

CSE 311

Transactions



Outline

- Transaction Definition
- Transaction Boundaries
- Committing a transaction
- Rolling Back transaction
- Transaction Control
 - Savepoint
- An Example



Transactions

- Transaction is **a series of one or more SQL statements** that are logically related or **a series of operations** performed on Oracle table data.
- Transactions are a means to break programming code into manageable units.
- A successfully executed SQL statement and a committed transaction are not same. Even if an SQL statement is executed successfully, unless the transaction containing the statement is committed, it can be rolled back and all changes made by the statement(s) can be undone.



A

5000

B

4000

Transfer 2000 from A to B

After Successful transfer

A

3000

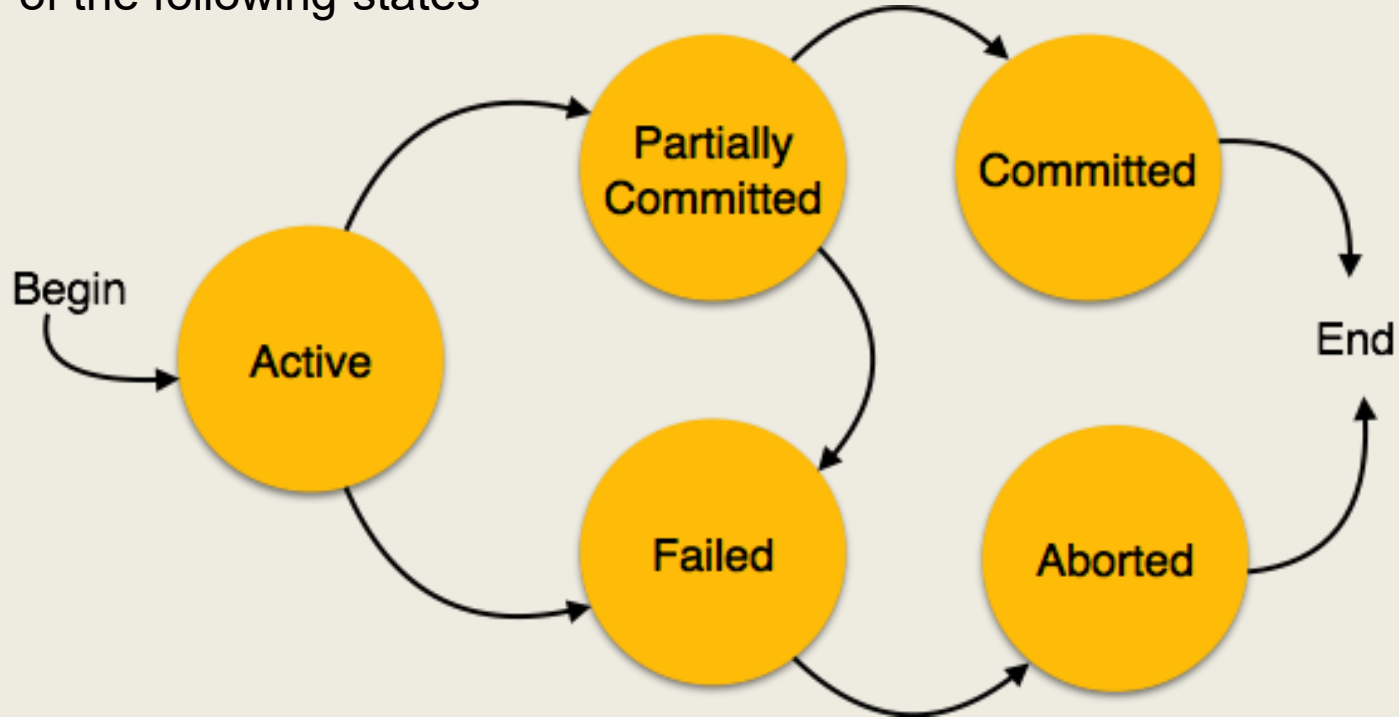
B

6000



Transaction States

A transaction in a database can be in one of the following states –





Transaction States

- **Active** – In this state, the transaction is being executed. This is the initial state of every transaction.
- **Partially Committed** – When a transaction executes its final operation, it is said to be in a partially committed state.
- **Failed** – A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.
- **Aborted** – If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts –
 - Re-start the transaction
 - Kill the transaction
- **Committed** – If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.



Transaction Boundaries

- A transaction has a beginning and an end.
- A **transaction begins** when one of the following events take place:
 - When the **first SQL statement is executed** after connecting to the database.
 - At **each new SQL statement issued** after a transaction is completed.



Transaction Boundaries (Cont..)

- A **transaction ends** when one of the following events take place:
 - A **COMMIT** or a **ROLLBACK** statement is issued.
 - **COMMIT** makes events within a transaction permanent.
 - **ROLLBACK** erases events within a transaction.
 - A **DDL statement**, like CREATE TABLE statement, is issued; because in that case a COMMIT is automatically performed.
 - The **system crashes**.

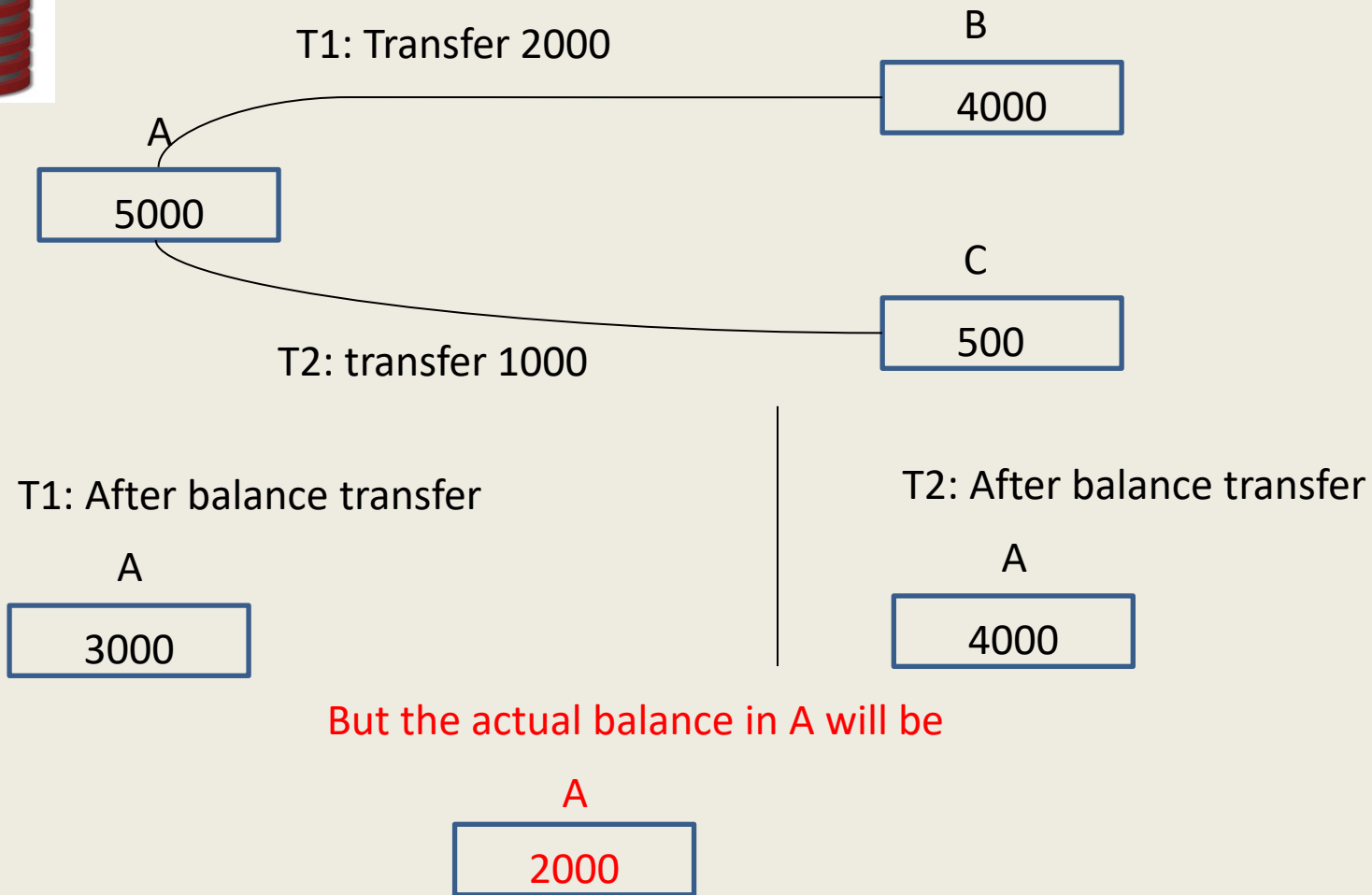


Committing a Transaction

- A **transaction is made permanent** by issuing the SQL command COMMIT.
- The general syntax for the COMMIT command is:

Commit;

- When a COMMIT statement is issued to the database, the following results are true:
 - All work done by the transaction becomes permanent.
 - Other users can see changes in data made by the transaction.
 - Any locks acquired by the transaction are released.





For T1:

A

Initial:

5000

B

4000

T2 is in
waiting state

A

After T1:

3000

B

6000

After Completing T1, t2 Begins

A

Initial:

5000

C

500

A

After T2:

2000

C

1500



Example of Commit

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3, 'kaushik', 23, 'Kota', 2000.00 );
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00 );
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00 );
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6, 'Komal', 22, 'MP', 4500.00 );
COMMIT;
```



Rolling Back Transactions

- Changes made to the database without **COMMIT** could be undone using the ROLLBACK command.
- The general syntax for the ROLLBACK command is:
`ROLLBACK [TO SAVEPOINT < savepoint_name>];`
- If you are not using savepoint, then simply use the following statement to rollback all the changes:
`ROLLBACK;`
- When a ROLLBACK statement is issued to the database, the following results are true:
 - All work done by the transaction is undone.
 - Any locks acquired by the transaction are released.



Transaction Control

- **Savepoints:**
 - Savepoints are **sort of markers** that help in splitting a long transaction into smaller units by setting some **checkpoints**.
 - By setting savepoints within a long transaction, you can **roll back to a checkpoint** if required.
 - This is done by issuing the SAVEPOINT command.
- ✓ The general syntax for the SAVEPOINT command is:
SAVEPOINT < savepoint_name >;



Example of Rollback and Savepoint

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (7, 'Rajnish', 27, 'HP', 9500.00 );
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (8, 'Riddhi', 21, 'WB', 4500.00 );
```

```
SAVEPOINT sav1;
```

```
UPDATE CUSTOMERS
SET SALARY = SALARY + 1000;
```

```
ROLLBACK TO sav1;
```

```
UPDATE CUSTOMERS
SET SALARY = SALARY + 1000
WHERE ID = 7;
UPDATE CUSTOMERS
SET SALARY = SALARY + 1000
WHERE ID = 8;
COMMIT;
```

Here, **ROLLBACK TO sav1**; statement rolls back the changes up to the point, where you had marked **savepoint sav1** and after that new changes will start.



Advantages of COMMIT and ROLLBACK

- With COMMIT and ROLLBACK statements, you can:
 - Ensure data consistency
 - Preview data changes before making changes permanent
 - Group logically related operations