

This module provides an introduction to database transactions and why they are important.

Some Transactions

- › Transaction A

```
SELECT name, location, category
FROM product WHERE category='Lawn mowers';
```
- › Transaction B

```
INSERT INTO supply (prod_id, quantity, arrival)
VALUES ('LW24453', 3000, '2015-09-26');

UPDATE stocklevel
SET quantity=quantity+3000
WHERE prod_id='LW24453';
```
- › Transaction C

```
UPDATE product set category='Electric tools'
WHERE id='T00053';
```

SWINBURNE
UNIVERSITY
TECHNOLOGY

2

Here are some examples of transactions.

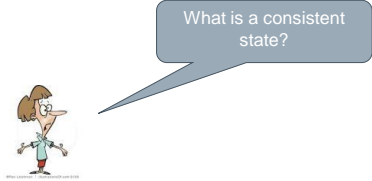
Transaction A consists of a single Read statement that retrieves all products in the lawn mower category.

Transaction B apparently stores a new delivery: It inserts a new tuple into the supply table, then increases the stock level of the product in another table.

Transaction C is a single update that corrects a misclassification of a product.

Definition

- › A Transaction is a logical unit of work on the database that transforms the database **from one consistent state to another**.



A cartoon character with a large head and small body, wearing a green shirt and blue pants, is pointing upwards with a speech bubble that says "What is a consistent state?".

SWINBURNE
UNIVERSITY
OF TECHNOLOGY

3

This is an important statement worth remembering.

A Transaction is a logical unit of work on the database that transforms the database **from one consistent state to another?**


What is a consistent state? A consistent state is a situation where no part of the database contradicts any other part of the database.

Inconsistent State


id	name	location	category
LW24453	Grass eliminator V5	A5B17	Lawn mowers
T00053	Makita Screwdriver M7	A13B5	Electric tools
CL19005	Kärcher HP5005	A2B22	Cleaners

prod_id	quantity	arrival
LW24453	3000	2015-9-26
T00053	1000	2015-9-25

prod_id	quantity
LW24453	0
CL19005	7


...

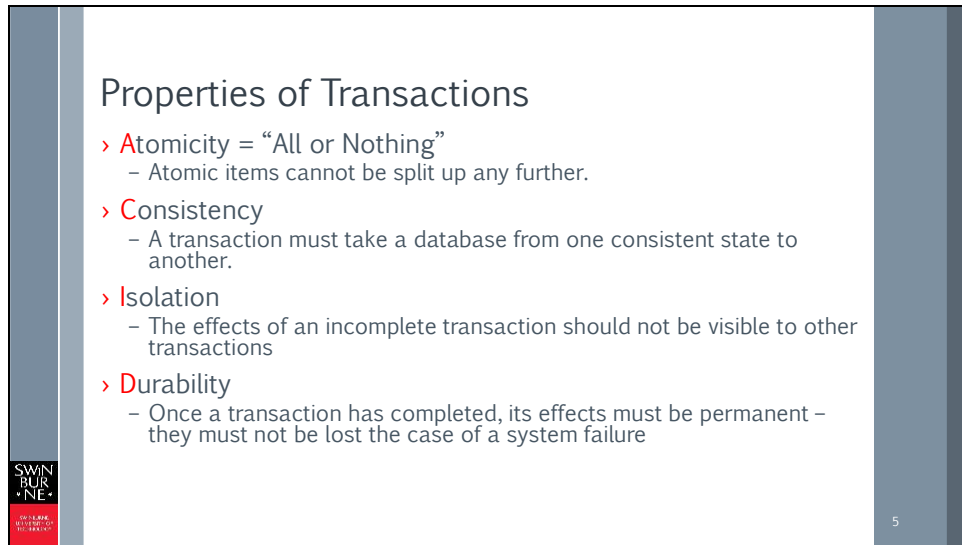
Where did the delivery of 3000 lawn mowers go??


4

Here we have an example of an inconsistent state.

Let's assume a company that sells tools and equipment has these three relations in their database: a Product relation, a Supply relation that stores the deliveries of the products, and one for the current stock level of each product.

When products are delivered, a new delivery tuple has to be inserted into the Supply relation. The delivery increases the stock level of the product, therefore the quantity attribute for the product has to be updated in the Stocklevel relation. If there is a delivery that does not increase the stock level, the database is inconsistent. In the worst case, the company can no longer take orders for the lawn mowers because, according to the database, there are none to sell.



Properties of Transactions

- > **A**tomicity = “All or Nothing”
 - Atomic items cannot be split up any further.
- > **C**onsistency
 - A transaction must take a database from one consistent state to another.
- > **I**solation
 - The effects of an incomplete transaction should not be visible to other transactions
- > **D**urability
 - Once a transaction has completed, its effects must be permanent – they must not be lost the case of a system failure

SWINBURN
UNIVERSITY OF
TECHNOLOGY

5

The ACID properties of transactions are a famous cornerstone of relational DBMSs. Every transaction is ‘atomic’, it can only be executed in its entirety or not at all. It cannot be split up into smaller transactions.

The database must remain consistent, and the effects of a transaction should not be visible outside the transaction context until the transaction is complete. Once a change has been made, it must not be lost.

The ACID properties have been criticised being a bit contrived – after all, the atomicity of transactions and their isolation are not goals in themselves. They merely help achieve the important goal of consistency. Durability is not necessarily transaction-related. It just means once a piece of information has been stored, the database management system or DBMS must not lose it.

Atomicity of Transactions

› Transaction A



```
UPDATE product SET category='Botanical stylers'
WHERE category='Lawn mowers';
```

› Transaction B

```
INSERT INTO supply (prod_id, quantity, arrival)
VALUES ('LW24453', 3000, '2015-09-26');

UPDATE stocklevel
SET quantity=quantity+3000
WHERE prod_id='LW24453';
```

Watch out for multi-statement transactions!



6

Database management systems always treat individual statements as atomic actions, even when they affect several tuples in the database. Only transactions that consist of several statements need special attention.

Commit and .. Rollback

> Transaction

```
INSERT INTO supply (prod_id, quantity, arrival)
VALUES ('LW24453', 3000, '2015-09-26');

UPDATE stocklevel
SET quantity=quantity+3000
WHERE prod_id='LW24453';

COMMIT;
```

Not permanent. Will be undone (rolled back) if we lose connection.

Not permanent. Will be undone (rolled back) if we lose connection.

Both changes are permanent now.

The diagram illustrates a database transaction. A light blue box on the left contains three SQL statements: an INSERT statement for the 'supply' table, an UPDATE statement for the 'stocklevel' table, and a COMMIT statement. To the right of this box are three speech bubbles. The first bubble points to the INSERT statement and contains the text 'Not permanent. Will be undone (rolled back) if we lose connection.' The second bubble points to the UPDATE statement and contains the same text. The third bubble points to the COMMIT statement and contains the text 'Both changes are permanent now.' The entire slide is framed by a dark blue border. In the bottom left corner, there is a small red logo with the text 'SWINBURNE' and 'UNIVERSITY OF TECHNOLOGY'. In the bottom right corner, the number '7' is displayed.

In most cases, the relational database management system starts a transaction when it executes the first statement of a transaction, such as this INSERT statement. The transaction ends when the COMMIT is issued. If there is a problem with the transaction, perhaps because we lose connection to the database management system, or because there is a conflict with another transaction, all effects of both the Insert and the Update statement will be undone – the database management system issues a rollback, which also terminates the transaction.

Transactions and Autocommit



> Transaction

```
INSERT INTO supply (prod_id, quantity, arrival)
VALUES ('LW24453', 3000, '2015-09-26'); COMMIT;
```

UPDATE stocklevel
SET quantity=quantity+3000
WHERE prod_id='LW24453'; COMMIT;

Should anyone be allowed to see this change without the other?

Plan your transactions!



8



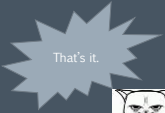
You may have worked with a relational database before, perhaps issued embedded SQL statements from a program. Or you may have used an SQL client and issued statements that were executed by the server. In both cases, you may never have issued a COMMIT statement, and yet, all your changes were made permanent. When you connect to a database, your session is often in Autocommit mode – every statement is committed immediately after it has been executed.

Making a basic connection to a relational database from a program, by default your statements are issued in Autocommit mode. It is good practice to switch Autocommit off after connecting to the database. The same applies if you are working with an SQL client application, such as php myadmin, or MySQL Workbench. If you have autocommit switched off, and you make a mistake, you can undo the changes without having to study the consequences and make up new SQL statements that reverse the problem.

When you are writing a program that uses a relational database, you have to plan your transactions. You have to think about how many statements it takes to take the database from one consistent state to another. You then have to bundle these into a single transaction. All this will only work if you have switched off autocommit.

Slide 9

Summary



- › Transactions are atomic interactions with a database which ensure that the database always appears consistent to the world.
- › Transactions can consist of one or more SQL statements.
- › The first statement initiates the transaction, the COMMIT or ROLLBACK statement ends the transaction.
- › Many database connections work on Autocommit mode by default, which makes multi-statement transactions impossible.
- › Plan your transactions carefully to ensure no other user can see the database while it is inconsistent..

9

Here are the most important points discussed in this module. You may want to stop the recording to have a read through them. When you are ready, start the quiz about this module.