All About Table Locks:
DML, DDL, Foreign Key, Online Operations,...

About me

Franck Pachot
Senior consultant
Oracle Technology Leader

Mobile +41 79 963 27 22
franck.pachot@dbi-services.com
www.dbi-services.com

ORACLE Certified Professional
Oracle Database 12c
Administrator

ORACLE Certified Expert
Oracle Database 11g
Performance Tuning

ORACLE Certified Master
Oracle Database 11g
Administrator

ORACLE ACE

@FranckPachot

Developpez.com Club des développeurs
UKOUG UK ORACLE USER GROUP
DOAG
SOUG

Who we are
dbi services

Experts At Your Service

- 40 specialists in IT infrastructure
- Certified, experienced, passionate

Based In Switzerland

- 100% self-financed Swiss company
- Over CHF 6 mio. turnover

Leading In Infrastructure Services

- More than 100 customers in CH, D, & F
- Over 35 SLAs dbi FlexService contracted

Certified Management System SQS ISO/IEC 20000-1

Agenda

1. TM/TX, Share/Exclusive, Row/Table
2. Do we still need to index Foreign Key?
3. What is an online operation?

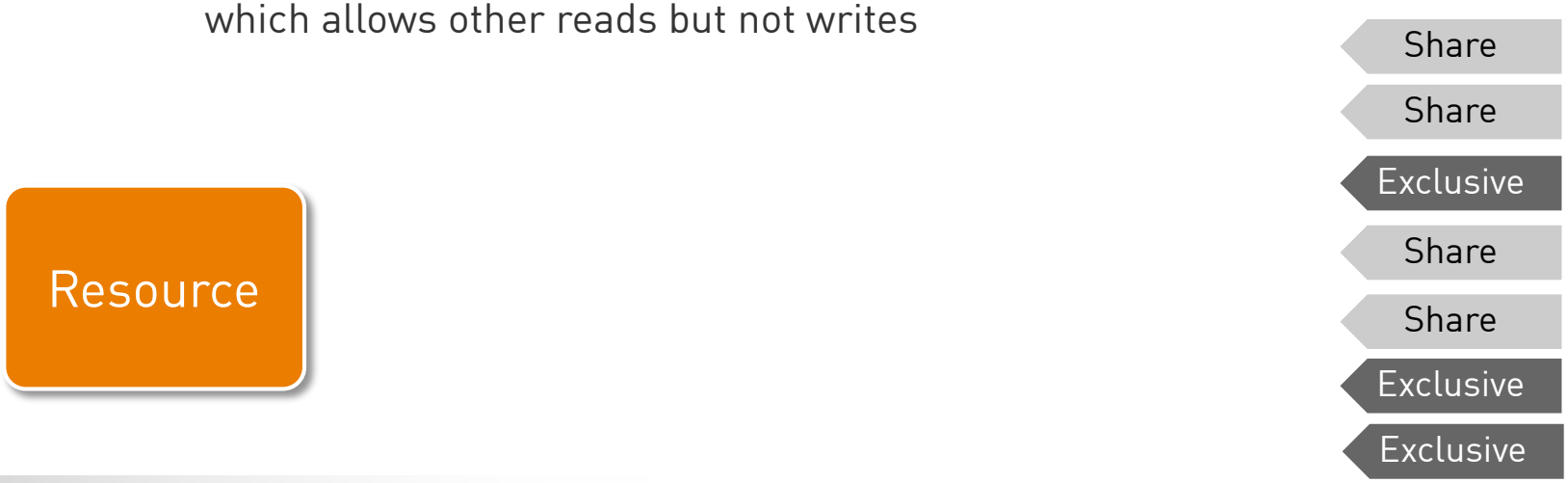
TX/TM, Share/Exclusive, Row/Table
Lock mode: Exclusive and Shared locks

Two sessions can't write at the same time

- Each write requests an exclusive lock
- The first one acquires the lock and can write
- The next ones wait to acquire the lock before they can write

You can't read while somebody writes,

- If there is an exclusive lock, then you wait
- But several sessions can read at the same time
- They acquire a shared lock
  which allows other reads but not writes



TX/TM, Share/Exclusive, Row/Table
Lock type: TX and TM

TX are row locks

- Acquired by row changes to enqueue transactions that modifies same row

TM are table locks

- Acquired by DML or DDL statements to enqueue other DML or DDL

There are other lock types. Here we talk only about TX and TM

```
SQL> select type,name from v$lock_type;

TYPE NAME
---- ----------------------------------
...
TM   DML
TX   Transaction
...

256 rows selected.
```

- Documented in V$LOCK documentation

TX/TM, Share/Exclusive, Row/Table
Lock type TX

TX controls what you can do on a row within a block

- When a transaction starts to write, it acquires a lock (exclusive)
- When it writes to a row (INSERT, DELETE, UPDATE, SELECT FOR UPDATE)
  - It writes its transaction id into the block header (ITL)
  - It flags the row as locked by that transaction id
- When another transaction wants to write on the same row
  - It check if the locking transaction has commited or not
  - It waits (enqueue) for that transaction to finish

- It's a row lock but the resource locked is a transaction.
  - Each transaction acquires a TX in exclusive mode
  - Each session that has to wait because of row lock request a TX

Lock type TX
demo

Transaction that modifies rows acquire a TX lock

- Resource locked is their transaction ID
- And flag each modified row with lock flag and ITL

Transactions that modifies locked row request TX lock

- Resource locked is the transaction that locked the row (from ITL)

```
SQL> select * from v$lock;

SID TYPE    ID1    ID2 LMODE REQUEST CTIME BLOCK
--- ---- ------ ------ ----- ------- ----- -----
 53 TX   196628 127933     6       0     6     1
 78 TX   196628 127933     0       6     1     0

SQL> select * from dba_locks;

SESSION_ID LOCK_TYPE  MODE_HELD  MODE_REQUE LOCK_ID1 LAS BLOCKING_OTH
---------- ---------- ---------- ---------- -------- --- ------------
        53 Transactio Exclusive  None       196628     7 Blocking
        78 Transactio None       Exclusive  196628     2 Not Blocking
```

TX/TM, Share/Exclusive, Row/Table
Lock type TX as an enqueue event

Session waiting on locks is instrumented by 'enqueue' wait event

- In V$SESSION

```
   SID STATE   EVENT                          P1RAW P1TEXT
------ ------- ------------------------------ ---------- ---------
    78 WAITING enq: TX - row lock contention  0054580006 name|mode
```

- 54 58 is 'T','X' in ASCII. Was useful in 9i as the name was only 'enqueue'
- 6 is the mode (lock mode '6' in V$LOCK is 'Exclusive' in DBA_LOCKS)

V$session has information about the row locked

```
ROW_WAIT_OBJ# ROW_WAIT_FILE# ROW_WAIT_BLOCK# ROW_WAIT_ROW#
------------- -------------- --------------- -------------
       116748              6             211             0
```

- Can be used to get the ROWID with DBMS_ROWID.CREATE
  - ROW_WAIT_OBJ# is the OBJECT_ID. Get DATA_OBJECT_ID from DBA_OBJECTS
  - ROW_WAIT_FILE# is the FILE_ID. Get RELATIVE_FNO from DBA_DATA_FILES

TX/TM, Share/Exclusive, Row/Table
Lock type TM

TM controls what you can do on a table

By DDL:

- TM Exclusive when you want to block any DML or DDL (that read or write)
- TM Share when you want to prevent only DML or DDL that write

By DML:

- It's not the modification but the intention to read or write
  - TM Row-X is the intention to write some rows
    - Must block/be blocked by DDL
  - TM Row-S is the intention to read some rows
    - Must block DDL that writes
  - Also called Sub-S and Sub-X: intention to read/write a subset of table

Lock type TM
demo

Session that has the intention to modify some rows

- Acquires Row-X TM lock
- Is compatible with other Tow-X TM locks

```
SQL> select * from dba_locks  where LOCK_TYPE in
('Transaction','DML');

SESSION_ID LOCK_TYPE MODE_HELD  MODE_REQUE LOCK_ID1   BLOCKING_OTH
---------- --------- ---------- ---------- ---------- ------------
        55 DML       Row-X (SX) None       116881     Not Blocking
        65 DML       Row-X (SX) None       116881     Not Blocking
```

- Lock ID1 is the OBJECT_ID

Primary goal is to prevent DDL

- CREATE INDEX requests a Share lock

```
SESSION_ID LOCK_TYPE  MODE_HELD  MODE_REQUE LOCK_ID1   BLOCKING_OTH
---------- ---------- ---------- ---------- ---------- ------------
        55 DML        None       Share      116881     Not Blocking
        63 DML        Row-X (SX) None       116881     Blocking
```

TM/TX, Share/Exclusive, Row/Table
Updating rows

When the intention to insert/update/delete :

- Acquires TM Row-X on table before touching any rows

For each block that is updated:

- Write the transaction in ITL in block header (link to TX lock)
- Acquires TX share on the transaction
  - Can wait here 'enq TX- allocate ITL' mode 4

For each row inserted/updated/deleted:

- Flags the row as locked by the transaction
- Acquires TX exclusive on the transaction
- If already locked, waits on the locking transaction
  - Can wait here 'enq: TX - row lock contention' mode 6

TM/TX, Share/Exclusive, Row/Table Row- and Sub-

When DML scope is the whole table

- Acquires TM Share or Exclusive

When DML scope is a subset (some rows, or a partition, ...)

- Acquires TM Row-S or TM Row-X, or RS or RX
- Also called Sub-X or Sub-X, or SS or SX

	Mode	Names	When (examples)
DML	2	Row-S,RS,SS	Select for update <10g
DML	3	Row-X,RX,SX	Insert, Update, Delete, Merge
DDL	4	Share	Create index
DDL	5	Share Row-X, SSX	When RX and Share
DDL	6	eXclusive	Insert append, PDML

Non blocking reads
demo

SELECT do not request any lock

```
11:55:18 SID=60> drop table SCOTT.DEMO purge;
Table dropped.
11:55:45 SID=60> select * from SCOTT.DEMO;
select * from SCOTT.DEMO
                    *
ERROR at line 1:
ORA-00942: table or view does not exist
```

- I can even read from a dropped table:

```
 N
----------
        98
        99
       100
100 rows selected.
11:56:31 SID=63>
```

SELECT (without for update) reads a past image

- No concurrent writes on that - no lock needed

Agenda

1. TM/TX, Share/Exclusive, Row/Table
2. Do we still need to index Foreign Key?
3. What is an online operation?

Do we still need to index Foreign Key?
Demo with/without index

EMP (DEPTNO) references DEPT, with index

- We delete from DEPT

```
SID=41> delete from DEPT where DEPTNO=50;
0 rows deleted.
```

- Remaining locks:

```
SESSION_ID OWNER      NAME       MODE_HELD  MODE_REQUE
---------- ---------- ---------- ---------- ----------
41 SCOTT      DEPT       Row-X (SX)
41 SCOTT      EMP        Row-X (SX)
```

EMP (DEPTNO) references DEPT, without index

- We delete from DEPT
- Remaining locks:

```
SESSION_ID OWNER      NAME       MODE_HELD  MODE_REQUE
---------- ---------- ---------- ---------- ----------
41 SCOTT      DEPT       Row-X (SX)
```

Do we still need to index Foreign Key?
What happens?

EMP (DEPTNO) references DEPT, without index

- We delete from DEPT
  - A TM-Share lock is acquired on EMP
  - Is released as soon as the delete is done
- No lock remains, but may be long to acquire
  - because it enqueues behind any DML
  - And blocks further DML

EMP (DEPTNO) references DEPT, with index

- We delete from DEPT
  - No TM-Share lock is required thanks to the index
  - But any operation on parent or child need a TM Row-X on child on both
  - Can be TM Row-S in some situations/version

Do we still need to index Foreign Key?
Demo on multiple tables

EMP (DEPTNO) references DEPT, with index

- Insert into BONUS
  - Locks EMP in Row-X
- Delete from DEPT
  - Waits to acquire Share
- Any other DML on tables that are linked to EMP
  - Waits to acquire Row-X

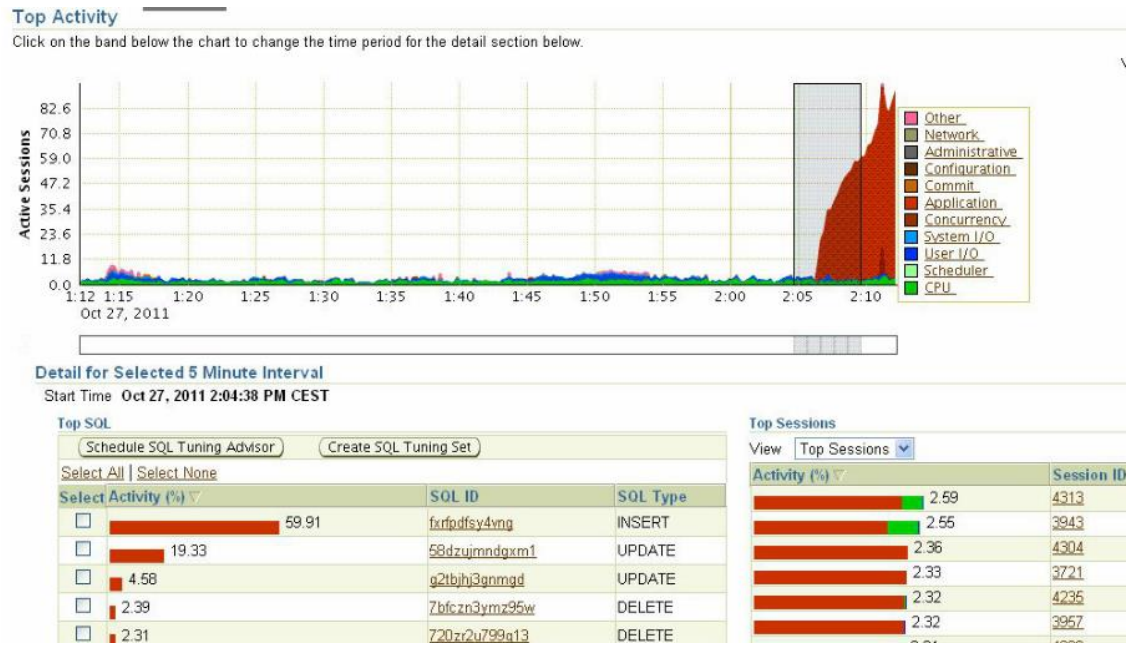
```
SID NAME   MODE_HELD   MODE_REQ  SEC BLOCKING_OTH
--- ------ ----------- -------- --- ------------
 51 EMP    Row-X (SX)  None      81 Blocking
 51 BONUS  Row-X (SX)  None      81 Not Blocking
 41 DEPT   Row-X (SX)  None      16 Not Blocking
 41 EMP    None        Share     16 Not Blocking
 45 EMP    None        Row-X (SX) 8 Not Blocking
```

- Other DML related with the child are enqueued behind the TM-Share

Do we still need to index Foreign Key?
What happens?

TM-Share is quick, but...

- Only if child is small (full scan to check no child exists)
- Only if you have low DML activity on child table (or has to wait)
- and low DML activity on any table linked with child table (since 11g)



- A delete small lookup table can block the whole OLTP

Do we still need to index Foreign Key?
Row-S and Row-X

Any DML on ne table acquires TM Row-S/X on opposite table

- Mainly to avoid deadlocks in special combination of operations
- Was a big issue when migrating to 11g

Operation on parent	Lock on child			
	10g	**11g**	**12c**	
Insert	RS	RX	**RS**	
Delete	RS	RX	RX	*Or Share if unindexed foreign key*
Update	RS	RX	RX	

Operation on child	Lock on parent			
	10g	**11g**	**12c**	
Insert	RS	RX	RX	
Delete	RS	RX	**RS**	
Update	RS	RX	RX	

Do we still need to index Foreign Key?
Why?

We want to avoid the following situation:

- User A inserts a row into the child table
  - Check that parent exists (but don't lock it... see why later)
- User B deletes the parent
  - transaction isolation (ACID) allows that, A did not commit yet
- Both A and B commit and we have an orphean

This can be avoided by locks

- Lock the parent row when we insert into the child
- Lock the child row when we delete from parent

But...

- It's not so easy...

Do we still need to index Foreign Key?
Why not lock parent row?

I want to delete DEPTNO 40

- Concurrent inserts must be blocked

EMPNO	ENAME	DEPTNO
7369	SMITH	20
7499	ALLEN	30
7521	WARD	30
7566	JONES	20
7654	MARTIN	30
7698	BLAKE	30
8000	LARRY	40

DEPTNO	DNAME
10	ACCOUNTING
20	RESEARCH
30	SALES
40	OPERATIONS

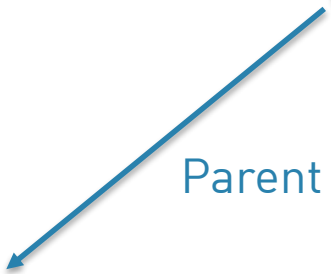
First idea:
lock the parent row and wait
to see if committed or rolled back

Do we still need to index Foreign Key?
Why not lock parent row?

Can't insert with DEPTNO 40

```
EMPNO ENAME   DEPTNO
----- ------ ------
 7369 SMITH      20
 7499 ALLEN      30
 7521 WARD       30
 7566 JONES      20
 7654 MARTIN     30
 7698 BLAKE      30
 8000 LARRY      40
```

```
DEPTNO DNAME
------ ----------
    10 ACCOUNTING
    20 RESEARCH
    30 SALES
```



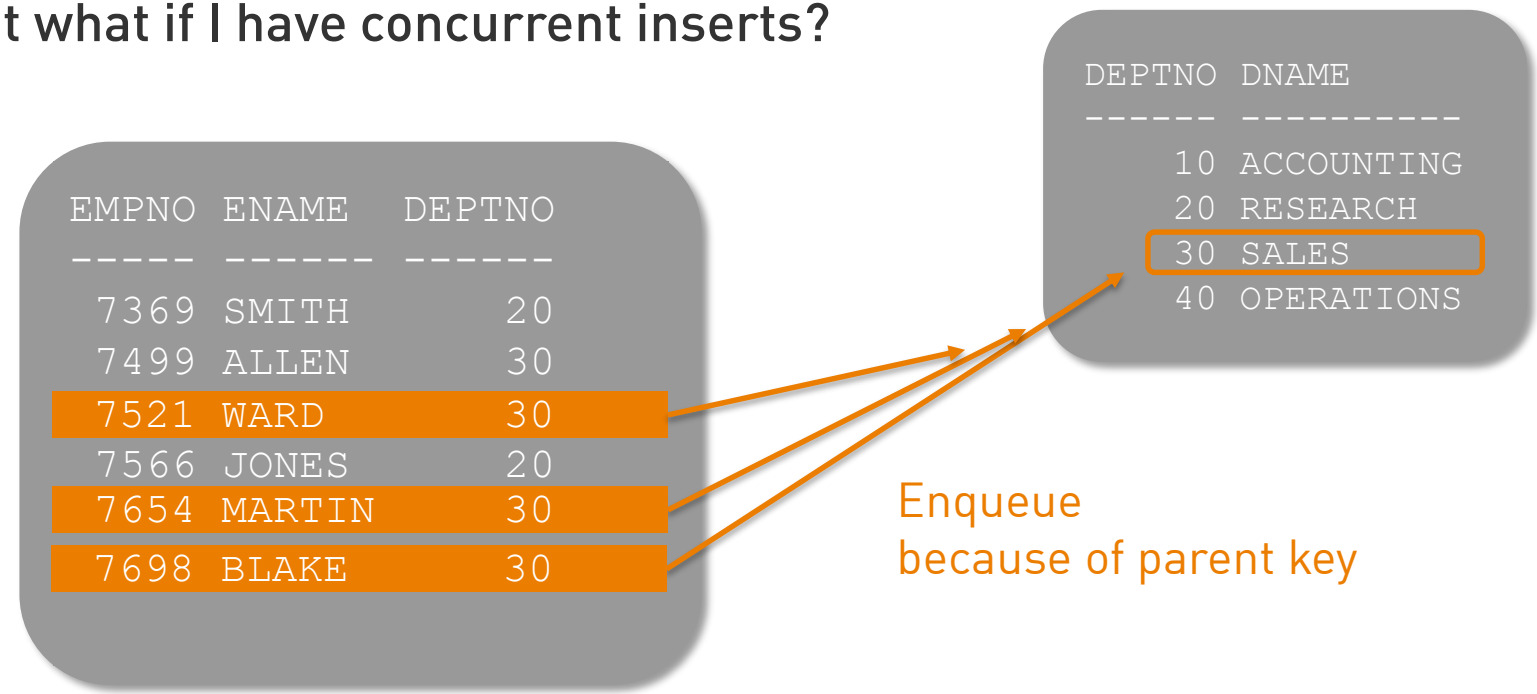
Parent key not found

- This could be a way to enable referential integrity

but...

Do we still need to index Foreign Key?
Why not lock parent row?

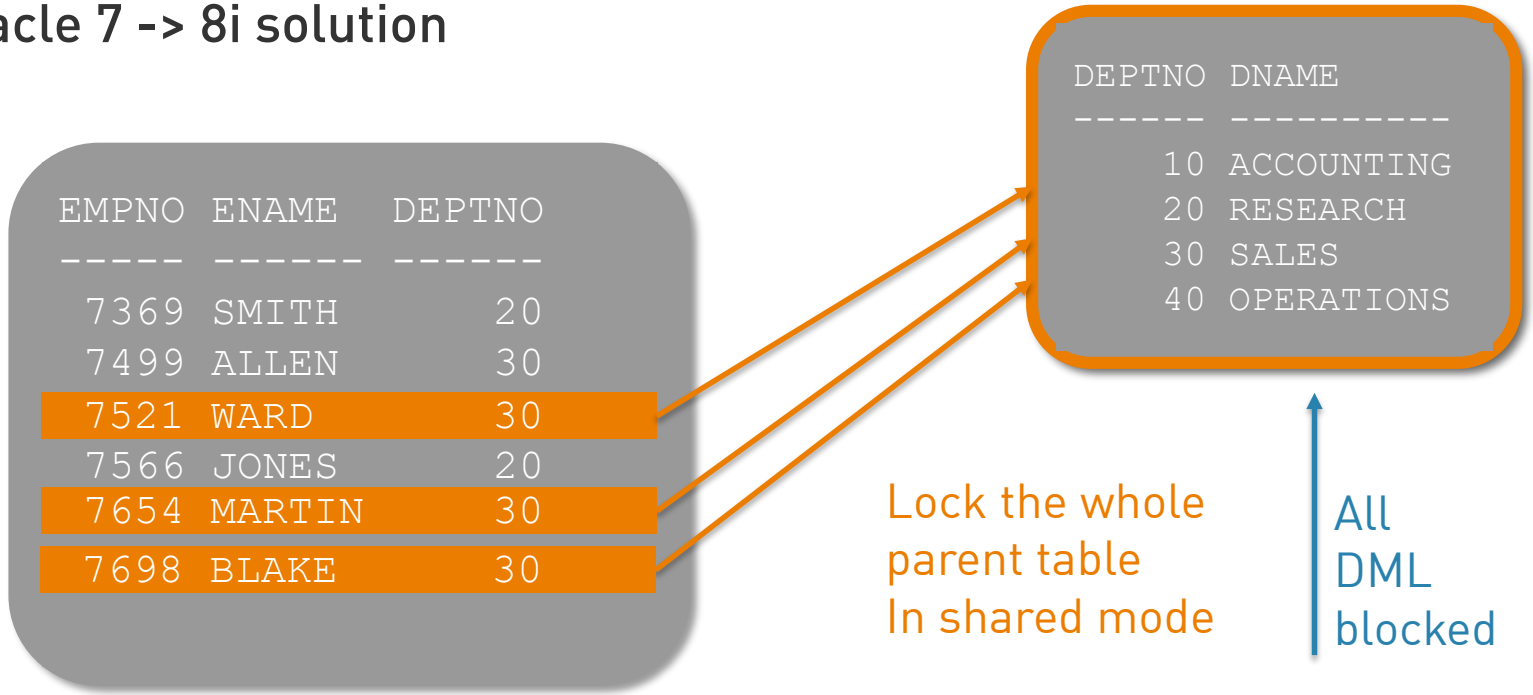
But what if I have concurrent inserts?



- Oracle do not have shared row lock. Only exclusive.
- So concurrent inserts on same parent locks themselves (even without DML on parent).
- So that solution (lock parent row) has never been implemented

Do we still need to index Foreign Key?
Why not lock parent table?

Oracle 7 -> 8i solution



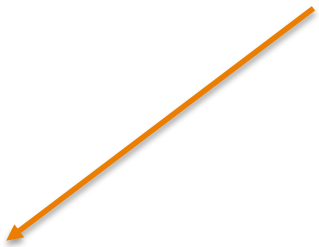
- No problem with concurrency because it's shared.
- But locks de whole table:
  any DML on parent, for any value blocks the insert.

Do we still need to index Foreign Key?
Why not lock child rows?

I want to lock a range of value

```
EMPNO ENAME   DEPTNO
----- ------ ------
 7369 SMITH      20
 7499 ALLEN      30
 7521 WARD       30
 7566 JONES      20
 7654 MARTIN     30
 7698 BLAKE      30
 %%%% %%%%%     40
```

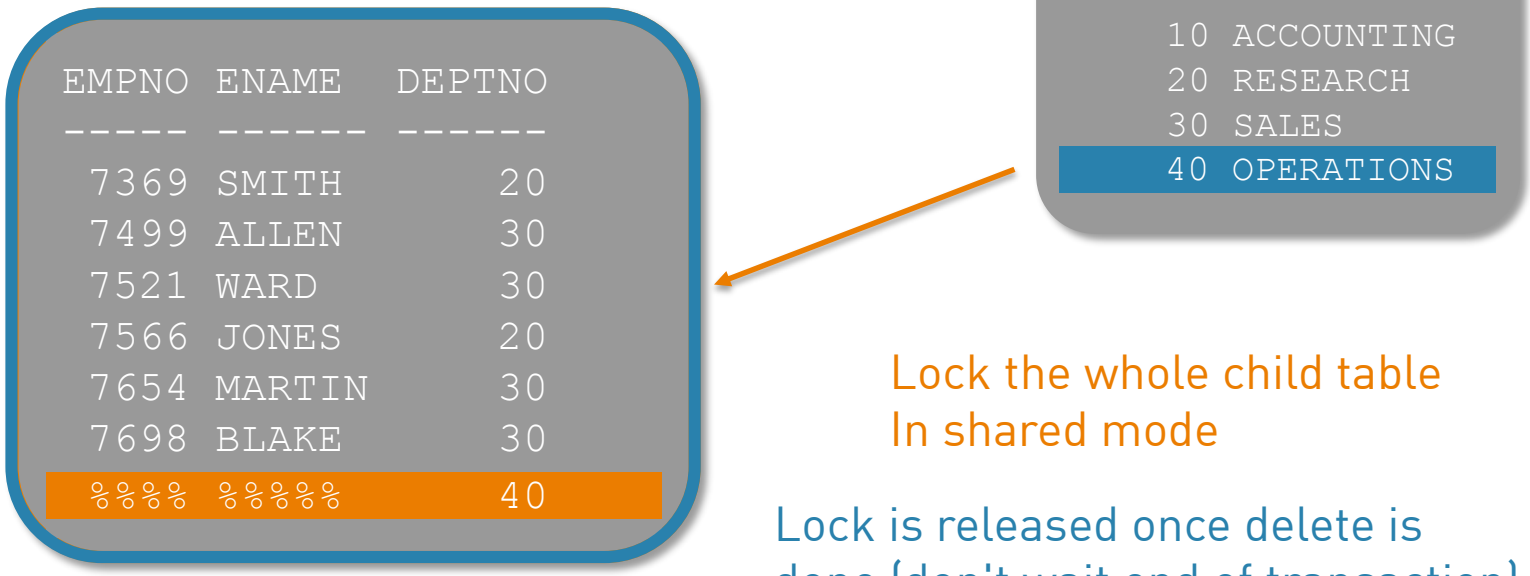
```
DEPTNO DNAME
------ ----------
    10 ACCOUNTING
    20 RESEARCH
    30 SALES
    40 OPERATIONS
```



- But Oracle do not have table range locks.
- You cannot lock a row that is not there (yet)

Do we still need to index Foreign Key?
Why not lock child table?

Oracle 9i -> 12c solution



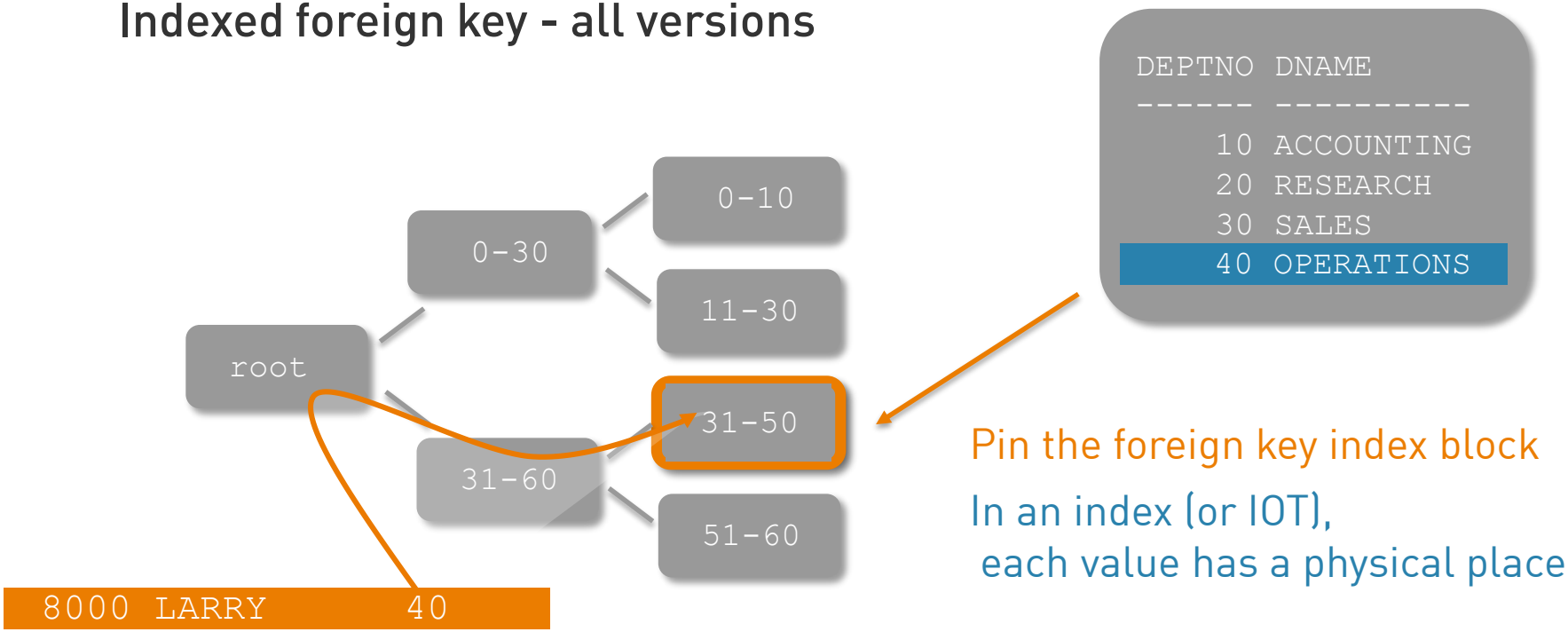
Lock the whole child table
In shared mode

Lock is released once delete is done (don't wait end of transaction)

- It's the delete on parent that locks the child table to prevent inserts.
- In Heap Table you must lock the table to prevent any insert.

Do we still need to index Foreign Key?
Why not lock child range?

Indexed foreign key - all versions



- With an index on foreign key:
  - Delete on parent can pin the buffer to check for uncommitted insert
  - If there is one, wait on TX in mode 4 (share) for the end of transaction

Do we still need to index Foreign Key?
Which index?

```
table CHILD (a,b,c) FOREIGN KEY (a,b) REFERENCES PARENT(a,b)
```

```
create index on (a,b)
create index on (b,a)
create index on (a,b,c)
create index on (a,b,c) compress
create unique index on (a,b,c)
create index on (a,b,upper(c))
create table primary key(a,b,c) organization index
```

```
create index on (c,b,a)
create index on (a,c,b)
create index on (a,upper(b))
create index on (a,b desc)
create bitmap index on (a,b)
```

- Don't index blindly, index first for performance and then see what you need to add for locking issues

Do we still need to index Foreign Key?
What to do?

When is a Share lock on child table requested?

- Delete on parent
- Update of referenced column in the parent table
- Bugs (cascade constraints even if it doesn't touch the referenced column)

How to avoid the TM-Share?

- Have a regular index that starts with the foreign key
  - When the insert goes to one block, no lock is required
    - Must start with the foreign key
    - Index is also used to check if there are child
- IOT is an index
- Bitmap index are delayed so can't be used
  - https://jonathanlewis.wordpress.com/2014/04/18/bitmap-loading/
- Or don't delete from parent table (flag it and delete at night)

Agenda

1. TM/TX, Share/Exclusive, Row/Table
2. Do we still need to index Foreign Key?
3. What is an online operation?

What is an online operation?
Read vs DML vs DDL

SELECT query

- No lock because it reads past image that nobody can update

INSERT,UPATE,DELETE,MERGE, SELECT FOR UPDATE

- Row-X on the target table
- Row-S or Row-X on tables linked by referential integrity
- Share in the unindexed foreign key case

DDL

- That only read (e.g create index): Share
- That modifies: eXclusive

An online operation is an operation that allow concurrent DML

- Only Row-S and Row-X locks
- Warning: some 'online' operations can require Share for a short period

What is an online operation?
Concurrency vs resources

Usually an online operation

- Need more resources
  - More DB time
  - More storage space
  - Takes longer
- But nobody cares
  - The service is still available

Most of online operations need Enterprise Edition

- REBUILD ONLINE, MOVE ONLINE, DBMS_REDEFINITION

Exception:

- DROP INDEX ONLINE
- ALTER INDEX [IN]VISIBLE is online in 12c (even in Standard Edition)

What is an online operation?
What's new in 12c

12c

- Enterprise Edition

```
ALTER DATABASE MOVE DATAFILE … TO …; No lock
ALTER INDEX … UNUSABLE ONLINE; RS
ALTER TABLE … DROP CONSTRAINT … ONLINE; RX
ALTER TABLE … SET UNUSED(…) ONLINE; RX
ALTER TABLE … MOVE PARTITION … ONLINE; RX
```

- Standard Edition

```
DROP INDEX ... ONLINE; RS
ALTER INDEX … INVISIBLE;RS
ALTER INDEX … VISIBLE; RS
```

  - In 11g ALTER INDEX [IN]VISIBLE acquires eXclusive lock.

11g

- ALTER INDEX ... REBUILD ONLINE is really online (no TM-Share)

What is an online operation?
How to trace?

```
oerr ora 10704
10704, 00000, "Print out information about what enqueues are being
obtained"
// *Cause:  When enabled, prints out arguments to calls to ksqcmi and
//          ksqlrl and the return values.
// *Action: Level indicates details:
//   Level: 1-4: print out basic info for ksqlrl, ksqcmi
//          5-9: also print out stuff in callbacks:  ksqlac, ksqlop
//          10+: also print out time for each line
```

Event 10704

- ksqgtl: get lock
- ksqrcl: release lock
- ksqcnv: convert lock

```
alter session set events=10704 trace name context forever, level 3';
```

```
alter session set events='10704 trace name context off';
```

What is an online operation?
demo

alter table DEPT move;

```
ksqgtl *** TM-0001CD17-00000000-00000000 mode=6 flags=0x400 timeout=0
```

- mode=6 is eXclusive: prevent any DML

alter index PK_DEPT rebuild;

```
ksqgtl *** TM-0001CD17-00000000-00000000 mode=4 flags=0x400 timeout=0
```

- mode=4 is Share: prevent modification

alter index PK_DEPT rebuild online;

```
ksqgtl *** TM-0001CD17-00000000-00000000 mode=2 flags=0x400 timeout=0
```

- mode=2 is Row-S: compatible with all DML and some DDL

All about table locks
Core Message

Lock behavior is complex but basic mechanisms are simple

We still need to index foreign keys

- When parent have delete (or some updates)
- When child or linked table have DML activity

Don't wait to see blocked sessions: trace with 10704

- Trace foreign key lock situations
- Trace online operations to be sure they are online

Any questions? Please do ask.