Schedules, a low-level view and notation

Overview of this video

The video will introduce a simplified and lower level view of transactions as well as schedules and notations for all of these

Translating SQL into Low-Level Operations

Employees(e_id, first_name, last_name, birth_day, salary)

Two SQL Statements

```
SELECT salary
FROM Employees
WHERE e_id = 1234;

UPDATE Employees
SET salary = salary*1.1
WHERE e_id = 1234;
```

produces →

THREE TRANSACTION OPERATIONS

1. read(e_id=1234, salary);
2. salary=salary*1.1;
3. write(e_id=1234, salary);

Notes:

Abstraction (at a low level)

Read data item 'salary' from tuple with primary key 1234

Two database operations: op1 (**read**) and op3 (**write**)

One non-database operation: op2 (the calculation)

Simplifying Low-level Operations

THREE TRANSACTION OPERATIONS

1. read(e_id=1234, salary);
2. salary=salary*1.1;
3. write(e_id=1234, salary);

produces →

THREE SIMPLIFIED TRANSACTION OPERATIONS

1. read(X);
2. X=X*1.1;
3. write(X);

X is in this case e_id 1234's salary, but it does not really matter and we omit it

Basic Operations of Transactions

read(X): Reads a database item X into a program variable (also named X, for simplicity)

- Find the address of the disk block (page) that contains item X
- Copy that disk block into a buffer in main memory
  - if that disk block is not already in some main memory buffer
- Copy item X from the buffer to the program variable X

write(X): Writes the value of program variable X into the database item named X

- Find the address of the disk block (page) that contains item X.
- Copy that disk block into a buffer in main memory
  - if that disk block is not already in some main memory buffer.
  - Copy item X from the program variable X into its correct location in the buffer
  - Store the updated block from the buffer back to disk either immediately or at some later point in time.

Transactions (in general)

A logical unit of processing using access operations

- Begin
- End
- read(retrieval – SELECT etc.)
- write(insert, update, or delete)
- + other non-database operations

Begin/end are are omitted
when the beginning/end
of a transaction are understood

Schedules

Schedules hold many transactions for execution

The operations making up the transactions are then executed by the schedule in some order

- It **must** preserve that the operations in each transaction happens in the right order!

Two types:

- Serial Schedules
  - Executes the transactions one after another (i.e. first each operation of the first schedule, then each operation of the second and so on)
- Concurrent Schedules
  - Can interleave operations from the transactions (while still preserving that the operations in each transaction happens in the right order) - formally speaking, a serial schedule is therefore also concurrent…

A Serial Schedule

Executes all operations in transaction T1, then all operations in transaction T2.

For simplicity we will typically ignore the non-database operations…

```
Begin (T1)
  read(X);
  X := X + 100;
  write(X);
  read(Y);
  Y := Y + 50;
  write(Y);
  commit;
End (T1)
Begin (T2)
  read(X);
  read(Y);
  X := X + Y;
  write(X);
  commit;
End (T2)
```

Shorthand Notation for Schedules

Shorthand notation for this schedule:

S_a: $r_1(X); w_1(X); r_1(Y); w_1(y); c_1; r_2(X); r_2(Y); w_2(X); c_2$

Symbols:

- S_{id} = schedule (*id* is the schedule ID)
- $r_i(X)$ = read(X) in transaction *i*
- $w_i(X)$ = write(X) in transaction *i*
- c_i = commit in transaction *i*
- a_i = abort ("rollback") in transaction *i*

```
Begin (T1)
    read(X);
    X := X + 100;
    write(X);
    read(Y);
    Y := Y + 50;
    write(Y);
    commit;
End (T1)
Begin (T2)
    read(X);
    read(Y);
    X := X + Y;
    write(X);
    commit;
End (T2)
```

Another Example

Time	S_b	X
t0		100
t1	read(X)	100
t2	X = X – 10	90
t3	write(X)	90
t4	Commit	90
t5	read(X)	90
t6	X = X * 10	900
t7	write(X)	900
t8	commit	900

What is the shorthand notation for this schedule?

Another Example

Time	S_b	X
t0		100
t1	read(X)	100
t2	X = X – 10	90
t3	write(X)	90
t4	Commit	90
t5	read(X)	90
t6	X = X * 10	900
t7	write(X)	900
t8	commit	900

What is the shorthand notation for this schedule?

S_b: $r_1(x); w_1(x); c_1; r_2(x); w_2(x); c_2$

Order matters:

Time	S_b	X
t0		100
t1	read(X)	100
t2	X = X – 10	90
t3	write(X)	90
t4	Commit	90
t5	read(X)	90
t6	X = X * 10	900
t7	write(X)	900
t8	commit	900

vs.

Time	S_c	X
t0		100
t1	read(X)	100
t2	X = X * 10	1000
t3	write(X)	1000
t4	Commit	1000
t5	read(X)	1000
t6	X = X – 10	990
t7	write(X)	990
t8	commit	990

Concurrent Schedule

Shorthand notation for schedule:

S_a: $r_1(X); w_1(X); r_1(Y); w_1(y); c_1; r_2(X); r_2(Y); w_2(X); c_2$

S_d: $r_2(X); r_2(Y); w_2(X); c_2; r_1(X); w_1(X); r_1(Y); w_1(y); c_1$

Note that these are serial schedules as well as a concurrent schedules (in that all serial schedules are concurrent schedules

Examples of other concurrent schedules (that are not serial):

S_e: $r_1(X); w_1(X); r_2(X); r_1(Y); w_1(y); c_1; r_2(Y); w_2(X); c_2$

S_f: $r_1(X); r_2(X); w_1(X); r_2(Y); r_1(Y); w_2(X); w_1(y); c_2; c_1$

Examples of something that is **not** a schedule:

S_g: $r_1(X); r_2(Y); w_1(X); r_2(X); r_1(Y); w_2(X); w_1(y); c_2; c_1$

```
Begin (T1)
    read(X);
    X := X + 100;
    write(X);
    read(Y);
    Y := Y + 50;
    write(Y);
    commit;
End (T1)
```

```
Begin (T2)
    read(X);
    read(Y);
    X := X + Y;
    write(X);
    commit;
End (T2)
```

Summary

We want to focus on low-level details (i.e. reads and writes) and their interaction over high-level (i.e. queries) when we talk about transactions (which is a sequence of queries) and schedules (which is a set of transactions that should be executed, so that each operation in each transaction comes in order)

We have a notations for such, using symbols:

- $\mathbf{S}_{id}$ = schedule (*id* is the schedule ID)
- $\mathbf{r}_i\mathbf{(X)}$ = read(X) in transaction *i*
- $\mathbf{w}_i\mathbf{(X)}$ = write(X) in transaction *i*
- $\mathbf{c}_i$ = commit in transaction *i*
- $\mathbf{a}_i$ = abort (“rollback”) in transaction *i*