

Reference

1 MySQL, “MySQL Community Downloads”, https://dev.mysql.com/downloads/mysql/

2 TPC, “TPC BENCHMARK C”, http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-c_v5.11.0.pdf

3 Veronica Lagrange, Changho Choi, Vijay Balakrishnan, “Accelerating OLTP performance with NVMe SSDs”, SDC 2016, https://www.snia.org/sites/default/files/SDC/2016/presentations/solid_state_storage/VeronicaLaGrange_Accelerating_OLTP_Performance_V6.pdf

4 Percona-Lab, “tpcc-mysql”, Github repository, https://github.com/Percona-Lab/tpcc-mysql

5 Scott T. Leutenegger and Daniel Dias, “A modeling study of the TPC-C benchmark”, SIGMOD Rec. 22, 2 (June 1, 1993), 22–31

6 Most of the slides are made by Mijin An(meeeejin@gmail.com)