Oracle Composite Indexes and Foreign Key Constraints

Andy Rivenes
AppsDBA Consulting

The following will describe how composite indexes can be used to index foreign key constraints. In Oracle, foreign key constraints should be indexed if the parent table incurs updates or deletes. With unindexed foreign key columns, Oracle will lock the child table when an update (rare – you really shouldn't be updating a primary key) or a delete occurs on the parent table. In the case of multiple foreign keys in a child table, there are times when a composite index (e.g. multi-column) provides a better access path than two separate indexes. The following will first show what happens with indexed and unindexed foreign key columns and then will show how to implement a composite index on multiple foreign key columns.

A simple setup was made that consisted of two "parent" tables and two "child" tables. Each parent table had a single primary key column and each child table had a primary key column and two foreign key columns, each pointing to the two parent table primary key columns. The DDL for these tables, along with sample data inserts and index definitions, appears at the end.

The following shows a parent table insert and the corresponding locks that Oracle obtains. The lock display is provided by the "lockmon" script available at appsdba.com.

```
PARENT> INSERT INTO parent_one VALUES(16);

1 row created.

PARENT>
                                                                  Last
ORACLE        SQL         Lock         Mode       Mode         Cnvrt
User      SID Command     Type         Held       Request       Time Blocking Owner    Object
------- ----- ----------- ------------ ---------- ---------- ------- -------- -------- ---------------
DBAMON    247 No command  Table Lock   Row-S (RS) NONE          1022 No Block DBAMON   CHILD_ONE
DBAMON        No command  Table Lock   Row-S (RS) NONE          1022 No Block DBAMON   CHILD_TWO
DBAMON        No command  Table Lock   Row-X (RX) NONE          1022 No Block DBAMON   PARENT_ONE
DBAMON        No command  Row Lock     Excl (X)   NONE          1022 No Block SYS      _SYSSMU4$
```

With a simple insert into a parent table we can see that Oracle must take out "row share[1]" locks on each child table. This occurs regardless of any indexing on the child tables. This prevents another session from placing an exclusive lock on either table. If we then attempt to manipulate a row in one of the child tables (e.g. child_one):

```
CHILD> delete from child_one
  2  where child_one_id = 11;

1 row deleted.

CHILD>
                                                                  Last
```

[1] Please see the Oracle 10g Concepts manual for an explanation of the different types of locks Oracle uses.


```
ORACLE        SQL         Lock         Mode       Mode         Cnvrt
User      SID Command     Type         Held       Request       Time Blocking Owner    Object
------- ----- ----------- ------------ ---------- ---------- ------- -------- -------- ---------------
DBAMON    247 No command  Table Lock   Row-S (RS) NONE          1515 No Block DBAMON   CHILD_ONE
DBAMON        No command  Row Lock     Excl (X)   NONE          1515 No Block SYS      _SYSSMU4$
DBAMON        No command  Table Lock   Row-X (RX) NONE          1515 No Block DBAMON   PARENT_ONE
DBAMON        No command  Table Lock   Row-S (RS) NONE          1515 No Block DBAMON   CHILD_TWO

DBAMON    257 No command  Table Lock   Row-S (RS) NONE             6 No Block DBAMON   PARENT_TWO
DBAMON        No command  Row Lock     Excl (X)   NONE             6 No Block SYS      _SYSSMU9$
DBAMON        No command  Table Lock   Row-S (RS) NONE             6 No Block DBAMON   PARENT_ONE
DBAMON        No command  Table Lock   Row-X (RX) NONE             6 No Block DBAMON   CHILD_ONE
```

We see that now Oracle has taken row share locks on both parent tables as well as a row exclusive lock for the row being deleted from the child_one table. The child table delete works with no conflicts from the insert statement on the parent table. So far so good. Now, what if we delete a row in the parent table and then delete a row from a child table?

```
PARENT> delete from parent_one
  2  where parent_one_id = 16;

1 row deleted.

PARENT>
                                                          Last
ORACLE        SQL         Lock         Mode       Mode     Cnvrt
User      SID Command     Type         Held       Request   Time Blocking Owner    Object
------- ----- ----------- ------------ ---------- ---------- ------- -------- -------- ---------------
DBAMON    267 No command  Table Lock   Row-X (RX) NONE             9 No Block DBAMON   PARENT_ONE
DBAMON        No command  Row Lock     Excl (X)   NONE             9 No Block SYS      _SYSSMU4$
```

Deleting a row from the parent table does not require any locks on the child table(s). Oracle can use the integrity constraint to verify that there are no child records. To prove this we can try to delete a parent row that has children.

```
PARENT> delete from parent_one where parent_one_id = 12;
delete from parent_one where parent_one_id = 12
*
ERROR at line 1:
ORA-02292: integrity constraint (DBAMON.CHILD_ONE_FK1) violated - child record
found


PARENT>
```

In this case, the integrity constraint catches the error and no locks are taken. Now, we delete a child row while still holding a lock on the parent table for a row that was deleted.

```
CHILD> delete from child_one
  2  where child_one_id = 12;

1 row deleted.

CHILD>
                                                          Last
ORACLE        SQL         Lock         Mode       Mode     Cnvrt
User      SID Command     Type         Held       Request   Time Blocking Owner    Object
------- ----- ----------- ------------ ---------- ---------- ------- -------- -------- ---------------
DBAMON    250 No command  Table Lock   Row-S (RS) NONE            42 No Block DBAMON   PARENT_ONE
DBAMON        No command  Table Lock   Row-X (RX) NONE            42 No Block DBAMON   CHILD_ONE
DBAMON        No command  Table Lock   Row-S (RS) NONE            42 No Block DBAMON   PARENT_TWO
DBAMON        No command  Row Lock     Excl (X)   NONE             3 No Block SYS      _SYSSMU5$
```



```
DBAMON    267 No command  Table Lock   Row-X (RX) NONE            91 No Block DBAMON   PARENT_ONE
DBAMON        No command  Row Lock     Excl (X)   NONE            91 No Block SYS      _SYSSMU4$
```

Again, so far so good. Now, what happens if we delete a row from the child table and then manipulate a row in the parent table?

```
CHILD> delete from child_one
  2  where child_one_id = 11;

1 row deleted.

SQL>
                                                                  Last
ORACLE        SQL         Lock         Mode       Mode         Cnvrt
User      SID Command     Type         Held       Request       Time Blocking Owner    Object
------- ----- ----------- ------------ ---------- ---------- ------- -------- -------- ---------------
DBAMON    257 No command  Table Lock   Row-X (RX) NONE            18 No Block DBAMON   CHILD_ONE
DBAMON        No command  Table Lock   Row-S (RS) NONE            18 No Block DBAMON   PARENT_ONE
DBAMON        No command  Table Lock   Row-S (RS) NONE            18 No Block DBAMON   PARENT_TWO
DBAMON        No command  Row Lock     Excl (X)   NONE            18 No Block SYS      _SYSSMU9$


PARENT> delete from parent_one
  2  where PARENT_ONE_ID = 16;
                                                                  Last
ORACLE        SQL         Lock         Mode       Mode         Cnvrt
User      SID Command     Type         Held       Request       Time Blocking Owner    Object
------- ----- ----------- ------------ ---------- ---------- ------- -------- -------- ---------------
DBAMON    247 DELETE      Table Lock   NONE       Share (S)     10 No Block DBAMON   CHILD_ONE
DBAMON        DELETE      Table Lock   Row-X (RX) NONE          10 No Block DBAMON   PARENT_ONE

DBAMON    257 No command  Table Lock   Row-X (RX) NONE          76 Blocking DBAMON   CHILD_ONE
DBAMON        No command  Row Lock     Excl (X)   NONE          76 No Block SYS      _SYSSMU9$
DBAMON        No command  Table Lock   Row-S (RS) NONE          76 No Block DBAMON   PARENT_TWO
DBAMON        No command  Table Lock   Row-S (RS) NONE          76 No Block DBAMON   PARENT_ONE
```

We now have a blocking lock and a hung session. We know that the parent row has no children so in theory these two transactions should be able to complete. However, since there are no indexes on the child table foreign key constraint columns, Oracle is forced to take a "row exclusive" lock on the child_one table. This is the same table that we have deleted a row from, but have not committed. If we commit, or rollback, the delete from the child_one table we will see that the parent_one table delete can take place.

```
CHILD> rollback;

Rollback complete.

CHILD>
                                                                  Last
ORACLE        SQL         Lock         Mode       Mode         Cnvrt
User      SID Command     Type         Held       Request       Time Blocking Owner    Object
------- ----- ----------- ------------ ---------- ---------- ------- -------- -------- ---------------
DBAMON    247 No command  Table Lock   Row-X (RX) NONE           224 No Block DBAMON   PARENT_ONE
DBAMON        No command  Row Lock     Excl (X)   NONE            29 No Block SYS      _SYSSMU4$
```

Now we add an index to the child_one foreign key that references the parent_one table.

```
CHILD> CREATE INDEX child_one_i1 ON child_one
  2  (parent_one_id);

Index created.

CHILD> delete from child_one
  2  where child_one_id = 11;
```



```
1 row deleted.

CHILD>
                                                                  Last
ORACLE        SQL         Lock         Mode       Mode         Cnvrt
User      SID Command     Type         Held       Request       Time Blocking Owner    Object
------- ----- ----------- ------------ ---------- ---------- ------- -------- -------- ---------------
DBAMON    257 No command  Table Lock   Row-S (RS) NONE            15 No Block DBAMON   PARENT_TWO
DBAMON        No command  Table Lock   Row-X (RX) NONE            15 No Block DBAMON   CHILD_ONE
DBAMON        No command  Table Lock   Row-S (RS) NONE            15 No Block DBAMON   PARENT_ONE
DBAMON        No command  Row Lock     Excl (X)   NONE            15 No Block SYS      _SYSSMU9$


PARENT> delete from parent_one
  2  where PARENT_ONE_ID = 16;

1 row deleted.

PARENT>
                                                                  Last
ORACLE        SQL         Lock         Mode       Mode         Cnvrt
User      SID Command     Type         Held       Request       Time Blocking Owner    Object
------- ----- ----------- ------------ ---------- ---------- ------- -------- -------- ---------------
DBAMON    247 No command  Table Lock   Row-S (RS) NONE            21 No Block DBAMON   CHILD_ONE
DBAMON        No command  Row Lock     Excl (X)   NONE            21 No Block SYS      _SYSSMU4$
DBAMON        No command  Table Lock   Row-X (RX) NONE            21 No Block DBAMON   PARENT_ONE

DBAMON    257 No command  Table Lock   Row-S (RS) NONE           112 No Block DBAMON   PARENT_TWO
DBAMON        No command  Table Lock   Row-S (RS) NONE           112 No Block DBAMON   PARENT_ONE
DBAMON        No command  Row Lock     Excl (X)   NONE           112 No Block SYS      _SYSSMU9$
DBAMON        No command  Table Lock   Row-X (RX) NONE           112 No Block DBAMON   CHILD_ONE
```

Now we see that both transactions can complete. The index has allowed Oracle to avoid taking a table level lock on the child table.

Now what about a composite index?

```
CHILD> drop index child_one_i1;

Index dropped.

CHILD> CREATE INDEX child_one_i1 ON child_one
  2  (parent_one_id, parent_two_id);

Index created.

CHILD> delete from child_one
  2  where child_one_id = 11;

1 row deleted.

CHILD>
                                                                  Last
ORACLE        SQL         Lock         Mode       Mode         Cnvrt
User      SID Command     Type         Held       Request       Time Blocking Owner    Object
------- ----- ----------- ------------ ---------- ---------- ------- -------- -------- ---------------
DBAMON    257 No command  Table Lock   Row-S (RS) NONE            15 No Block DBAMON   PARENT_ONE
DBAMON        No command  Table Lock   Row-S (RS) NONE            15 No Block DBAMON   PARENT_TWO
DBAMON        No command  Row Lock     Excl (X)   NONE            15 No Block SYS      _SYSSMU9$
DBAMON        No command  Table Lock   Row-X (RX) NONE            15 No Block DBAMON   CHILD_ONE

PARENT> delete from parent_one
  2  where PARENT_ONE_ID = 16;

1 row deleted.

PARENT>
```



```
                                                          Last
ORACLE        SQL         Lock         Mode       Mode     Cnvrt
User      SID Command     Type         Held       Request   Time Blocking Owner    Object
------- ----- ----------- ------------ ---------- ---------- ------- -------- -------- ---------------
DBAMON    247 No command  Table Lock   Row-X (RX) NONE          12 No Block DBAMON   PARENT_ONE
DBAMON        No command  Row Lock     Excl (X)   NONE          12 No Block SYS      _SYSSMU4$
DBAMON        No command  Table Lock   Row-S (RS) NONE          12 No Block DBAMON   CHILD_ONE

DBAMON    257 No command  Table Lock   Row-S (RS) NONE          49 No Block DBAMON   PARENT_ONE
DBAMON        No command  Table Lock   Row-S (RS) NONE          49 No Block DBAMON   PARENT_TWO
DBAMON        No command  Row Lock     Excl (X)   NONE          49 No Block SYS      _SYSSMU9$
DBAMON        No command  Table Lock   Row-X (RX) NONE          49 No Block DBAMON   CHILD_ONE
```

Works great! But what if we had created the index with the keys reversed so that the parent_one_id was not the leading edge of the index?

```
CHILD> drop index child_one_i1;

Index dropped.

CHILD> CREATE INDEX child_one_i1 ON child_one
  2  (parent_two_id, parent_one_id)
  3  NOLOGGING
  4  /

Index created.

CHILD> delete from child_one
  2  where child_one_id = 11;

1 row deleted.

CHILD>
                                                          Last
ORACLE        SQL         Lock         Mode       Mode     Cnvrt
User      SID Command     Type         Held       Request   Time Blocking Owner    Object
------- ----- ----------- ------------ ---------- ---------- ------- -------- -------- ---------------
DBAMON    257 No command  Table Lock   Row-S (RS) NONE          12 No Block DBAMON   PARENT_TWO
DBAMON        No command  Table Lock   Row-X (RX) NONE          12 No Block DBAMON   CHILD_ONE
DBAMON        No command  Table Lock   Row-S (RS) NONE          12 No Block DBAMON   PARENT_ONE
DBAMON        No command  Row Lock     Excl (X)   NONE          12 No Block SYS      _SYSSMU9$

PARENT> delete from parent_one
  2  where PARENT_ONE_ID = 16;
                                                          Last
ORACLE        SQL         Lock         Mode       Mode     Cnvrt
User      SID Command     Type         Held       Request   Time Blocking Owner    Object
------- ----- ----------- ------------ ---------- ---------- ------- -------- -------- ---------------
DBAMON    247 DELETE      Table Lock   NONE       Share (S)    18 No Block DBAMON   CHILD_ONE
DBAMON        DELETE      Table Lock   Row-X (RX) NONE          18 No Block DBAMON   PARENT_ONE

DBAMON    257 No command  Table Lock   Row-S (RS) NONE          61 No Block DBAMON   PARENT_TWO
DBAMON        No command  Table Lock   Row-S (RS) NONE          61 No Block DBAMON   PARENT_ONE
DBAMON        No command  Row Lock     Excl (X)   NONE          61 No Block SYS      _SYSSMU9$
DBAMON        No command  Table Lock   Row-X (RX) NONE          61 Blocking DBAMON   CHILD_ONE
```

Woops! We're in the same boat as if we didn't have an index. So, the foreign key must be the leading edge of an index if a composite index is used. If secondary columns are foreign keys as well, then make a standalone index for the secondary column(s) of your composite index if those columns are defined as foreign keys.

```
CHILD> CREATE INDEX child_one_i2 ON child_one
  2  (parent_one_id);
```



```
Index created.

CHILD> delete from child_one
  2  where child_one_id = 11;

1 row deleted.

CHILD>
                                                          Last
ORACLE        SQL         Lock         Mode       Mode    Cnvrt
User      SID Command     Type         Held       Request  Time Blocking Owner    Object
------- ----- ----------- ------------ ---------- ---------- ------- -------- -------- ---------------
DBAMON    257 No command  Table Lock   Row-S (RS) NONE           12 No Block DBAMON   PARENT_ONE
DBAMON        No command  Table Lock   Row-S (RS) NONE           12 No Block DBAMON   PARENT_TWO
DBAMON        No command  Row Lock     Excl (X)   NONE           12 No Block SYS      _SYSSMU9$
DBAMON        No command  Table Lock   Row-X (RX) NONE           12 No Block DBAMON   CHILD_ONE

PARENT> delete from parent_one
  2  where PARENT_ONE_ID = 16;

1 row deleted.

PARENT>
                                                          Last
ORACLE        SQL         Lock         Mode       Mode    Cnvrt
User      SID Command     Type         Held       Request  Time Blocking Owner    Object
------- ----- ----------- ------------ ---------- ---------- ------- -------- -------- ---------------
DBAMON    247 No command  Table Lock   Row-X (RX) NONE           10 No Block DBAMON   PARENT_ONE
DBAMON        No command  Row Lock     Excl (X)   NONE           10 No Block SYS      _SYSSMU4$
DBAMON        No command  Table Lock   Row-S (RS) NONE           10 No Block DBAMON   CHILD_ONE

DBAMON    257 No command  Table Lock   Row-S (RS) NONE           52 No Block DBAMON   PARENT_ONE
DBAMON        No command  Table Lock   Row-S (RS) NONE           52 No Block DBAMON   PARENT_TWO
DBAMON        No command  Row Lock     Excl (X)   NONE           52 No Block SYS      _SYSSMU9$
DBAMON        No command  Table Lock   Row-X (RX) NONE           52 No Block DBAMON   CHILD_ONE
```

Now we have one index with both columns and another index with just the second key, and no problems. Even though we need both indexes there may be cases where the composite index satisfies enough queries to justify having both columns in one of the indexes.


```
CREATE TABLE parent_one
( parent_one_id      NUMBER
    CONSTRAINT parent_one_nn1 NOT NULL )
/

ALTER TABLE parent_one ADD CONSTRAINT parent_one_pk
 PRIMARY KEY (parent_one_id)
 USING INDEX
/

INSERT INTO parent_one VALUES(11);
INSERT INTO parent_one VALUES(12);
INSERT INTO parent_one VALUES(13);
INSERT INTO parent_one VALUES(14);
--INSERT INTO parent_one VALUES(15);

COMMIT;

CREATE TABLE parent_two
( parent_two_id      NUMBER
    CONSTRAINT parent_two_nn1 NOT NULL )
/

ALTER TABLE parent_two ADD CONSTRAINT parent_two_pk
 PRIMARY KEY (parent_two_id)
 USING INDEX
/

INSERT INTO parent_two VALUES(21);
INSERT INTO parent_two VALUES(22);
INSERT INTO parent_two VALUES(23);
INSERT INTO parent_two VALUES(24);
INSERT INTO parent_two VALUES(25);

COMMIT;

CREATE TABLE child_one
( child_one_id      NUMBER
    CONSTRAINT child_one_nn1 NOT NULL,
  parent_one_id      NUMBER
    CONSTRAINT child_one_nn2 NOT NULL,
  parent_two_id      NUMBER
    CONSTRAINT child_one_nn3 NOT NULL )
/

ALTER TABLE child_one ADD CONSTRAINT child_one_pk
 PRIMARY KEY (child_one_id)
 USING INDEX
/

ALTER TABLE child_one ADD (
  CONSTRAINT child_one_fk1 FOREIGN KEY (parent_one_id)
    REFERENCES parent_one (parent_one_id))
/

ALTER TABLE child_one ADD (
  CONSTRAINT child_one_fk2 FOREIGN KEY (parent_two_id)
    REFERENCES parent_two (parent_two_id))
/

CREATE INDEX child_one_i1 ON child_one
(parent_two_id, parent_one_id)
/

-- CREATE INDEX child_one_i1 ON child_one
-- (parent_two_id)
-- /

CREATE INDEX child_one_i2 ON child_one
(parent_one_id)
/
```



```
INSERT INTO child_one VALUES(11,11,21);
INSERT INTO child_one VALUES(12,12,22);
INSERT INTO child_one VALUES(13,13,23);
INSERT INTO child_one VALUES(14,14,24);
--INSERT INTO child_one VALUES(15,15,25);

COMMIT;


CREATE TABLE child_two
( child_two_id      NUMBER
    CONSTRAINT child_two_nn1 NOT NULL,
  parent_one_id      NUMBER
    CONSTRAINT child_two_nn2 NOT NULL,
  parent_two_id      NUMBER
    CONSTRAINT child_two_nn3 NOT NULL )
/

ALTER TABLE child_two ADD CONSTRAINT child_two_pk
 PRIMARY KEY (child_two_id)
 USING INDEX
/

ALTER TABLE child_two ADD (
  CONSTRAINT child_two_fk1 FOREIGN KEY (parent_one_id)
    REFERENCES parent_one (parent_one_id))
/

ALTER TABLE child_two ADD (
  CONSTRAINT child_two_fk2 FOREIGN KEY (parent_two_id)
    REFERENCES parent_two (parent_two_id))
/

--CREATE INDEX child_two_i1 ON child_two
--(parent_one_id, parent_two_id)
--/

CREATE INDEX child_two_i1 ON child_two
(parent_one_id)
/

CREATE INDEX child_two_i2 ON child_two
(parent_two_id)
/

INSERT INTO child_two VALUES(11,11,21);
INSERT INTO child_two VALUES(12,12,22);
INSERT INTO child_two VALUES(13,13,23);
INSERT INTO child_two VALUES(14,14,24);
--INSERT INTO child_two VALUES(15,15,25);

COMMIT;
```

