HERIOT WATT UNIVERSITY

F27WD: Web Design & Databases

Databases Lecture 5:

Writing to the database and transactions

Fiona McNeill

11th March 2019

LEADERS IN IDEAS AND SOLUTIONS

HERIOT WATT UNIVERSITY

LEADERS IN IDEAS AND SOLUTIONS

The rest of the course

HERIOT WATT UNIVERSITY

After the feedback from you …

- Going over the stuff we've done so far is important to many of you
- But looking at the bigger world of data beyond what we are doing in this course is also important
- So …

HERIOT WATT UNIVERSITY

Lectures

Lecture	Topic	Who should go?	What else
Week10Lecture1	SQL & transactions	Everyone	
Week10Lecture2	Beyond SQL	Everyone	Not examinable
Week11Lecture1	Database management	Everyone	
Week11Lecture2	Practice with SQL, relational diagrams & PHP	Anyone who wants a bit more practice with the basics	Bring your laptop
Week12Lecture1	Course overview & exam preview	Everyone	Very helpful exam preparation
Week12Lecture2	No lecture	No one	

HERIOT WATT UNIVERSITY

Labs

Lab	Topic	Who should go?
Week10	PHP	everyone
Week11	Relational diagrams	everyone
Week12	Revision	everyone

HERIOT WATT UNIVERSITY

Changing databases

LEADERS IN IDEAS AND SOLUTIONS

HERIOT WATT UNIVERSITY

Dynamic data

One of the most important feature about data is that it can by **dynamic**.

What does this mean?

HERIOT WATT UNIVERSITY

Dynamic data

One of the most important feature about data is that it can by **dynamic**.

What does this mean? It means it can change over time.

Note that some data is **static** - which means it never changes. But a system that can only deal with static data is pretty limited.

HERIOT WATT UNIVERSITY

Dynamic data

Generally, in a large database things are changing **a lot**.

For example, an individual actor probably doesn't change their agent very often. But if you have a database of 100,000 actors, you'll need to make changes to actors' agents very frequently.

HERIOT WATT UNIVERSITY

Dynamic data

So creating a database with a schema you like and populating it with instances/rows is not enough. You also need to be able to change what is already there.

HERIOT WATT UNIVERSITY

What changes do we want to make?

So far we have seen how to create a **schema** using CREATE TABLE.

Changing this means changing the tables themselves - for example, adding, deleting or renaming columns.

We'll look at how to do this in the next lecture.

HERIOT WATT UNIVERSITY

What changes do we want to make?

For example, if we suddenly realise that it would be really useful to have actor's phone numbers, we need to alter our **actor table:**

Actor						
Name	DoB	Sex	Fee	Agent	Account	Phone No
Carl Pratt	1982-03-20	M	$1.1m	Smith	14839234	
Anna Stone	1985-02-17	F	$1.3m	Jones	18294038	
Rosie Ridley	1987-07-14	F	$1.5m	Jones	19204382	
Jemma Laurence	1980-12-01	F	$1.7m	Lane	13738925	

This is a **schema change.**

HERIOT WATT UNIVERSITY

What changes do we want to make?

We have also seen how to populate our schema (tables) with instances (rows) using INSERT INTO.

Changing this means we want to keep the structure of the table the same but want to alter a particular instance, or group of instances.

This is what we'll look at today.

HERIOT WATT UNIVERSITY

What changes do we want to make?

Deleting rows can be done using the DELETE FROM command:

DELETE FROM <*table name*> WHERE <*condition to tell you what rows to delete*>

e.g., DELETE FROM Actor WHERE name = “Anna Stone”;

Actor					
Name	DoB	Sex	Fee	Agent	Account
Carl Pratt	1982-03-20	M	$1.1m	Smith	14839234
Rosie Ridley	1987-07-14	F	$1.5m	Jones	19204382
Jemma Laurence	1980-12-01	F	$1.7m	Lane	13738925

HERIOT WATT UNIVERSITY

What changes do we want to make?

- We have to be careful about this.
- If other tables are pointing towards this data (through foreign keys) we can’t delete it - we need to delete that first.

HERIOT WATT UNIVERSITY

What changes do we want to make?

But if Rosie Ridley changed her agent, that's a change to an **instance.**

Actor					
Name	DoB	Sex	Fee	Agent	Account
Carl Pratt	1982-03-20	M	$1.1m	Smith	14839234
Anna Stone	1985-02-17	F	$1.3m	Jones	18294038
Rosie Ridley	1987-07-14	F	$1.5m	Lane	19204382
Jemma Laurence	1980-12-01	F	$1.7m	Lane	13738925

HERIOT WATT UNIVERSITY

Altering data

Today, we're going to look at **updating existing data** using the UPDATE command.

Say Rosie Ridley changed her agent to Lane.

We can change the database using:

UPDATE Actor SET agent = "Lane" WHERE name = "Rosie Ridley";

HERIOT WATT UNIVERSITY

Altering data

UPDATE Actor SET agent = “Lane” WHERE name = “Rosie Ridley”;

The UPDATE command:

HERIOT WATT UNIVERSITY

Altering data

UPDATE Actor SET agent = “Lane” WHERE name = “Rosie Ridley”;

The UPDATE command:

- needs the table that is being changed

HERIOT WATT UNIVERSITY

Altering data

UPDATE Actor SET agent = “Lane” WHERE name = “Rosie Ridley”;

The UPDATE command:

- needs the table that is being changed

- SET specifies the column that is being changed and the new value it is being changed to

HERIOT WATT UNIVERSITY

Altering data

UPDATE Actor SET agent = “Lane” WHERE name = “Rosie Ridley”;

The UPDATE command:

- needs the table that is being changed
- SET specifies the column that is being changed and the new value it is being changed to
- can also specify what row to alter - if there is no WHERE it will alter all of them.

HERIOT WATT UNIVERSITY

Altering data

UPDATE Actor SET agent = “Lane” WHERE name = “Rosie Ridley”;

- needs to know what row to alter

  *this is essentially saying alter **every** row where name = “Rosie Ridley”. Because this is the primary key, this will only be one row, but we don’t need to do this on the primary key.*

HERIOT WATT UNIVERSITY

Altering lots of data

What if we want to change lots of data according to certain constraints

- *for example, in our studio we have a new policy that no actor is ever paid more than $1.4m. So any actor with a fee higher than that has it lowered.*

HERIOT WATT UNIVERSITY

Altering lots of data

What if we want to change lots of data according to certain constraints

- for example, in our studio we have a new policy that no actor is ever paid more than $1.4m. So any actor with a fee higher than that has it lowered.

```
UPDATE Actor SET fee=1400000 WHERE
 fee > 1400000;
```

Here, fee is **not** the primary key, so this change can (and does in our example) apply to multiple rows.

HERIOT WATT UNIVERSITY

Altering complex things

When an actor makes a payment to their agent, their bank account needs to be lowered appropriately.

Say Jemma Laurence pays her agent $100,000.

We need to update BankAccount.balance - but we need to know which one and what to.

HERIOT WATT UNIVERSITY

Altering complex things

We can find out Jemma Laurence's account number and balance using SELECT and JOIN:

HERIOT WATT UNIVERSITY

Altering complex things

We can find out Jemma Laurence's account number and balance using SELECT and JOIN:

```
SELECT balance AS jemma_Balance,account AS
jemma_Account_Number FROM Actor JOIN BankAccount ON
number=account WHERE name="Jemma Laurence";
```

Jemma_Balance	Jemma_Account_Number
10390000	13738925

HERIOT WATT UNIVERSITY

Altering complex things

Now we can work out the new balance:

10,390,000 - 100,000 = 10,290,000

And we know what account we need to change (13738925), so we can change this:

UPDATE BankAccount SET balance= 10,290,000 WHERE number=13738925;

HERIOT WATT UNIVERSITY

Altering complex things

Now we can work out the new balance:

10,390,000 - 100,000 = 10,290,000

And we know what account we need to change (13738925), so we can change this:

UPDATE BankAccount SET balance= 10,290,000 WHERE number=13738925;

But this is a terrible way to update things! It is far too manual.

HERIOT WATT UNIVERSITY

Subqueries

We want SQL to automatically update Jemma's bank account when we provide it with her name and the amount.

First of all we need to find out Jemma's balance in a single transaction.

We need to start with

SELECT amount FROM BankAccount WHERE number = ??

HERIOT WATT UNIVERSITY

Subqueries

What could we use to frame a query like this?

HERIOT WATT UNIVERSITY

Subqueries

What could we use to frame a query like this? JOIN

HERIOT WATT UNIVERSITY

Subqueries

What could we use to frame a query like this? JOIN

But we can also use **subqueries**. This means using a SELECT command **inside** another SELECT command.

We'll have a look now at how that would work in this situation.

HERIOT WATT UNIVERSITY

Subquery

SELECT amount FROM BankAccount WHERE number = ??

To find out what her bank account number is, we need SQL to SELECT information from a different table - the Actor's table:

```
SELECT balance FROM BankAccount WHERE number=(
  SELECT account FROM Actor WHERE name="Jemma Laurence"
);
```

HERIOT WATT UNIVERSITY

Subquery

```
SELECT balance FROM BankAccount WHERE number=(
  SELECT account FROM Actor WHERE name="Jemma Laurence"
);
```

This is called a **sub-query** - because we have SELECT command nested within another SELECT command.

Now we've selected what we need, we need to put this sub-select into an UPDATE.

HERIOT WATT UNIVERSITY

Subquery

Should we use a JOIN or a SUBQUERY?

The example we've looked at is quite a simple situation, and here JOIN and subqueries do the same thing

=> we can use whichever we prefer (JOIN is probably neater).

But JOIN and subquery are not the same.

HERIOT WATT UNIVERSITY

JOIN v Subquery

Should we use a JOIN or a SUBQUERY?

- They are both used to combine data from different tables into a single result

HERIOT WATT UNIVERSITY

JOIN v Subquery

Should we use a JOIN or a SUBQUERY?

- They are both used to combine data from different tables into a single result
- JOINS are used to return **rows**

HERIOT WATT UNIVERSITY

JOIN v Subquery

Should we use a JOIN or a SUBQUERY?

- They are both used to combine data from different tables into a single result
- JOINS are used to return **rows**
- Subqueries can be used to return a single value or a row set.

HERIOT WATT UNIVERSITY

JOIN v Subquery

Should we use a JOIN or a SUBQUERY?

- They are both used to combine data from different tables into a single result
- JOINS are used to return **rows**
- Subqueries can be used to return a single value or a row set.

Don’t worry too much about what that means - looking into this is outside the scope of this course. What you should take from this is that subqueries exist and can be equivalent to JOINs in simple cases, but can also be used in more complex queries where JOIN wouldn’t work.

HERIOT WATT UNIVERSITY

Altering complex things

Remember the UPDATE and SET format:

UPDATE <*table name*> SET <*column = new value*> WHERE <*rows to be changed.*>

HERIOT WATT UNIVERSITY

Altering complex things

Remember the UPDATE and SET format:

UPDATE <*table name*> SET <*column = new value*> WHERE <*rows to be changed.*>

Column to be changed is Jemma's bank balance:

```
SELECT balance FROM BankAccount WHERE number=(
    SELECT account FROM Actor WHERE name="Jemma Laurence");
```

HERIOT WATT UNIVERSITY

Altering complex things

Remember the UPDATE and SET format:

UPDATE <*table name*> SET <*column = new value*> WHERE <*rows to be changed.*>

What is *new value*?

It is $100,000 less than the existing value.

```
balance = SELECT balance FROM BankAccount WHERE number=(
    SELECT account FROM Actor WHERE name="Jemma Laurence")
  - 100,000
```

HERIOT WATT UNIVERSITY

Altering complex things

So the whole command is:

```
UPDATE BankAccount SET balance=
(
  SELECT balance FROM (SELECT * FROM BankAccount) AS oldBalance
  WHERE number=(
    SELECT account FROM Actor WHERE name="Jemma Laurence"
  )
) - 100000
WHERE number=(
  SELECT account FROM Actor WHERE name="Jemma Laurence"
);
```

What is going on with this extra SELECT?

HERIOT WATT UNIVERSITY

Altering complex things

You can't SELECT from a table that you are currently UPDATING.

So
```
SELECT balance FROM BankAccount WHERE number=(
    SELECT account FROM Actor WHERE name="Jemma Laurence");
```
works as a sub-select, but won't work in this update because we are updating the BankAccount table.

So we use SELECT * FROM BankAccount to effectively create a copy of BankAccount to allow us to do this.

HERIOT WATT UNIVERSITY

Altering complex things

So the whole command is:

```
UPDATE BankAccount SET balance=
(
  SELECT balance FROM (SELECT * FROM BankAccount) AS oldBalance
  WHERE number=(
    SELECT account FROM Actor WHERE name="Jemma Laurence"
  )
) - 100000
WHERE number=(
  SELECT account FROM Actor WHERE name="Jemma Laurence"
);
```

HERIOT WATT UNIVERSITY

Altering complex things

Note that we are looking up the same thing twice:

```
UPDATE BankAccount SET balance=
(
  SELECT balance FROM (SELECT * FROM BankAccount) AS oldBalance
  WHERE number=(
    SELECT account FROM Actor WHERE name="Jemma Laurence"
  )
) - 100000
WHERE number=(
  SELECT account FROM Actor WHERE name="Jemma Laurence"
);
```

This can be done more efficiently - but we're not getting into how in this course.

HERIOT WATT UNIVERSITY

Dependent changes

So now we've altered Jemma's bank account appropriately - but that's only half the story.

We also have to update her agent's bank account by crediting it with the $100,000.

HERIOT WATT UNIVERSITY

Dependent changes

What does this look like?

```
UPDATE BankAccount SET balance=
(
  SELECT balance FROM (SELECT * FROM BankAccount) AS oldBalance
  WHERE number=(
    SELECT account FROM Agent WHERE name="Lane"
  )
) + 100000
WHERE number=(
  SELECT account FROM Agent WHERE name="Lane"
);
```

HERIOT WATT UNIVERSITY

Dependent changes

Of course, it's even better to do another sub-select finding out who Jemma's agent is rather than hard-coding into SQL.

Challenge: see if you can figure out how to do this in the lab or at home.

HERIOT WATT UNIVERSITY

When things go wrong!

So Jemma transferring $100,000 to her agent means:

1. Jemma's account is debited $100,000
2. Jemma's agent's account is credited $100,000.

So what happens if the system crashes between step 1 and step 2? The money goes missing …

HERIOT WATT UNIVERSITY

LEADERS IN IDEAS AND SOLUTIONS

Transactions

HERIOT WATT UNIVERSITY

When things going wrong!

Flight AZ493 to Brasilia on 14/02/18 has only one seat left.

Andrew sees there is availability at 16:02 and starts to book it, completing the booking at 16:08

Sarah starts to book the seat at 16:06 when it is still available, completing the booking at 16:11.

The same seat has been sold twice …

HERIOT WATT UNIVERSITY

We need to stop things going wrong!

We can think of loads of other similar situations where multiple things happening simultaneously lead to problems!

HERIOT WATT UNIVERSITY

We need to stop things going wrong!

We can think of loads of other similar situations where multiple things happening simultaneously lead to problems!

Of course, neither these examples should ever happen … because we control how databases are changed using **transactions.**

HERIOT WATT UNIVERSITY

What are transactions?

In this course, we are just going to briefly look at what transactions are and why they are important.

You'll go into this in much more detail in your second year databases course.

HERIOT WATT UNIVERSITY

What are transactions?

A **transaction** symbolises a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. A transaction generally represents any change in a database[1].

[1]Extracted from Wikipedia page on Database Transactions, extracted 10/01/18

HERIOT WATT UNIVERSITY

What are transactions?

Transactions in a database environment have two main purposes:

HERIOT WATT UNIVERSITY

What are transactions?

Transactions in a database environment have two main purposes:

1. To provide reliable units of work that allow correct recovery from failures and keep a database consistent even in cases of system failure, when execution stops (completely or partially) and many operations upon a database remain uncompleted, with unclear status.

HERIOT WATT UNIVERSITY

What are transactions?

Transactions in a database environment have two main purposes:

2 To provide isolation between programs accessing a database concurrently. If this isolation is not provided, the programs' outcomes are possibly erroneous.

HERIOT WATT UNIVERSITY

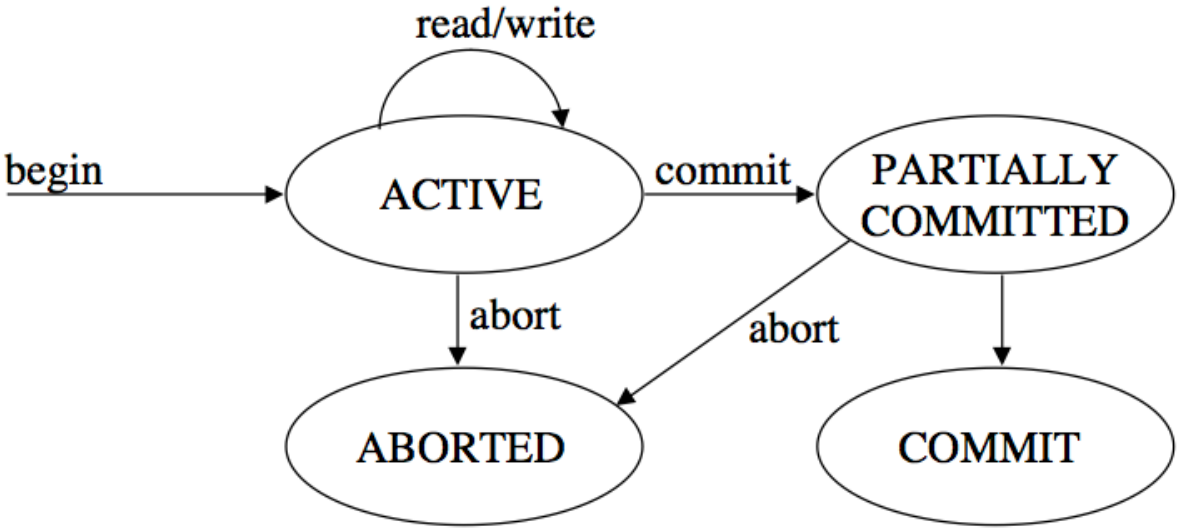
Transaction Management

- a transaction is the execution of a program that includes database access and begins with a START operation and ends with a COMMIT or ABORT operation;

HERIOT WATT UNIVERSITY

Transaction Management

- a transaction is the execution of a program that includes database access and begins with a START operation and ends with a COMMIT or ABORT operation;
- transaction state transition diagram:



HERIOT WATT UNIVERSITY

ACID properties

It's important that transactions adhere to **ACID** properties.

This means a DBMS must guarantee:

- **Atomicity**
- **Consistency**
- **Isolation**
- **Durability**

HERIOT WATT UNIVERSITY

Atomicity

A transaction's changes to a database are *atomic*, either all happen or none.

For example: during a bank transfer, you don't want to debit one account unless you credit the other or vice versa.

HERIOT WATT UNIVERSITY

Consistency

The actions of a transaction, taken as a group, must *not violate* any of the *integrity constraints* associated with the database.

For example, you can only delete a record that a foreign key points to if you also update the foreign key so that it points to something that will still exist after the transaction.

HERIOT WATT UNIVERSITY

Isolation (or serialisability)

Even though the DBMS may - for performance reasons - execute transactions concurrently, the results must be the same as if each transaction has been executed either before or after other transactions. Isolation is also known as **serialisability.**

This is why the flight example would be prevented. If Andrew completed his transaction, Sarah's would become impossible because the seat would already be sold. Therefore it must also be impossible even though the transaction actually overlap.

HERIOT WATT UNIVERSITY

Durability

Once a transaction completes successfully (commits), its changes to the database survive failures.

If you save something to the database and then it crashes, it will still be there when you restart the database.

HERIOT WATT UNIVERSITY

How do you guarantee this?

How, as someone who is creating a DB, can you bet sure that transactions are going to adhere to the ACID properties?

This is the job of the DBMS - you don’t have to worry about it.

HERIOT WATT UNIVERSITY

How do you guarantee this?

- Making sure this works is one of the roles of the DBMS - as a user, you can just take it for granted.

  The user describes the constraints; the DBMS enforces them.

  - The user describes the constraints using things like foreign keys.
- Next lecture, we'll look a little bit more into DBMS, including how they do this.