

Data Management, Warehousing, And Analytics

Lab 3

Fall 2023

TRANSACTION AND NORMALIZATION

Teaching Assistants / Markers

Arpitkumar Patel (arpitkumar.patel@dal.ca)

Ruchika.(rc315087@dal.ca)

Zainuddin Saiyed (zainuddin.s@dal.ca)

Dhruv Patel (dhruvpatel@dal.ca)

Some content in this presentation is referred from Summer 2023 (with permission)

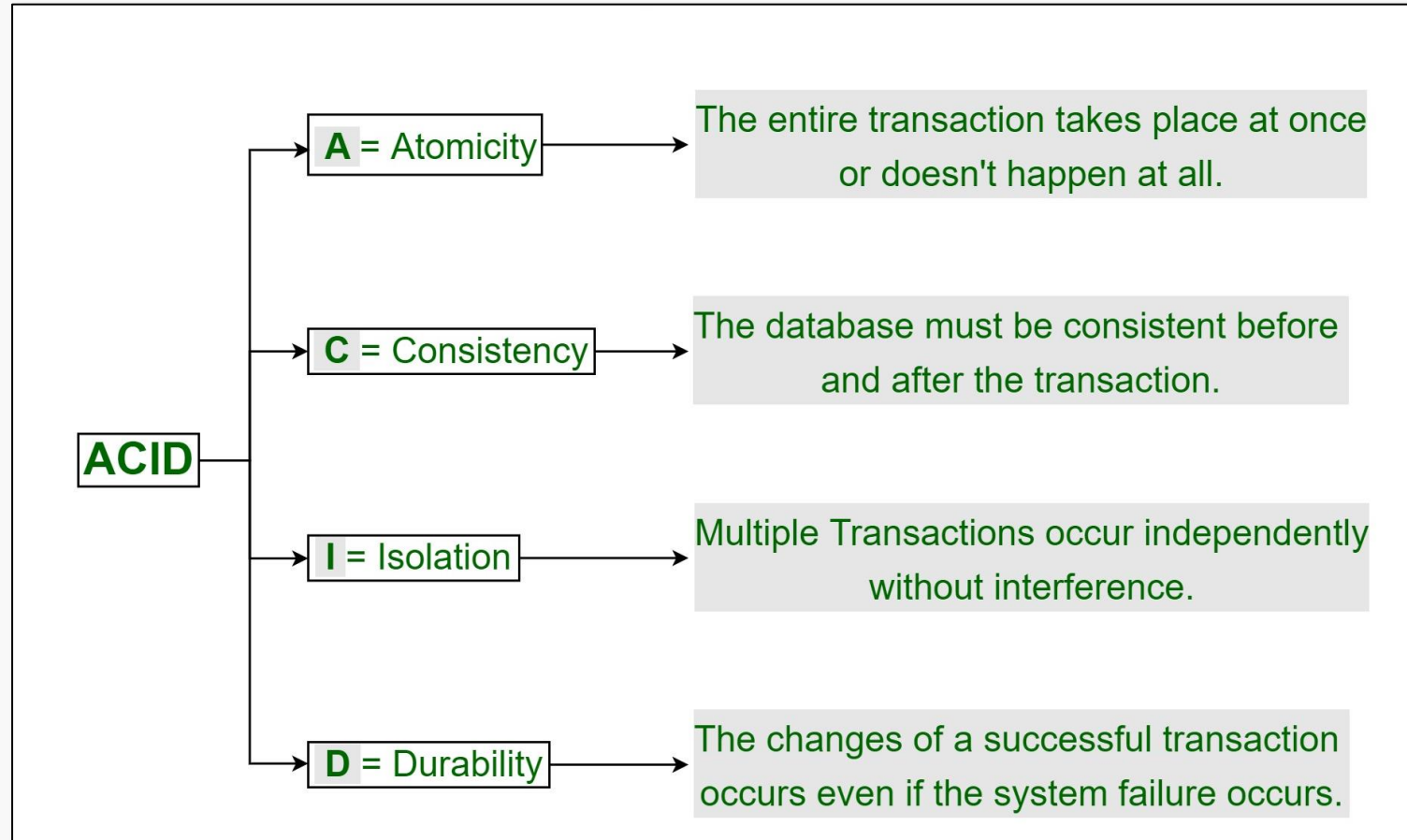
Agenda

- What are Transactions?
- Several States in Transactions
- Types of Transactions
- Normalization
- Hands On

Transactions

- A Database transaction is a feature in the database which helps to perform one or more operations maintaining the consistency of the database.
- All the statements between the beginning and ending of a transaction can be considered as a single unit.
- While performing a Transaction, the database will be in an inconsistent state.
- Only if the transaction is committed (COMMIT;) the database's state is changed from one consistent state to other.
- Transactions that do not modify data in the tables but only fetch the required data are known as read-only transactions.
- ACID properties.

ACID properties in DBMS



Transaction Syntax

- To inform the system that a transaction is being started, we use `START TRANSACTION` command.
- We can perform any operation which we need inside a transaction `UPDATE`, `INSERT`, `SELECT`, etc.
- At the end of the transaction, we can either `COMMIT` the transaction or `ROLLBACK` the transaction.

Transaction Syntax continued...

SET autocommit = 0;
before the start of the transaction.

```
START TRANSACTION;
```

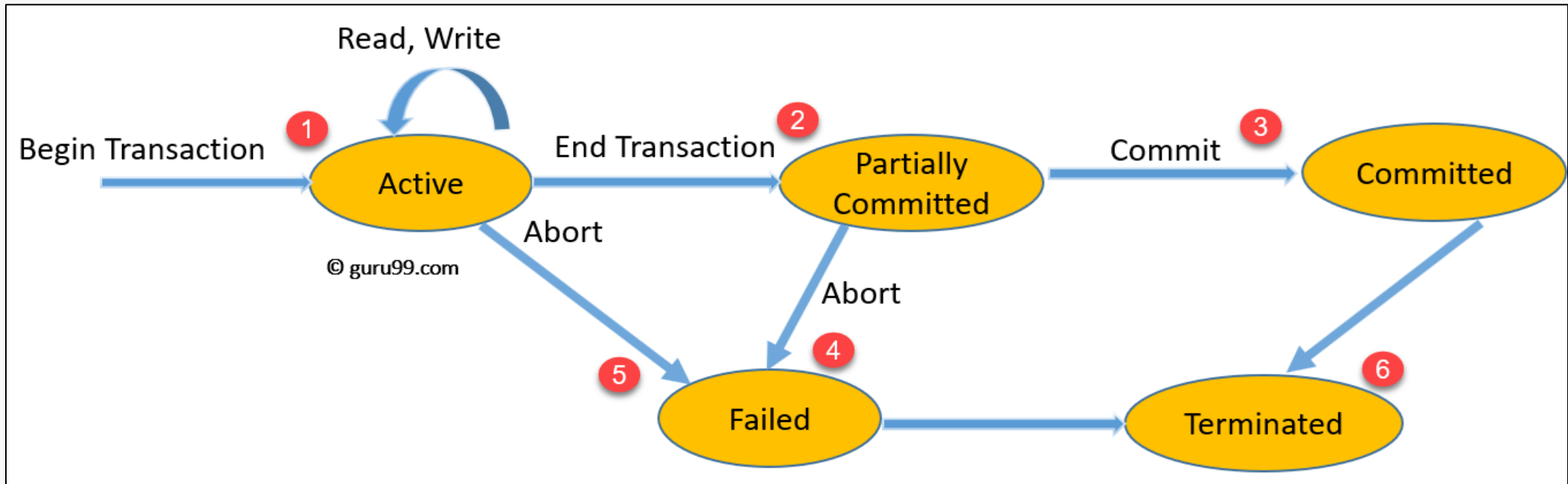
```
INSERT INTO Student VALUES(1, "XYZ", "XYZ@dal.ca");
```

```
INSERT INTO Student VALUES(2, "ABC", "ABC@dal.ca");
```

```
INSERT INTO Student VALUES(3, "PQR", "PQR@dal.ca");
```

```
COMMIT; ROLLBACK;
```

States in Transaction



Save point

- SAVEPOINT is like a flag which breaks down a task into several tasks.
- SAVEPOINT helps to rollback a part of the transaction.
- SAVEPOINTS are exactly like the checkpoints in the video games. We create a checkpoint when we have completed a sure or difficult task.
- The SAVEPOINTS are released when COMMIT or ROLLBACK. We use RELEASE SAVEPOINT to manually release a SAVEPOINT.

Normalization

- Process of organizing the database.
- Main objective of normalization is to:
 1. *Reduce the redundancy,*
 2. *Simplify the queries and reduce the cost,*
 3. *Reduce/Avoid data modification issues.*

Normalization Forms

Normal Form	Description
First Normal Form (1NF)	Should be in a proper format, no group values for a field and a primary key is identified.
Second Normal Form (2NF)	Should be in 1 st Normal Form and there should be no partial dependency.
Third Normal Form (3NF)	Should be in 2 nd Normal Form and there should be no transitive dependency.

1 NF Example

Employee ID	Employee Name	Phone Number	Salary
1EDU001	Alex	+91 8553206126 +91 9449424949	60,131
1EDU002	Barry	+91 8762989672	48,302
1EDU003	Clair	+91 9916255225	22,900
1EDU004	David	+91 6363625811 +91 8762055007	

Employee ID	Employee Name	Phone Number	Salary
1EDU001	Alex	+91 8553206126	60,131
1EDU001	Alex	+91 9449424949	60,131
1EDU002	Barry	+91 8762989672	48,302
1EDU003	Clair	+91 9916255225	22,900
1EDU004	David	+91 6363625811	81,538
1EDU004	David	+91 8762055007	81,538

2 NF Example

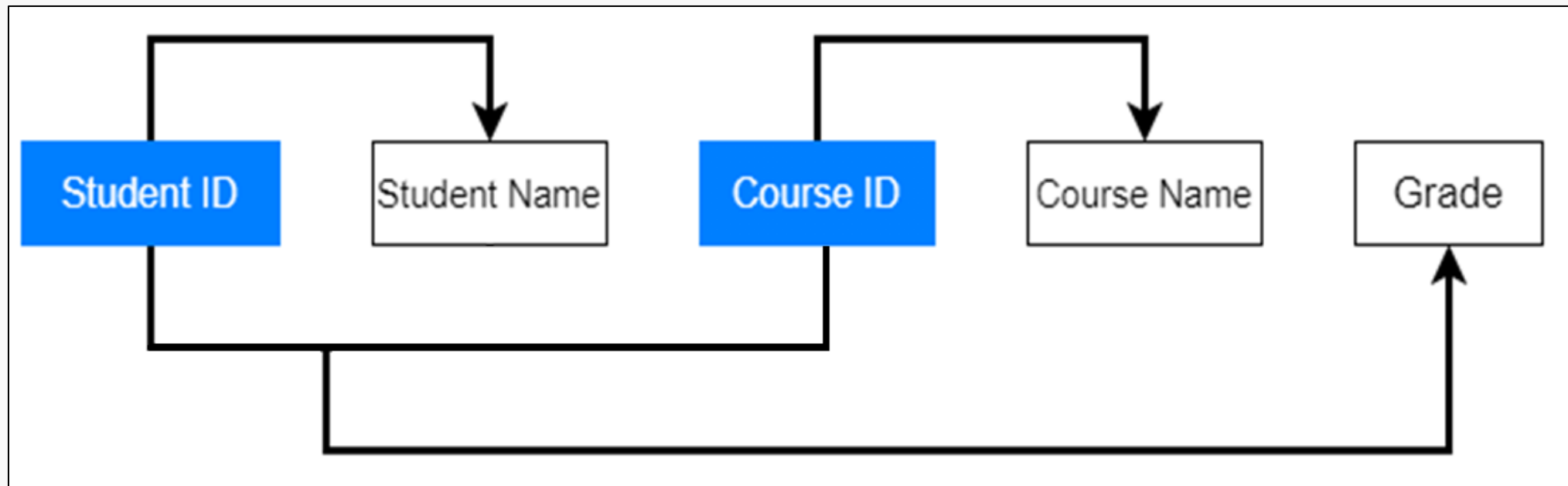
Employee Id	Department Id	Office Location
1EDU001	ED-T1	Pune
1EDU002	ED-S2	Bengaluru
1EDU003	ED-M1	Delhi
1EDU004	ED-T3	Mumbai

Employee Id	Department Id
1EDU001	ED-T1
1EDU002	ED-S2
1EDU003	ED-M1
1EDU004	ED-T3

Department Id	Office Location
ED-T1	Pune
ED-S2	Bengaluru
ED-M1	Delhi
ED-T3	Mumbai

Partial Dependency

- When we have a composite primary key and some attributes dependent only on a part of the primary key then it is a partial dependency.



3 NF Example

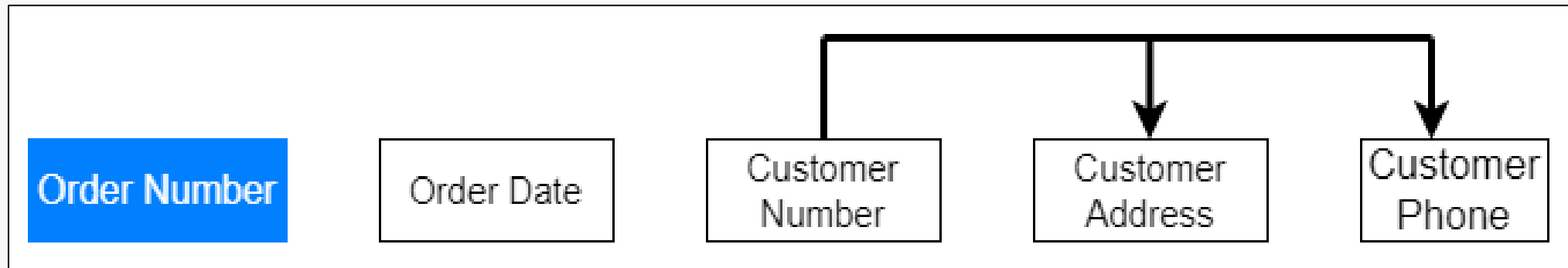
Student Id	Student Name	Subject Id	Subject	Address
1DT15ENG01	Alex	15CS11	SQL	Goa
1DT15ENG02	Barry	15CS13	JAVA	Bengaluru
1DT15ENG03	Clair	15CS12	C++	Delhi
1DT15ENG04	David	15CS13	JAVA	Kochi

Student Id	Student Name	Subject Id	Address
1DT15ENG01	Alex	15CS11	Goa
1DT15ENG02	Barry	15CS13	Bengaluru
1DT15ENG03	Clair	15CS12	Delhi
1DT15ENG04	David	15CS13	Kochi

Subject Id	Subject
15CS11	SQL
15CS13	JAVA
15CS12	C++
15CS13	JAVA

Transitive Dependency

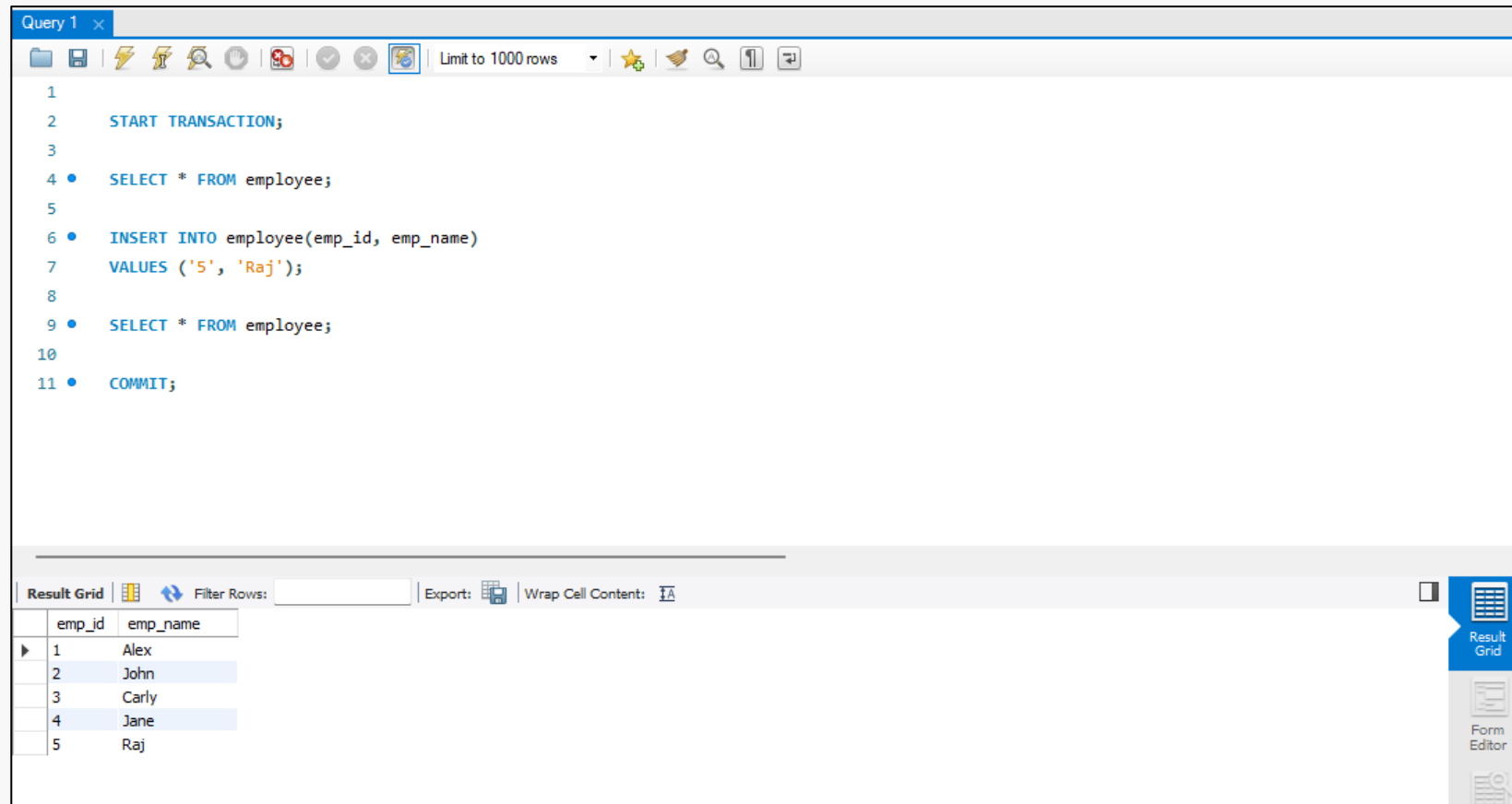
- When an attribute is independent of the primary key but dependent on some other attribute then it is transitive dependency.



Hands on

- In your MySQL workbench query page, you will first need to execute the following statement:
 - **SET** autocommit = 0;
- This is to ensure that the transaction is not committed until you explicitly mention the commit statement.

Hands on ...continued...



The screenshot shows a SQL query editor window titled "Query 1". The query is as follows:

```
1
2  START TRANSACTION;
3
4  • SELECT * FROM employee;
5
6  • INSERT INTO employee(emp_id, emp_name)
7  VALUES ('5', 'Raj');
8
9  • SELECT * FROM employee;
10
11 • COMMIT;
```

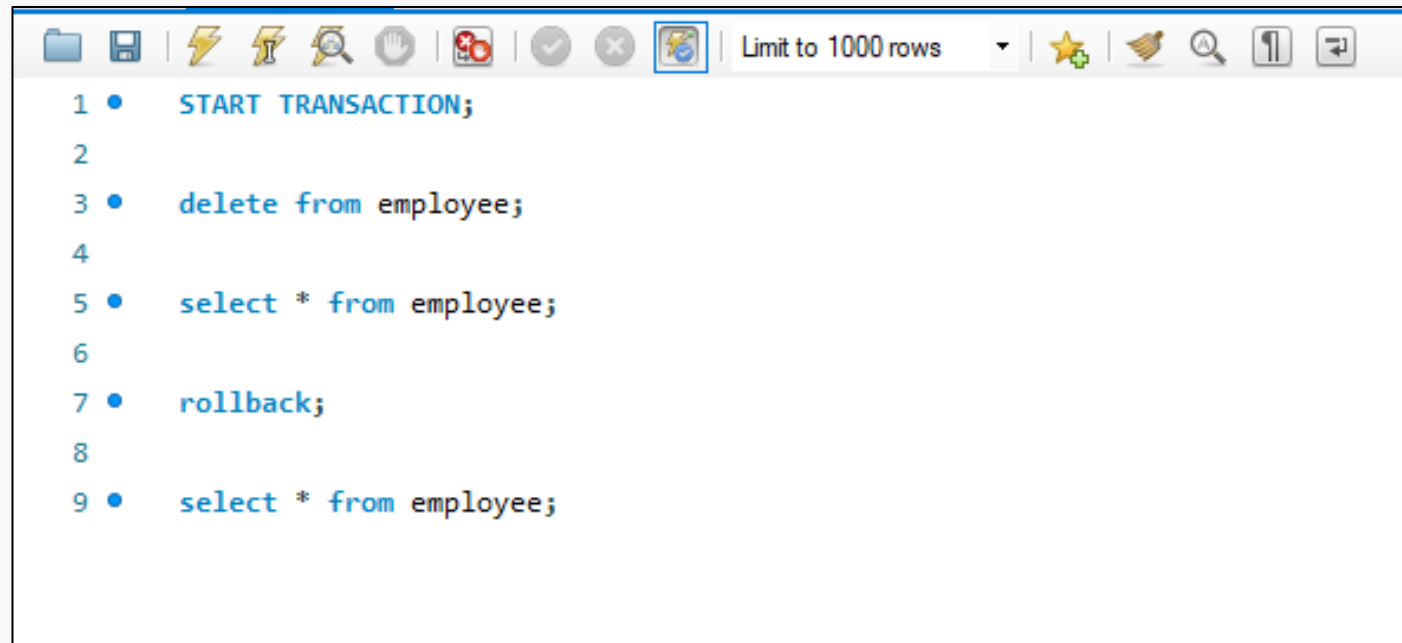
The query is executed, and the results are displayed in a table below the query editor. The table has two columns: "emp_id" and "emp_name". The results are:

	emp_id	emp_name
1	Alex	
2	John	
3	Carly	
4	Jane	
5	Raj	

The interface includes a toolbar at the top with various icons for file operations, execution, and formatting. A "Limit to 1000 rows" dropdown is visible. On the right side, there are buttons for "Result Grid", "Form Editor", and "Export".

Screenshot of example transaction and output.

Hands on with rollback

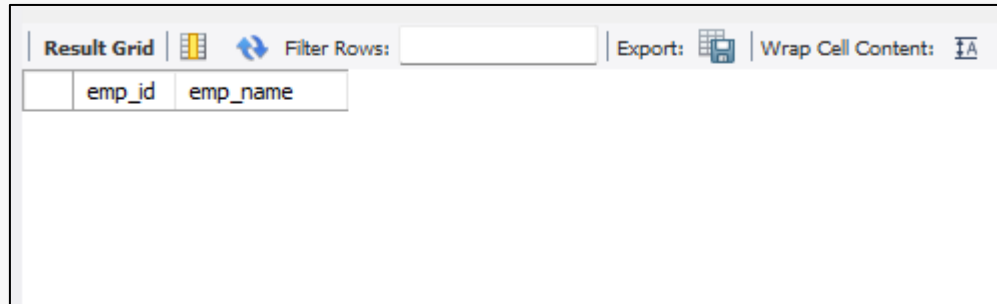


The screenshot shows a SQL IDE window with a toolbar at the top. The toolbar includes icons for file operations (folder, save), execution (lightning bolt, play), search (magnifying glass), and other database functions. A dropdown menu shows 'Limit to 1000 rows'. The main text area contains a SQL script with line numbers 1 through 9. The script starts with 'START TRANSACTION;', followed by a blank line, then 'delete from employee;', another blank line, 'select * from employee;', a blank line, 'rollback;', a blank line, and finally 'select * from employee;'.

```
1 • START TRANSACTION;  
2  
3 • delete from employee;  
4  
5 • select * from employee;  
6  
7 • rollback;  
8  
9 • select * from employee;
```

Transaction with rollback. Two outputs provided in the following slide.


Rollback continued



The screenshot shows a database interface with a toolbar at the top containing 'Result Grid', a grid icon, a refresh icon, a 'Filter Rows:' input field, an 'Export:' button with a grid icon, and a 'Wrap Cell Content:' button with a text icon. Below the toolbar is a table with two columns: 'emp_id' and 'emp_name'. The table is currently empty.

emp_id	emp_name
--------	----------

Output 1: Pre rollback post delete



The screenshot shows the same database interface as the first one. The table now contains five rows of data. The first row is highlighted with a blue background. The columns are 'emp_id' and 'emp_name'.

emp_id	emp_name
1	Alex
2	John
3	Carly
4	Jane
5	Raj

Output 2: Post rollback

Rollback continued

- You can see in the aforementioned example that the table from which all records were dropped, after rollback, the impact of that query on the database was erased. This is what rollback does.

Hands on with SAVEPOINT

- A SAVEPOINT in MySQL creates a checkpoint within the transaction.
- A SAVEPOINT can be either RELEASE (released) or ROLLBACK (rolled back)
- In case of RELEASE, the SAVEPOINT is destroyed without undoing the effects of the queries executed after the SAVEPOINT was created. You cannot ROLLBACK a SAVEPOINT after RELEASE.
- In case of ROLLBACK, all queries after the SAVEPOINT are rolled back without destroying the rest of the transaction

SAVEPOINT continued

```
4
5 • INSERT INTO employee(emp_id, emp_name)
6   VALUES (6, 'Rita');
7
8 • SAVEPOINT my_savepoint;
9
10 • INSERT INTO employee(emp_id, emp_name)
11   VALUES (7, 'Jeremy');
12
13 • ROLLBACK TO SAVEPOINT my_savepoint;
14
15 • INSERT INTO employee(emp_id, emp_name)
16   VALUES (8, 'Jonathan');
17
18 • SELECT * FROM employee;
19
20 • COMMIT;
```

Transaction with rolled back savepoint

	emp_id	emp_name
▶	1	Alex
	2	John
	3	Carly
	4	Jane
	5	Raj
	6	Rita
	8	Jonathan

Output of transaction

SAVEPOINT continued

```
1 • START TRANSACTION;
2 • INSERT INTO employee(emp_id, emp_name)
3   VALUES (9, 'Jimmy');
4
5 • SAVEPOINT my_savepoint;
6
7 • update employee set emp_name = 'Saville' where emp_id=9;
8
9 • RELEASE SAVEPOINT my_savepoint;
10
11 • INSERT INTO employee(emp_id, emp_name)
12   VALUES (11, 'Jake');
13
14 • SELECT * FROM employee;
15
16 • COMMIT;
```

Transaction with released savepoint

	emp_id	emp_name
▶	1	Alex
	2	John
	3	Carly
	4	Jane
	5	Raj
	6	Rita
	8	Jonathan
	9	Saville
	11	Jake

Output

Lab Assignment

- Task: Design a banking application database for managing customer's debit and credit.
- 1. Design a banking application database with the **customer's details** (minimal attributes - name, mailing address, permanent address, primary email, primary phone number), **account details** (minimal attributes - account number, account balance) and **account transfer details** (minimal attributes - account number, date of transfer, recipient name, status(varchar value waiting/accepted/declined)).
- 2. Create a transaction environment where on every account debit, the record in the **account details table** is updated but not committed. Upon this update, insert a record in the **account transfer details table** with waiting state.
- 3. Assume that the transaction status gets verified by some X business logic (pure assumption here, no need to implement any logic), the transaction status is changed to either accepted or declined. Based on this status, handle the update balance query.
(Hint: Based on the transaction status, either commit or rollback the **account details table SQL statements**. Try exploring SAVEPOINT).
- * **If you want**, you can create separate scenarios for Q2 and Q3 (Not necessary). Put your assumptions in the report.

Deadline: Before Saturday 11:59 pm. Please refer "Submission Guideline" in the Brightspace.

Q&A



References

- “Flowchart Maker & Online Diagram Software,” *Draw.io* [Online]. Available at: <https://app.diagrams.net/> [Accessed 24 September 2023].
- “DBMS Transaction Management: ACID Properties, Schedule,” *Guru99* [Online]. Available at: <https://www.guru99.com/dbms-transaction-management.html> [Accessed 24 September 2023].
- “What is Normalization? 1NF, 2NF, 3NF & BCNF with Examples,” *Guru99* [Online]. Available at: <https://www.guru99.com/database-normalization.html> [Accessed 24 September 2023].
- “DBMS - Normalization - Tutorialspoint,” *Tutorialspoint* [Online]. Available at: https://www.tutorialspoint.com/dbms/database_normalization.htm [Accessed 24 September 2023].
- “ACID Properties in DBMS - GeeksforGeeks,” *GeeksforGeeks* [Online]. Available at: <https://www.geeksforgeeks.org/acid-properties-in-dbms/> [Accessed 24 September 2023].
- “Everyday Transactions | Relating to Modern Databases | InformIT,” *Informit* [Online]. Available at: <https://www.informit.com/articles/article.aspx?p=101190&seqNum=2> [Accessed 24 September 2023].
- “Normalization in SQL | 1NF, 2NF, 3NF and BCNF in Database,” *Edureka* [Online]. <https://www.edureka.co/blog/normalization-in-sql/> [Accessed 24 September 2023].