Schools of: informatics

Department of: computer sciences

Course title : advanced database

Project title: university cafeteria management system

Group 4

Name	Id
Dagim Dawit	ugr/92584/16
Mahlet Amsalu	ugr/91944/16
Deginesh Dalga	ugr/91619/16
Ashenafi Degif	ugr/91463/16

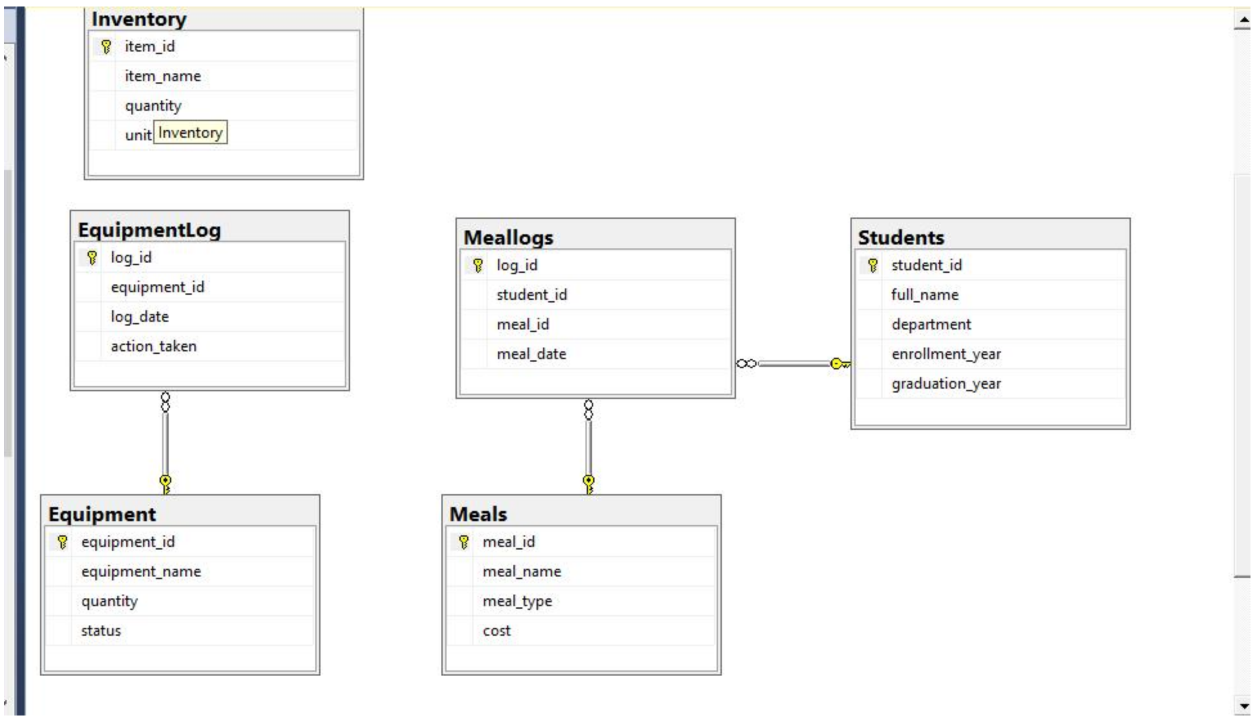
Project Overview:

This project aims to design and implement a relational database system to manage the operations of a university cafeteria. The system handles student meal logging, inventory tracking, equipment usage, and billing, while addressing data consistency and concurrency control issues using SQL Server features.

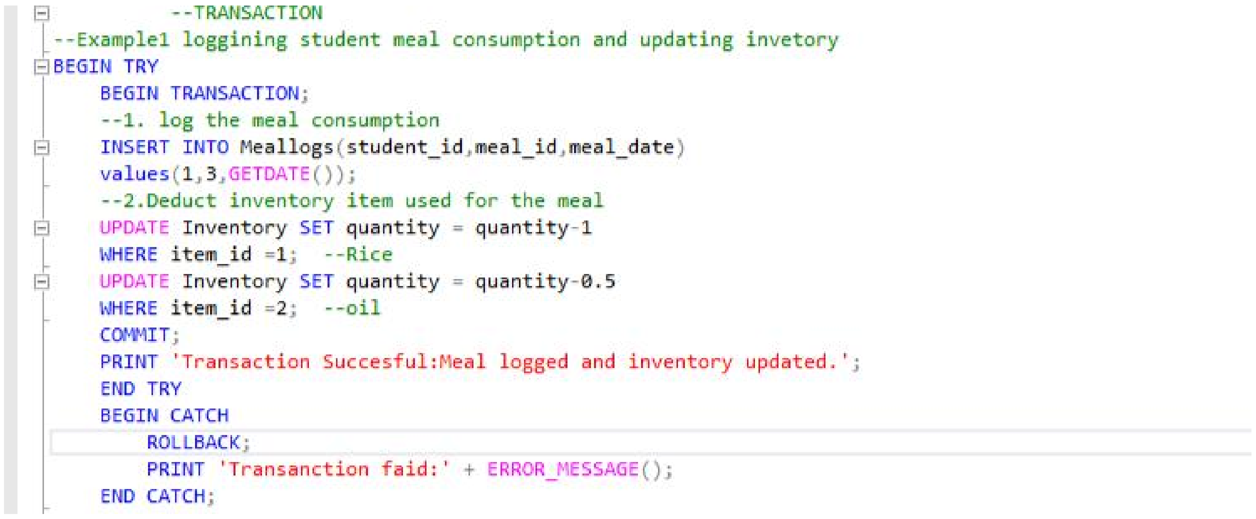
Objectives:

1. Manage student meal records efficiently.
2. Track inventory and food consumption.
3. Monitor kitchen equipment usage and maintenance.
4. Ensure accurate billing after graduation.
5. Implement safe concurrent access to avoid lost updates, dirty reads

1. design relational database application based on choose project



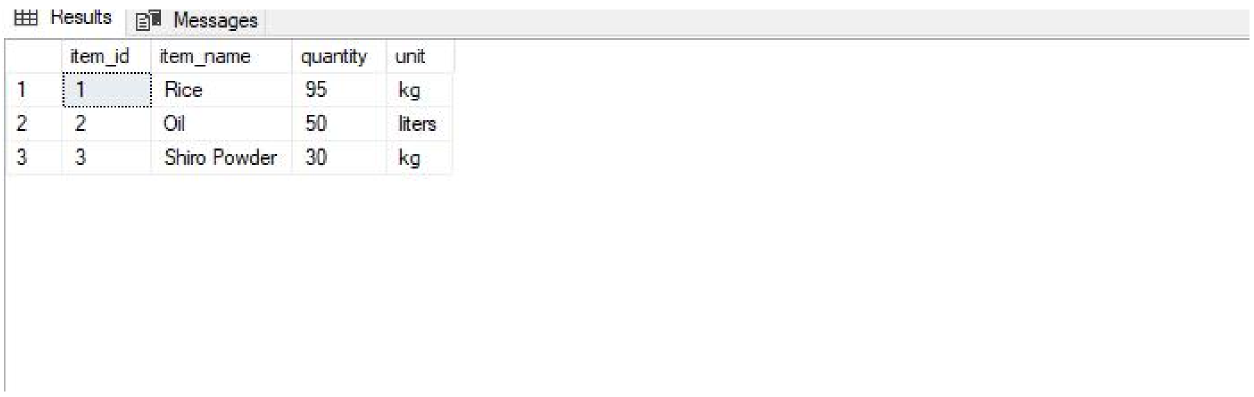
2.Implement transaction management for critical operations.



```
            --TRANSACTION
--Example1 loggining student meal consumption and updating invetory
BEGIN TRY
    BEGIN TRANSACTION;
    --1. log the meal consumption
    INSERT INTO Meallogs(student_id,meal_id,meal_date)
    values(1,3,GETDATE());
    --2.Deduct inventory item used for the meal
    UPDATE Inventory SET quantity = quantity-1
    WHERE item_id =1;  --Rice
    UPDATE Inventory SET quantity = quantity-0.5
    WHERE item_id =2;  --oil
    COMMIT;
    PRINT 'Transaction Succesful:Meal logged and inventory updated.';
    END TRY
    BEGIN CATCH
        ROLLBACK;
        PRINT 'Transanction faid:' + ERROR_MESSAGE();
    END CATCH;
```

3. demonstrate the use of commit,rollback and save point.

Values before transaction:



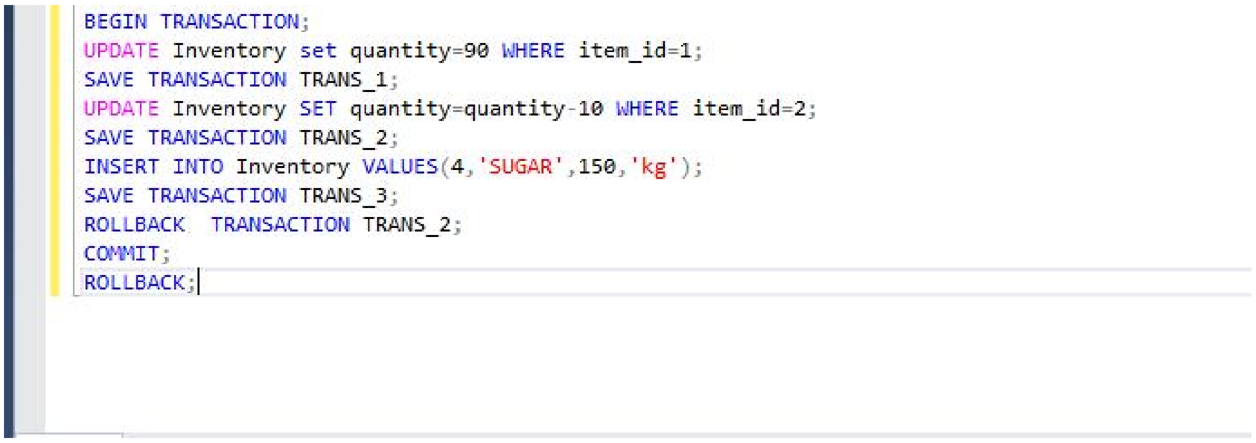
Results | Messages

	item_id	item_name	quantity	unit
1	1	Rice	95	kg
2	2	Oil	50	liters
3	3	Shiro Powder	30	kg

Save point: used with in transaction to create a named point that can be rolled back to if needed.

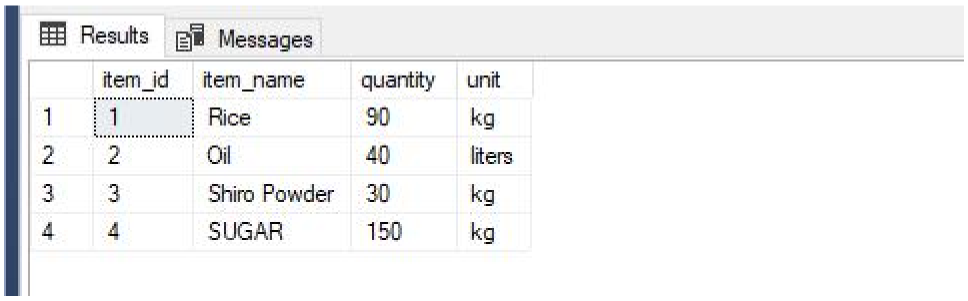
Rollback: used to undo transaction by using savepoint or rollback itself.

Commit: used to submit transaction into database, and once committed transaction cannot be rollback.



```
BEGIN TRANSACTION;
UPDATE Inventory set quantity=90 WHERE item_id=1;
SAVE TRANSACTION TRANS_1;
UPDATE Inventory SET quantity=quantity-10 WHERE item_id=2;
SAVE TRANSACTION TRANS_2;
INSERT INTO Inventory VALUES(4,'SUGAR',150,'kg');
SAVE TRANSACTION TRANS_3;
ROLLBACK  TRANSACTION TRANS_2;
COMMIT;
ROLLBACK;
```

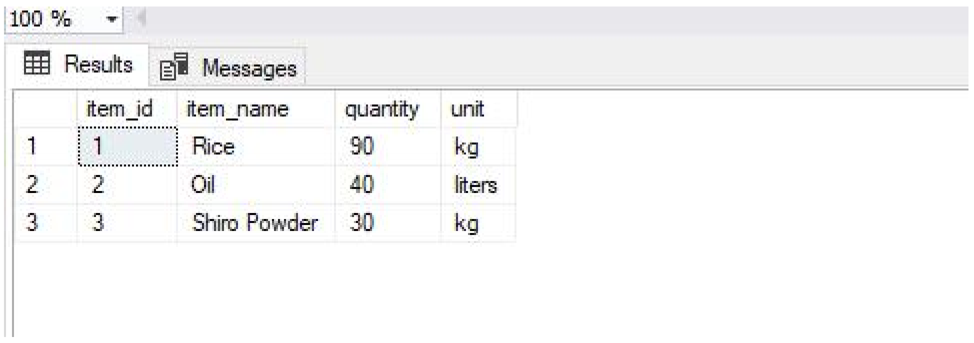
Results before commit and rollback:



Results | Messages

	item_id	item_name	quantity	unit
1	1	Rice	90	kg
2	2	Oil	40	liters
3	3	Shiro Powder	30	kg
4	4	SUGAR	150	kg

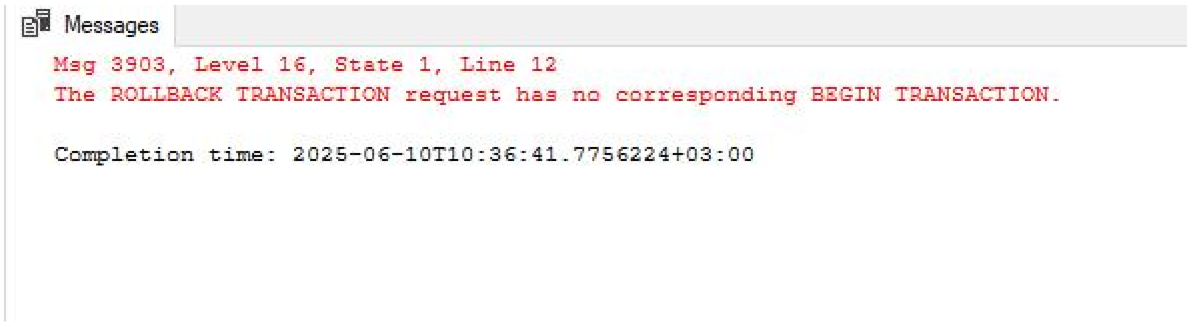
Result after rollback transaction into savepoint trans_2:


100 %

Results | Messages

	item_id	item_name	quantity	unit
1	1	Rice	90	kg
2	2	Oil	40	liters
3	3	Shiro Powder	30	kg

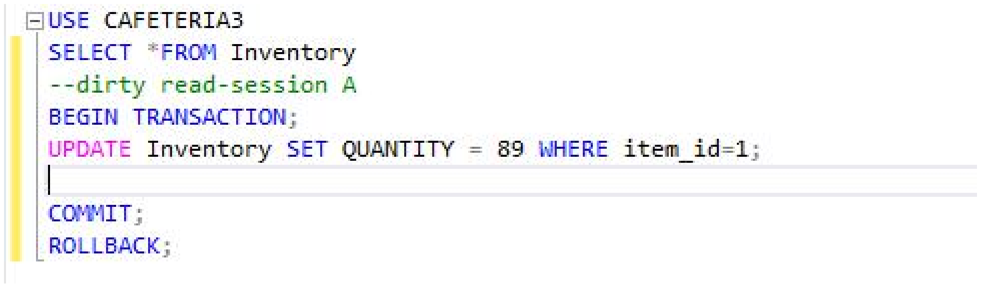
Result After committed, when we use rollback:

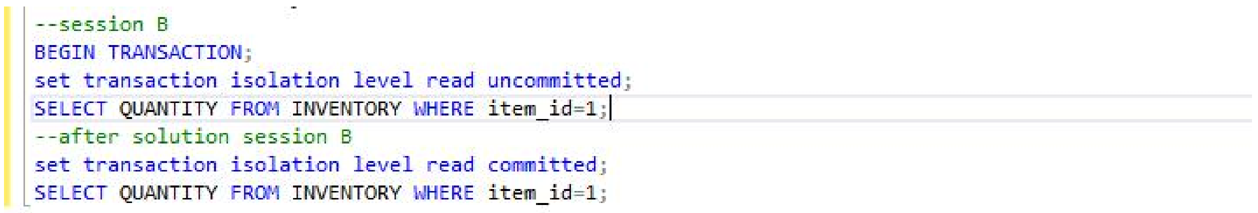

Messages

```
Msg 3903, Level 16, State 1, Line 12
The ROLLBACK TRANSACTION request has no corresponding BEGIN TRANSACTION.

Completion time: 2025-06-10T10:36:41.7756224+03:00
```

4. simulate concurrent transaction and analyze the impact of concurrency control techniques.

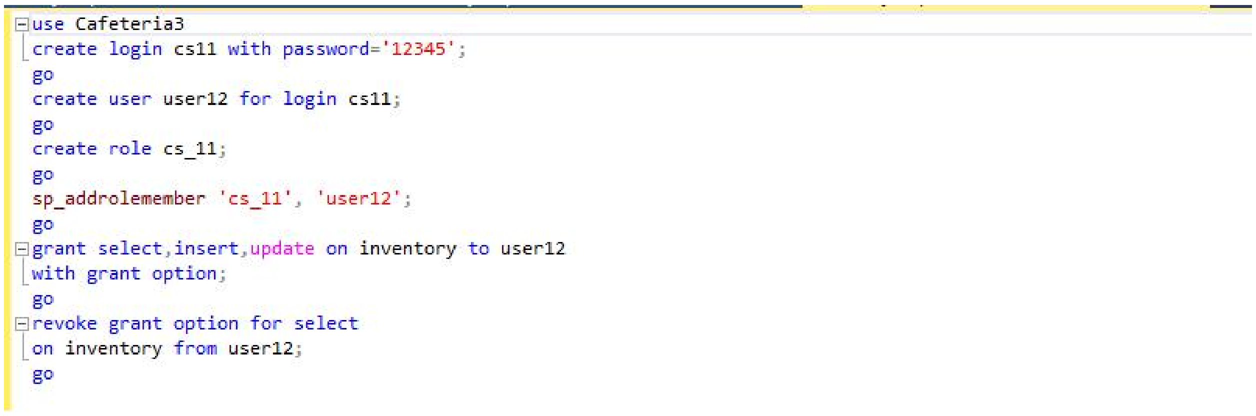
I. Dirty read: If session A makes changes but not commit yet, then session B reads uncommitted value , if session A later rollback, session B has dirty read.


```
USE CAFETERIA3
SELECT *FROM Inventory
--dirty read-session A
BEGIN TRANSACTION;
UPDATE Inventory SET QUANTITY = 89 WHERE item_id=1;

COMMIT;
ROLLBACK;
```


```
--session B
BEGIN TRANSACTION;
set transaction isolation level read uncommitted;
SELECT QUANTITY FROM INVENTORY WHERE item_id=1;
--after solution session B
set transaction isolation level read committed;
SELECT QUANTITY FROM INVENTORY WHERE item_id=1;
```

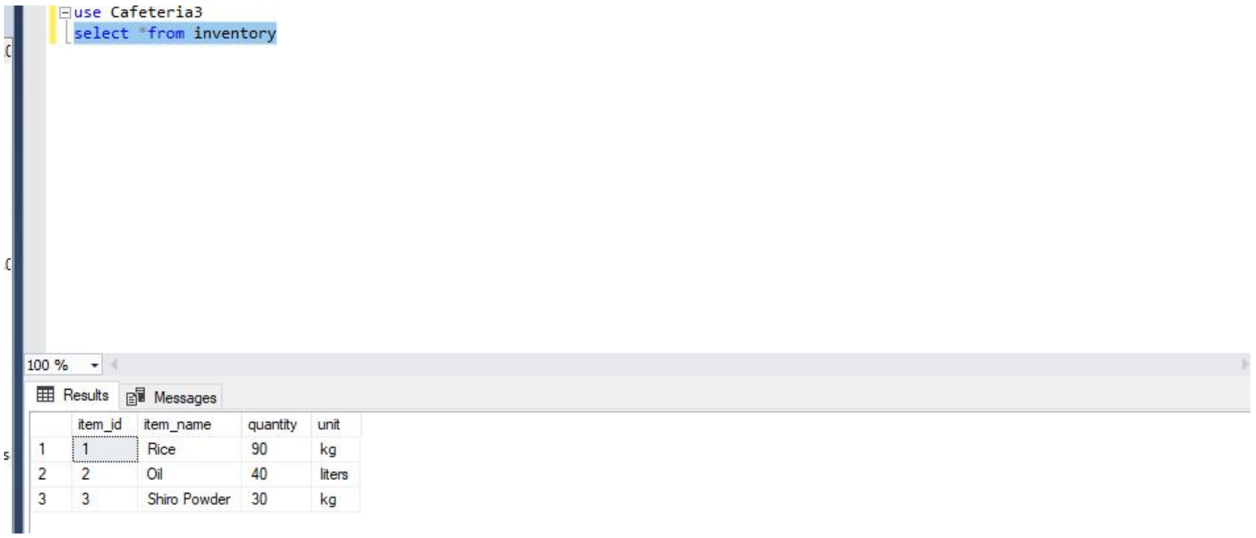
After solution when session A transaction is not committed, session B transaction waits for Session A to commit or rollback transaction.

5. Implement discretionary access control by granting and revoking privileges to user.



```
use Cafeteria3
create login cs11 with password='12345';
go
create user user12 for login cs11;
go
create role cs_11;
go
sp_addrolemember 'cs_11', 'user12';
go
grant select,insert,update on inventory to user12
with grant option;
go
revoke grant option for select
on inventory from user12;
go
```

After granting, user cs11 can access database, update, and select inventory table.



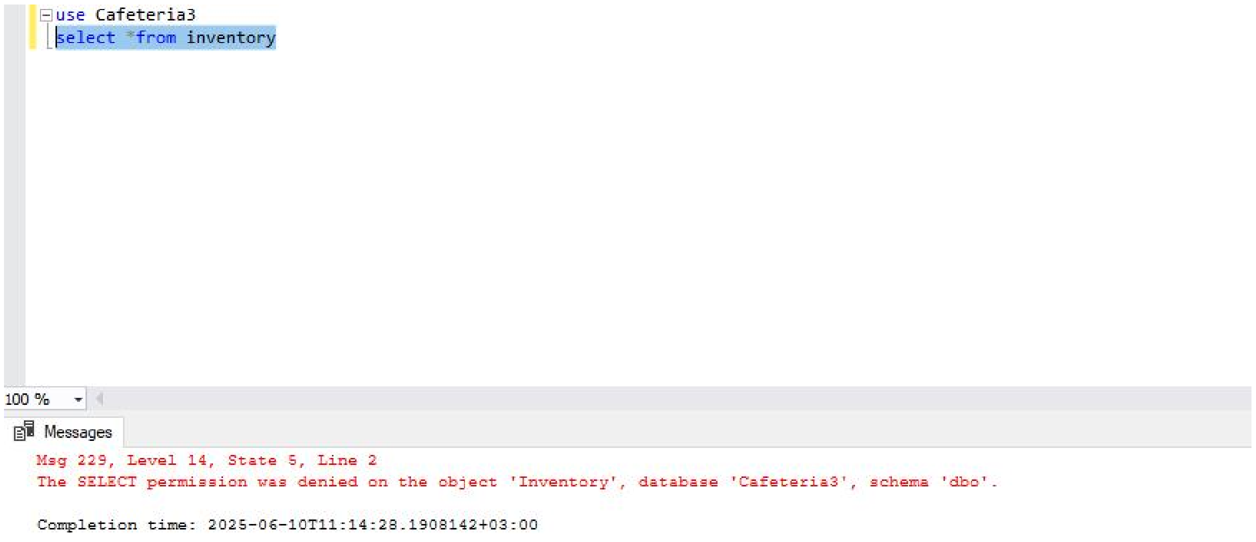
```
use Cafeteria3
select *from inventory
```

100 %

Results | Messages

	item_id	item_name	quantity	unit
1	1	Rice	90	kg
2	2	Oil	40	liters
3	3	Shiro Powder	30	kg

After revoking, user cs11 denied to select inventory from database.



```
use Cafeteria3
select *from inventory
```

```
Msg 229, Level 14, State 5, Line 2
The SELECT permission was denied on the object 'Inventory', database 'Cafeteria3', schema 'dbo'.

Completion time: 2025-06-10T11:14:28.1908142+03:00
```

Conclusion :

In our University Cafe Management System, we implemented transaction management and concurrency control strategies to ensure data integrity, consistency, and reliability. Transactions are carefully managed using BEGIN TRANSACTION, COMMIT, and ROLLBACK, with support for SAVEPOINT where necessary.

To handle concurrency:

We used appropriate isolation levels (READ COMMITTED, SERIALIZABLE) to prevent dirty reads and ensure serializability.