Seven (+-2) Sins of Concurrency

Chen Shapira

In which I will show classical concurrency problems and some techniques of detecting and avoiding them

I have a B.Sc. in CS and Statistics, OCP, 10 years of production IT experience and I'm an Oracle Ace. So I know what I'm talking about.

But you don't have to trust me – I have scripts that prove everything I say.


Since 1967
computers
can walk and
chew gum at
the same
time

Programs need to learn to share

Example: Shared Bank Account

```
create or replace procedure update_account(p_id
   number,p_amount number) as
   n number;
begin
   SELECT amount into n FROM bank_account WHERE
   id=p_id;
   UPDATE bank_account SET amount = n+p_amount;
end;
```

```
SQL> exec deposit(1,500)

SQL> commit;
```

```
SQL> exec withdraw(1,-500)

SQL> commit;
```

```
SQL> select amount from bank_account;
AMOUNT
----------
-500
```

Sin #1

Race Condition

Can your code share?

Are you 100% sure?

Does this look familiar?

```
spool XXX_drop_db_links.sql

select 'drop database link '||OBJECT_NAME||';'
from obj
where OBJECT_TYPE='DATABASE LINK';

spool off

@XXX_drop_db_links.sql
```

Sin #2

Ostrich Algorithm

Few words about critical sections and mutual exclusion

Laws of Good Concurrency

- No two processes will be in their critical section at same time
- No assumptions about number or speed of CPUs
- No process outside the critical section may block other processes
- No process will wait forever to enter critical section

Mutual Exclusion in Oracle

Locks and latches and mutexes,
oh my!

User Defined Locks

```
dbms_lock.allocate_unique(
      lockname    => 'Synchronize',
      lockhandle  => m_handle
    );

n1 := dbms_lock.request(
      lockhandle          => m_handle,
      lockmode            => dbms_lock.x_mode,
      timeout             => dbms_lock.maxwait,
      release_on_commit       => true
    );
```

```
dbms_lock.allocate_unique('Synchronize',m_handle);
dbms_lock.request(m_handle,dbms_lock.x_mode,
  dbms_lock.maxwait,false);
spool XXX_drop_db_links.sql

select 'drop database link '||OBJECT_NAME||';'
from obj where OBJECT_TYPE='DATABASE LINK';

spool off

@XXX_drop_db_links.sql

dbms_lock.release(m_handle);
```

Another Race

```
select max(id) into max_id from my_table;

insert into my_table values (max_id+1,some_data);

commit;
```

Protecting the critical section - I

```
select max(id) into max_id from my_table for update;

insert into my_table values (max_id+1,some_data);

commit;

ERROR at line 1:
ORA-01786: FOR UPDATE of this query expression is
   not allowed
```

Protecting the critical section - II

```
select id into max_id from my_table where id=(select
   max(id) from my_table) for update;

insert into my_table values (max_id+1,some_data);

commit;
```

Protecting the critical section - III

```
select max_id into p_max_id from extra_table for
   update;

insert into my_table values (max_id+1,some_data);

update extra_table set max_id=max_id+1;

commit;
```

Sin #3

Solving the race condition led to serialization

The right way to do it:

```
create sequence my_table_seq start with 1
  increment by 1 cache 20;


insert into my_table
  (my_table_seq.nextval,some_data);
commit;
```

Quick Review

"Insanity: Doing the same thing over and over again and expecting different results."

Albert Einstein.

Classical Concurrency Problems

Dining Philosophers

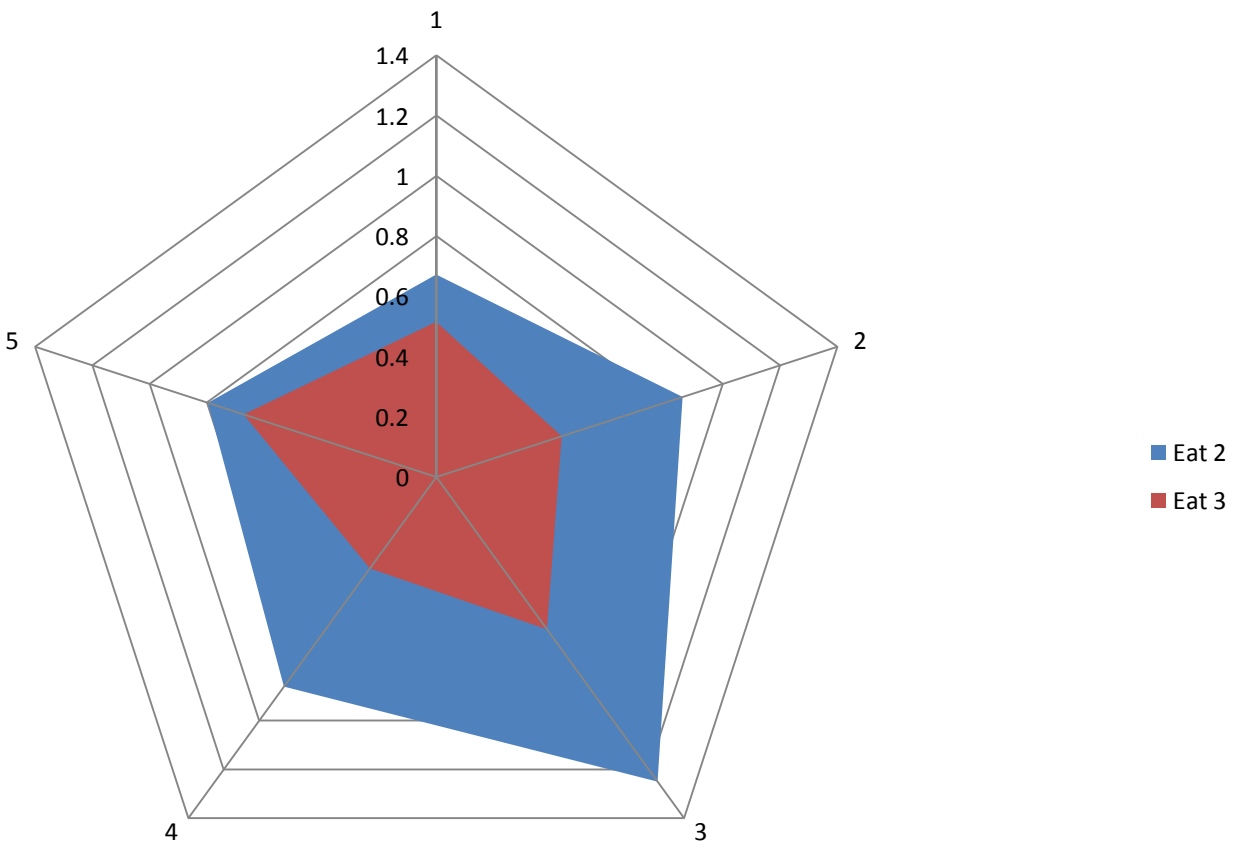
```
-- number of philosophers
select count(*) into N from sticks;
think();

update sticks set owner=philosopher_id where
  s_id=p_id; -- take right fork
update sticks set owner=philosopher_id where
  s_id=mod(p_id+1,N); -- take left fork

eat(); -- nom nom nom
commit; -- put down forks
```

ORA-00060: Deadlock detected

Sin #4

```
think();

update sticks set owner=in_p_id where s_id=in_p_id;

select s_id into r_s from sticks where
  s_id=mod(in_p_id+1,N) for update nowait;

update sticks set owner=in_p_id where
  s_id=mod(in_p_id+1,N);

eat();
commit;

exception
  when resource_busy then
      rollback;
```

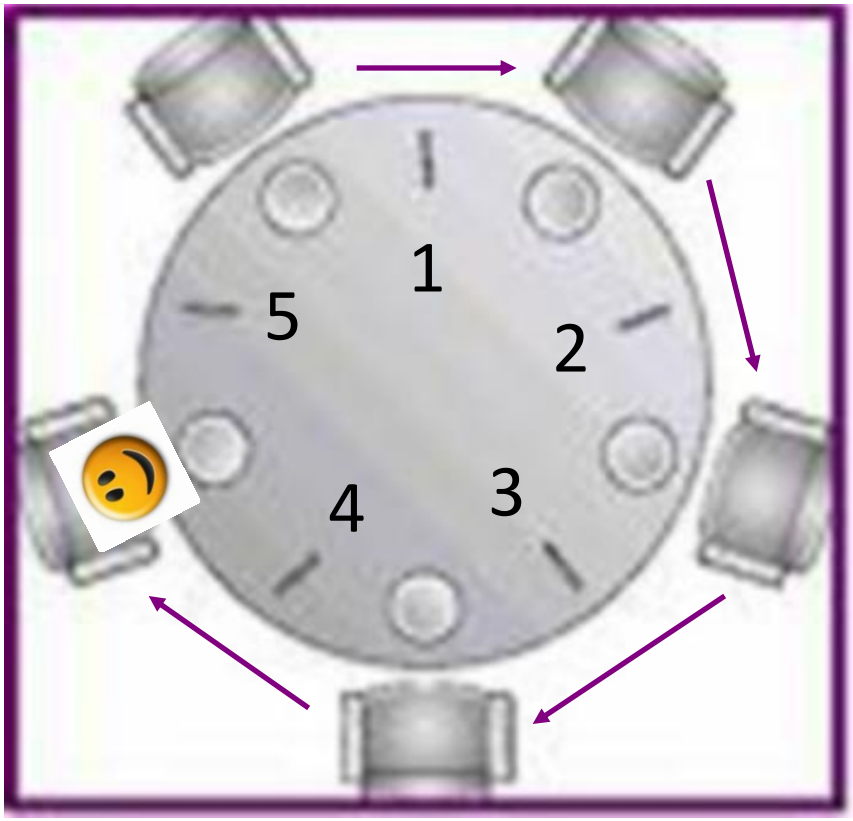
Starvation



Sin #5

```
think();

update sticks set owner=in_p_id where
  s_id=least(in_p_id,mod(in_p_id+1,N));

update sticks set owner=in_p_id where
  s_id=greatest(in_p_id,mod(in_p_id+1,N));

eat();

commit;
```

Partial Hierarchy Solution



Or just index your foreign keys!

Quick Review

Barbershop Queue

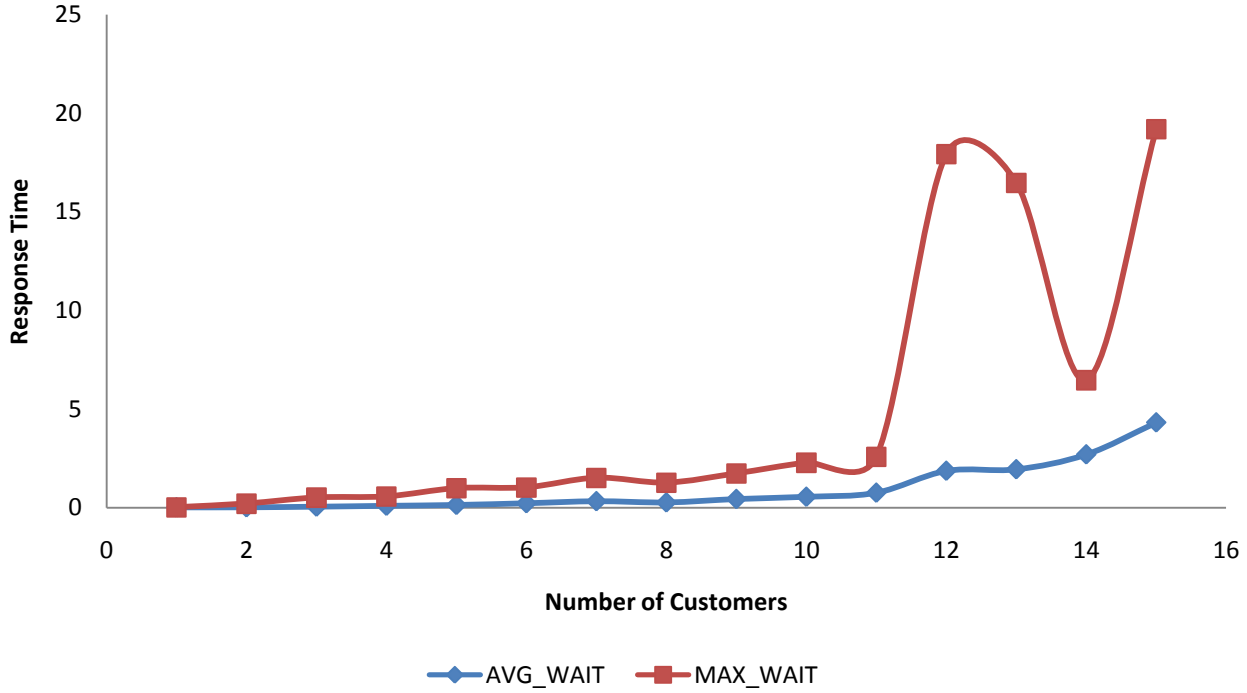
Generating customers

```
update customers set
needs_cut=1,entered_shop=systimestamp
where id in (
    select id from
           (select id from customers
           where needs_cut=0
           order by dbms_random.random)
    where
    rownum<=(dbms_random.value*(p_avg_customers_pe
    r_sec*2+1)));

commit;
dbms_lock.sleep(1);
```

Each barber does:

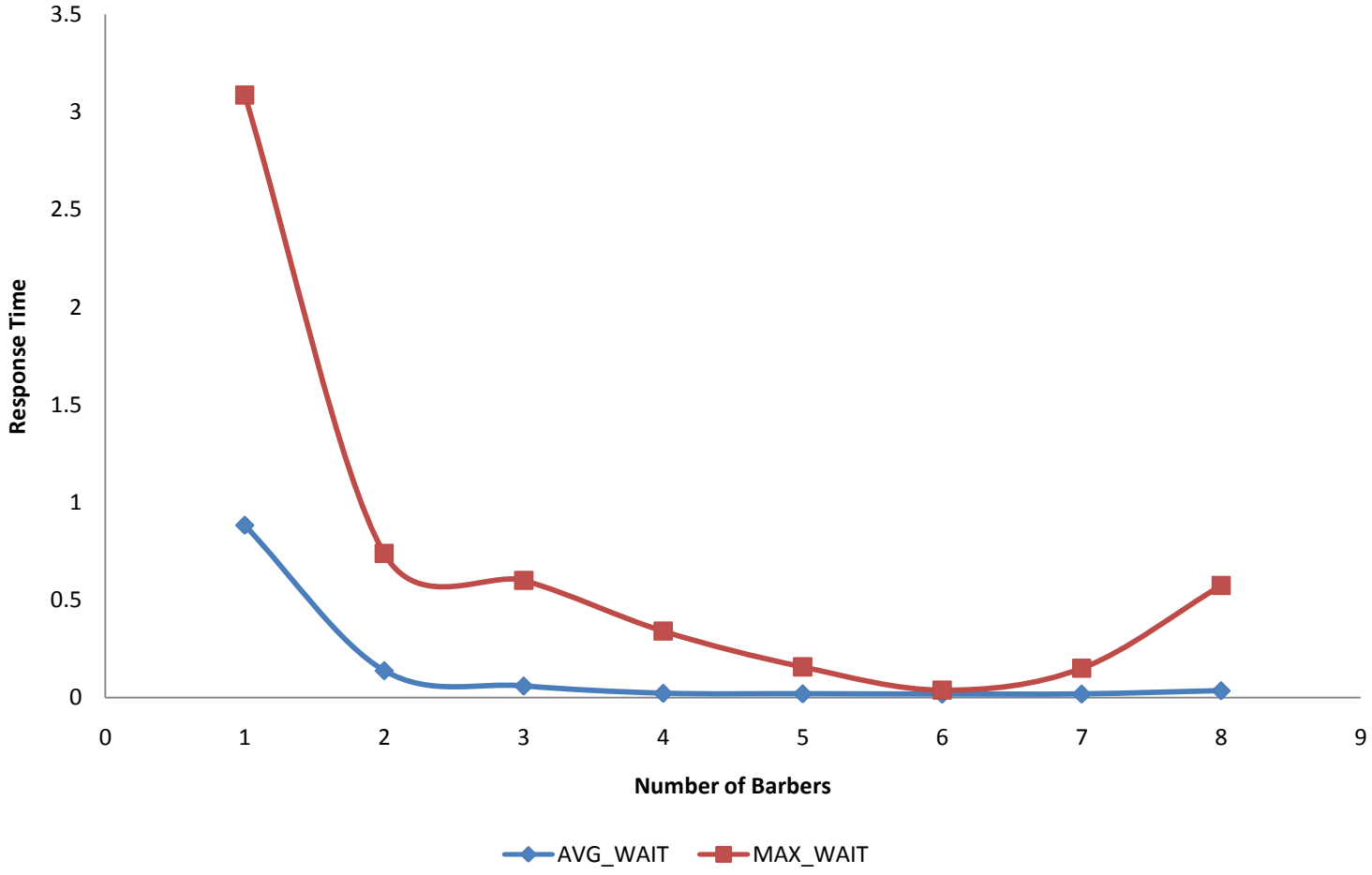
```
cursor c is select * from customers where
   needs_cut=1 order by entered_shop for update skip
   locked;
   …

fetch c into l_rec;
exit when c%NOTFOUND;
cut_hair(dbms_random.value*p_avg_cut_time*2);
finish_work(l_rec.id);
```

3 Barbers, Haircut in 0.3 seconds



Sin #6

About 10 concurrent customers



Sin #7

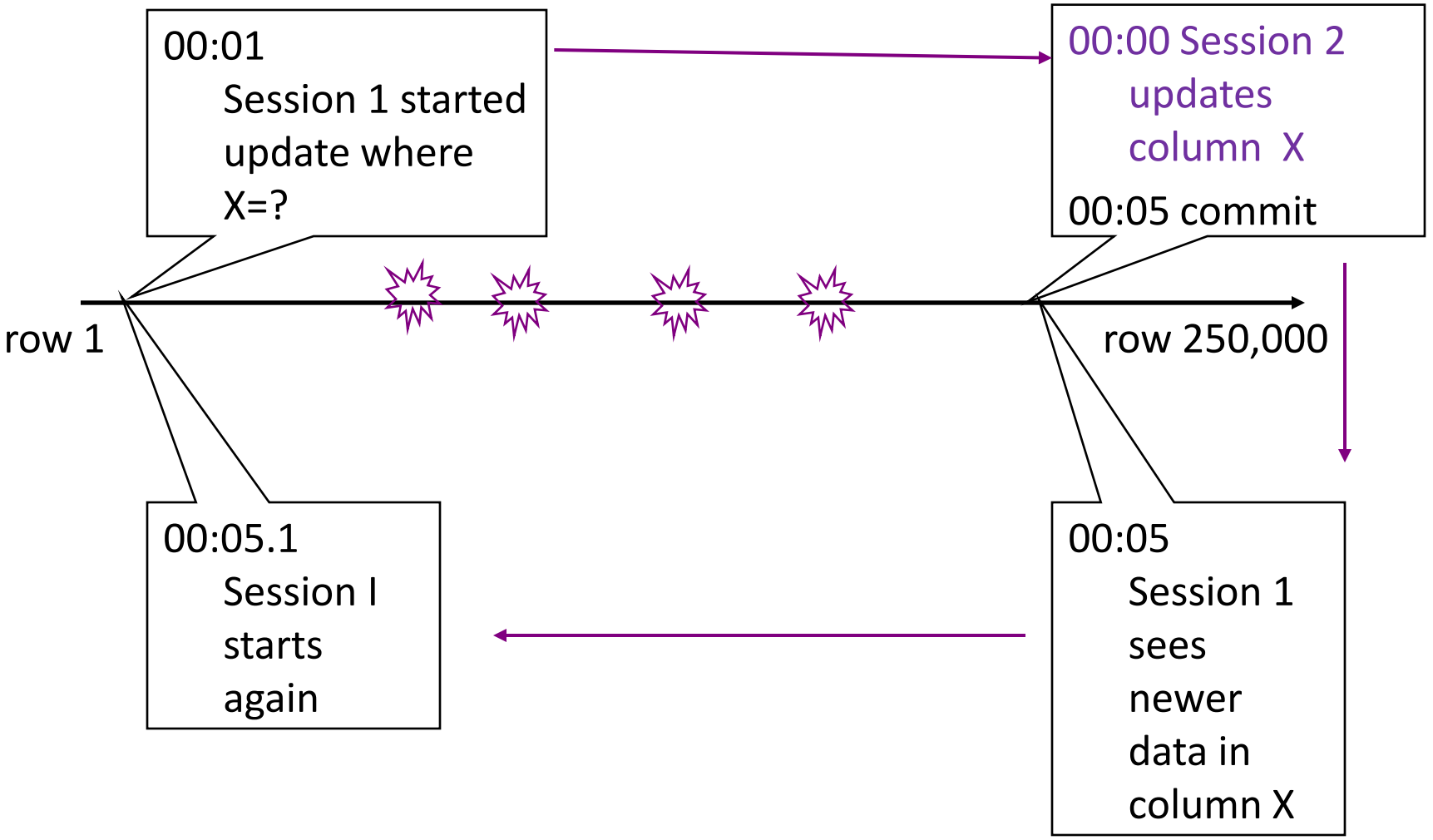
Quick Review

Oracle Concurrency Problems

Because consistency has a price

Non-transactional changes

Sin #8


00:01
Session 1 started update where X=?
00:00 Session 2 updates column X
00:05 commit
row 1
row 250,000
00:05.1
Session I starts again
00:05
Session 1 sees newer data in column X

Forgetting the extra IO

Sin #9