Transactions and Scoping: Reining in the Octopus

Scott M. Dulecki

BravePoint

BRAVEPOINT

Agenda

- **Setting the stage**
- Transactions
- Blocking
- Scope
- Gotchas
- Questions
- Wrap-Up

Setting The Stage: Me

- Scott M. Dulecki
- Presenter at Explore! and regional user groups
- Board Member, Midwest MFG/PRO Users Group
- President, West Michigan Progress Users Group
- Past President, Michigan Progress Users Group
- PEG member 1998061901
- Author of:
  - ***Safe Haven: Archiving in MFG/PRO***

Setting The Stage: Us

- BravePoint Inc. (www.BravePoint.com)
- 100+ Employees (30+ MFG/PRO Consultants)
- Progress Service Provider
- QAD Alliance and Channel Sales Partner
- Three of us have used Progress since 1984
- Supporting Progress and its customers since 1987

Setting The Stage: You

- Developers? DBAs? Managers?
- User Interface?
- Progress Version?
- Oracle?

Agenda

- Setting the stage
- **Transactions**
- Blocking
- Scope
- Gotchas
- Questions
- Wrap-Up

Transactions

- A transaction is a unit of work, to be completed or backed out in its entirety
- It controls when changes are **committed**, not when they are **written**
- Only one transaction is open at a time
- It sounds so easy…

Dirty Reads

- Data is written before the transaction is complete
- Someone else can read that data and use it
- If I back out of the original transaction…

- Does not apply to Oracle Data Servers

Considerations 1

- A background job is giving all technicians a 7% pay increase. Midway through, power fails.
- How much work should be backed out?
- Probably everything.

Considerations 2

- I’m assigning trouble tickets. I give Laura the fifth one, and then decide to give it to Doug.
- How much work should be backed out?
- Probably just this ticket, and not the first four.

Considerations 3

- Laura is adding a group of people from Keystone to the NCPUG Halloween party list. As she’s adding Brian White, she realizes that it’s time for the NCPUG board meeting and escapes out.
- How much work should be backed out?

- Probably just this record.

Backing Out A Transaction

- System crashes or fails
- User presses STOP key
- Explicit UNDO statement

Ending A Transaction

- Reach the end of the transaction block

Agenda

- Setting the stage
- Transactions
- **Blocking**
- Scope
- Gotchas
- Questions
- Wrap-Up

Blocking and Tackling

- Transactions occur within blocks
- Not all blocks have transactions
  - Some are native, some are forced
- Transactions occur on the outermost block

Transaction Levels

```
Blk1:
FOR EACH order:
  UPDATE order.

  Blk2:
  FOR EACH orderline OF order:
    UPDATE orderline.
  END.
END.
```

Default Transaction Blocks

- Procedure
- FOR
- REPEAT

- But no transaction unless there’s an update…

Forced Transaction Blocks

- DO TRANSACTION
- FOR EACH TRANSACTION
- REPEAT TRANSACTION
- PROCEDURE, FOR, or REPEAT w/ EXCLUSIVE locks
- DO ON ENDKEY
- DO ONE ERROR

Multiple Blocks 1

```
ItemBlock:
REPEAT:
  PROMPT-FOR item.itemnum.
  FIND item USING item.itemnum NO-ERROR.
  UPDATE item EXCEPT item.itemnum.
END.

SrepBlock:
REPEAT:
  PROMPT-FOR salesrep.salesrep.
  FIND salesrep USING salesrep.salesrep NO-ERROR.
  UPDATE salesrep EXCEPT salesrep.salesrep.
END.
```

Multiple Blocks 2

```
OrderBlock:
REPEAT:
  INSERT order.

  OlineBlock:
  REPEAT:
    CREATE orderline.
    ASSIGN orderline.ordernum = order.ordernum.
    UPDATE orderline EXCEPT order.ordernum.
  END.
END.
```

Making Transactions Smaller

```
FOR EACH order:
  DO TRANSACTION:
    UPDATE order WITH 1 COL TITLE “Order”.
  END.

  DO TRANSACTION:
    FOR EACH orderline OF order:
      UPDATE orderline WITH 1 COL TITLE “Orderlines of Order”.
    END.
  END.

END.
```

Making Transactions Larger

```
DO TRANSACTION:
 FOR EACH order:
  UPDATE order.
  FOR EACH orderline OF order:
   UPDATE orderline.
  END.
 END.
END.
```

Thinking Ahead…

```
FOR EACH order:
  UPDATE order WITH 1 COL TITLE “Order”.

  FOR EACH orderline OF order:
    DO TRANSACTION:
      UPDATE orderline WITH 1 COL TITLE “Orderlines of Order”.
    END.
  END.
END.
```

Hey! A Transaction's Already Active!

- So start a subtransaction
- The same rules apply
- Changes stored in the .lbi file
  - Database and variable activity
  - NO-UNDO is your friend… ☺

So… Is A Transaction Active?

- TRANSACTION function

```
IF TRANSACTION THEN active = TRUE
ELSE active = FALSE.
```

Agenda

- Setting the stage
- Transactions
- Blocking
- **Scope**
- Gotchas
- Questions
- Wrap-Up

What Is Scoping?

- Applies to frames, transactions, records, and record locks
- Today, locks, records, and transactions only

Fine. So What Is It?

- Scoping is another way of saying Availability
- Where is this FLRT available?

Record Locks

- Three types: no-lock, shared, exclusive
- No-lock: last momentarily (to ensure read)
- Share-lock: lasts until…
  - Upgraded
  - Canceled
  - Ended
- Exclusive
  - Lasts until EOTrx
  - Downgraded to share-lock until out of scope…

Special Lock Scenarios

- Deadly embrace
  - We both share-lock the same record
  - Neither one can commit until the other lets go
- Mexican standoff
  - We each share lock fifty records
  - I want to exclusive-lock one of yours
  - You want to exclusive-lock one of mine
  - But they’re not the same record
  - Neither one can commit until the other lets go

Record Scoping

- When is a record available?
- It says when to…
  - …write records to the DB
  - …release certain record locks
  - …invoke index and field validations
  - …reinitialize position in the index

Types of Scope

- Weak scope
  - FOR EACH, REPEAT
  - Scope can be raised by reference outside block
- Procedure scope
- Strong scope
  - DO FOR, REPEAT FOR
  - Scope cannot be raised outside block

Weak Scoping 1

```
FOR EACH customer WHERE name begins “U”:
  DISPLAY custnum name.
END.
```

Weak Scoping 2

```
FOR EACH customer WHERE name begins “U”:
  DISPLAY custnum name.
END.

IF AVAILABLE customer THEN
  DISPLAY customer.custnum customer.name.
```

Procedure Scoping

```
DO TRANSACTION:
  PROMPT-FOR custnum.
  FIND customer USING custnum EXCLUSIVE-LOCK.
  UPDATE customer.
END.
```

Strong Scoping

```
DO FOR customer:
  PROMPT-FOR custnum.
  FIND customer USING custnum EXCLUSIVE-LOCK.
  UPDATE customer.
END.
```

Strong Scoping – Bzzzt!

```
DO FOR customer:
  PROMPT-FOR custnum.
  FIND customer USING custnum EXCLUSIVE-LOCK.
  UPDATE customer.
END.

FIND FIRST customer.
DISPLAY customer.
```

Agenda

- Setting the stage
- Transactions
- Blocking
- Scope
- **Gotchas**
- Questions
- Wrap-Up

Scope Creep

- Inadvertently raising record scope
  - Use a RELEASE at the end of the transaction
  - Does not release locks, but scope
  - Reread the record outside the transaction

Subtransaction Creep

- When did that transaction start, anyway?
- Called programs may be subtransactions
  - Explicitly define scope where possible
  - Check for it everywhere

Double-Buffering

- If you use buffers and don’t clarify which one, the one in scope gets used

```
DEFINE BUFFER b-cust LIKE customer.
FIND FIRST customer.
FIND LAST b-cust.
FIND FIRST order WHERE order.custnum = custnum. ←
```

Where Are My Transactions?

- COMPILE LISTING

```
COMPILE progname LISTING prognam.lis.
```

Sample Output 1

```
{} Line Blk
-- ---- ---
      1     OUTER-BLOCK:
      2   1 FOR EACH order:
      3   1   UPDATE order.
      4   1     inner-block:
      5   2     REPEAT :
      6   2         CREATE orderline.
      7   2         SET orderline.ordernum = order.ordernum.
      8   2         UPDATE orderline EXCEPT orderline.ordernum.
      9   1     END.
     10     END.
```

Sample Output 2

```
     File Name       Line Blk. Type Tran Blk. Label
-------------------- ---- --------- ---- --------------------
.\test1.p               0 Procedure No
.\test1.p               2 For       Yes  OUTER-BLOCK
    Buffers: sports2000.Order
    Frames:  Unnamed

.\test1.p               5 Repeat    Yes  inner-block
    Buffers: sports2000.OrderLine
    Frames:  Unnamed
```

Agenda

- Setting the stage
- Transactions
- Blocking
- Scope
- Gotchas
- **Questions**
- Wrap-Up

Questions?

Agenda

- Setting the stage
- Transactions
- Blocking
- Scope
- Gotchas
- Questions
- **Wrap-Up**

Wrap-Up

- Transactions are your friends
- They can spread their tentacles everywhere
- COMPILE LISTING, your best friend, will help you control them

For Further Information

- **Business card with “Scoping”**
- **My business cards up front**

Scott M. Dulecki

BravePoint

616/957-3184

sdulecki@bravepoint.com