READ Performance for Developers

Session 740

Alan Bartholomew
Texas Instruments

CAUTION

- This presentation contains “typical” good practices
- There are ***exceptions*** to these guidelines
- Optimizing READ performance involves many factors outside the scope of this presentation
  - For example, index selection and transparent denormalization

Objectives

- Be reasonably efficient all the time
- Be ***exceptionally*** efficient 10 - 20% of the time
- Balance efficiency against business benefit

Outline

⇨
- Minimize execution of READ statements
  - Optimize the READ statement
  - Minimize READ locking

Minimize Execution of READs

- Save data in memory
- Perform re-sort/re-filter on client
- Use SOME ... THAT construct
- Take advantage of JOINs
- Use persistent views

Online: Save Data in Memory

- Read data in one action block and pass it to another action block
- Read data in one procedure and pass it via dialog flow client-to-client or screen-to-screen
- Read data during one execution and save it in PrAD import/export views
- Read data in one action block and save in uninitialized local views

Online: Save Data in Memory

- Non-optimized example:
  - Every execution of the transaction reads security_access

```
READ security_access
  WHERE DESIRED security_access user_id IS EQUAL
  TO USERID
WHEN SUCCESSFUL
  disable logic ...
```


Online: Save Data in Memory

- Optimized example:
  - Import view mapped to export view
  - Only initial execution of transaction reads security_access

```
IF import security_access authorization_level IS EQUAL TO
SPACES

  READ security_access
    WHERE DESIRED security_access user_id IS EQUAL
    TO USERID
  WHEN SUCCESSFUL
    MOVE security_access TO export security_access
    disable logic ...
```


Batch: Save Data in Memory

- Read data in one action block and pass it to another action block
- Read data in one action block and save in uninitialized local views

Batch: Save Data in Memory

- Non-optimized example:
  - Every execution of the action blocks reads shoe-size code

```
READ shoe_size
  WHERE DESIRED shoe_size length IS EQUAL TO
        import shoe_size length
WHEN SUCCESSFUL
...
WHEN NOT FOUND
  EXIT STATE IS invalid_shoe_size
```


Batch: Save Data in Memory

- Optimized example:
  - View property must be set to NOT initialize on every entry
  - Handling group view overflow:
    - »Stop processing

      -- OR --

    - »Keep processing but execute READ validation if no match is found in group view
      - ∗Write warning message to a log file
  - For large group views, consider binary search

Batch: Save Data in Memory

- Optimized example:

```
IF local_group_uninit IS EMPTY
   READ EACH shoe_size
       TARGETING local_group_uninit FROM THE
       BEGINNING UNTIL FULL
       MOVE shoe_size TO local_single shoe_size

SET local ief_supplied flag TO “N”
FOR EACH local_group_uninit
   IF import shoe_size length IS EQUAL TO
   local_single shoe_size length
     SET local ief_supplied flag TO “Y”

IF local ief_supplied flag IS EQUAL TO “N”
   EXIT STATE IS invalid_shoe_size
```


Perform Re-sorting/ Re-filtering on Client

- Where possible, have DBMS perform sort/filter for initial READ EACH
  - More maintainable than a client sort/filter
- If the group view can hold all rows that meet selection criteria, then perform re-sort/re-filter in the client procedure
- Data currency
  - Data on database may change while you are re-sorting on client

When Not to Re-sort on Client

- What's wrong with this example?

Customer Table

State	Number
AZ	777
AZ	999
AZ	555
CO	333

Initial Sort
State ASC
Group View (3)

State	Number
AZ	777
AZ	999
AZ	555

Client Re-Sort
Number ASC
Group View (3)

State	Number
AZ	555
AZ	777
AZ	999

- The re-sort should put CO 333 as the first occurrence

Use SOME ... THAT Construct

- Non-optimized example
  - This is non-optimized assuming that there is no need to read ORDER

```
READ order
  WHERE DESIRED order number IS EQUAL TO import
        order number

READ order_line
  WHERE DESIRED order_line number IS EQUAL TO
        import order_line number
    AND DESIRED order_line is_part_of CURRENT
        order
```

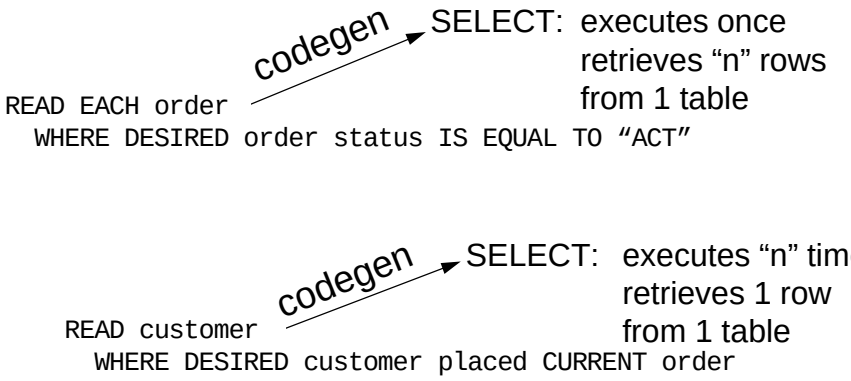

Use SOME ... THAT Construct

- Optimized example:

```
READ order_line
  WHERE DESIRED order_line number IS EQUAL TO
        import order_line number
    AND DESIRED order_line is_part_of SOME order
    AND THAT order number IS EQUAL TO import order
        number
```

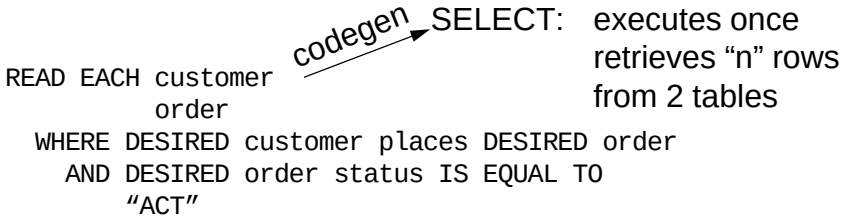

Take Advantage of JOINs

- Non-optimized example:



Take Advantage of JOINs

- Optimized example:



Use Persistent Views

- Composer requires that a READ occur in the same action diagram before certain actions
  - Update, Delete, Associate, Disassociate, Transfer
- If you have already READ in a calling action diagram and don't want to READ again, then use persistent views
- Review the need carefully before using persistent views

Passing Currency

- Non-optimized example:

```
Action Block A
READ x
USE b

Action Block B
READ x        <-------------  redundant
UPDATE x
CREATE y
  ASSOCIATE WITH x
```


Passing Currency

- Optimized example:

```
Action Block A
READ x
USE b

Action Block B
UPDATE persistent x
CREATE y
  ASSOCIATE WITH persistent x
```


Passing Currency

- Can only do an update if currency is passed “down”
- Illegal example:

```
Action Block A
USE b
UPDATE x   <-------- won't work

Action Block B
READ x
```


Outline

- Minimize execution of READ statements
- ⇨ Optimize the READ statement
- Minimize READ locking

Optimize the READ Statement

- Starve views
- Qualify correctly
- Minimize sorting
- Compare appropriately
- Sequence qualifiers (DBMS-specific)
- OR

Starve the Entity Types in the READ List

- READ List should include only:
  - What you need back from the database to display or manipulate
  - What you need to sort on
  - What you need to have entity types in READ List related to each other
    - » Example: if you need customer and order line only, go ahead and put order in the READ List.
- It is OK for an entity action view to be referenced in the WHERE clause only

Entity Action View Population

- Optimized example:

```
Entity Action
  order_line
    number      <--- populated
    price       <--- populated
  order
    number      <--- NOT populated!

READ order_line
  WHERE DESIRED order_line number IS EQUAL TO
        import order_line number
    AND DESIRED order_line is_part_of SOME order
    AND THAT order number IS EQUAL TO import order
        number
```


Starve the Attributes in Entity Action Views

- Non-optimized example (will cause join):

```
Entity Action Views
  customer
    number
    name
  order
    number
    date_placed

READ EACH customer
          order
  WHERE DESIRED customer places DESIRED order
    AND DESIRED order date_placed IS EQUAL TO
        CURRENT DATE
```


Variable Qualifying

- Commonly required for browsers or lists
- User can enter some or all of many different selection criteria
  - For example, five selection criteria yield 120 combinations!
- Minimize the number of combinations where possible
- Write specific optimized READs for common combinations
- Write “generic” unoptimized READ for uncommon combinations

Variable Qualifying Example

```
IF import customer state IS EQUAL TO SPACES
  SET local_low customer state TO ‘  ‘
  SET local_high customer state TO ‘ZZ’
ELSE
  SET local_low customer state TO import customer
  state
  SET local_high customer state TO import customer
  state
IF import customer name IS EQUAL TO SPACES
  SET local_low customer name TO ‘  ‘
  SET local_high customer name TO ‘ZZ’
ELSE
  SET local_low customer name TO import customer
  name
  SET local_high customer name TO import customer
  name
```


Variable Qualifying Example

- This READ handles four different combinations of selection criteria
  - (e.g., name only, state only, name & state, neither)

```
READ EACH customer
  WHERE DESIRED customer state IS LESS THAN OR
        EQUAL TO local_high customer state
    AND DESIRED customer state IS GREATER THAN OR
        EQUAL TO local_low customer state
    AND DESIRED customer name IS LESS THAN OR
        EQUAL TO local_high customer name
    AND DESIRED customer name IS GREATER THAN OR
        EQUAL TO local_low customer name
```


Minimize Sorting

- Sort only when necessary
- Sort only those columns that must be sorted
- Try to sort on column(s) that are part of an index
  - If you must sort but don't care what you are sorting on, choose identifying attributes
- Try to sort in the same order as the index columns are in
- Refer to earlier slide regarding sorting in client rather than server

Sorting Example 1

```
READ EACH customer
          order
  SORTED BY customer number ASCENDING
  SORTED BY order number ASCENDING
  WHERE DESIRED customer places DESIRED order
```


Sorting Example 1–Next

- Required when there are more rows on the database than can be saved in memory

```
READ EACH customer
          order
  SORTED BY customer number ASCENDING
  SORTED BY order number ASCENDING
  WHERE DESIRED customer places DESIRED order
    AND ((DESIRED customer number IS GREATER THAN
        import_last customer number)
     OR (DESIRED customer number IS EQUAL TO
        import_last customer number
    AND DESIRED order number IS GREATER THAN
        import_last order number))
```


Sorting Example 2

- Sorting on non-identifying attributes
- OK -- if the group view can hold all rows that meet selection criteria

```
READ EACH customer
  SORTED BY customer state ASCENDING
```


Sorting Example 2–Next

- Why won't this work?

```
READ EACH customer
  SORTED BY customer state ASCENDING
  WHERE DESIRED customer state IS GREATER THAN
        import_last customer state
```

Customer Table

State	Number
AZ	777
AZ	999
AZ	555
AZ	888
AZ	222
CO	333

Group View Max = 3

Sorting Example 2–Next

Customer Table

State	Number
AZ	777
AZ	999
AZ	555
AZ	888
AZ	222
CO	333

Initial Group View

State	Number
AZ	777
AZ	999
AZ	555

Next Group View

State	Number
CO	333

Sorting Example 2–Correct

Initial READ

```
READ EACH customer
  SORTED BY customer state ASCENDING
  SORTED BY customer number ASCENDING
```

Next READ

```
READ EACH customer
  SORTED BY customer state ASCENDING
  SORTED BY customer number ASCENDING
  WHERE (DESIRED customer state IS GREATER THAN
        import_last customer state)
     OR (DESIRED customer state IS EQUAL TO
        import_last customer state
    AND DESIRED customer number IS GREATER THAN
        import_last customer number)
```


Comparisons with LIKE

- May be inefficient if:
  - It causes comparisons against a large number of rows
  - No other qualifiers are used to narrow the search
- Review with DBA before changing your READ

Comparisons with LIKE

- Non-optimized example:
  - User wants to find names that include the letters “JO”
  - DBMS must scan all rows and do comparison in order to determine a match

```
READ EACH customer
  WHERE DESIRED customer name IS LIKE ‘%JO%’
```


Comparisons with LIKE

- Optimized example:

```
READ EACH customer
  WHERE DESIRED customer postal_code IS EQUAL TO 98765
    AND DESIRED customer name IS LIKE ‘%JO%’
```


Sequence Qualifiers

- Sequence of qualifiers may impact performance
  - For DB2, sequence does NOT matter
  - In general, put qualifiers first that help the DBMS narrow the number of rows to be searched
- Work with DBA to develop guidelines for your DBMS

OR and Attribute Qualifiers

- Composer DOES optimize when there is an OR separating attribute qualifiers
  - OR optimization capability is part of 5.2 Mainframe 9403C and Workstation 5.2.9312A and all subsequent releases

```
READ EACH customer
  WHERE (DESIRED customer state IS EQUAL TO 'DE'
     OR DESIRED customer state = 'NJ')
    AND DESIRED customer is_supported_by CURRENT
        customer_representative
```


OR and Relationship Qualifier

- Composer does NOT optimize when there is an OR next to a relationship qualifier
- Review with DBA before changing your READ

```
READ EACH customer
  WHERE DESIRED customer state IS EQUAL TO
        import customer state
     OR DESIRED customer is_supported_by CURRENT
        customer_representative
```


Outline

- Minimize execution of READ statements
- Optimize the READ statement
- ⇨ Minimize READ locking

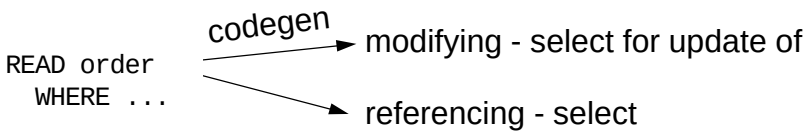
Minimize READ Locking

- Choose referencing option for a relationship
- Separate READs for conditional updates

Choose Referencing for Relationship

- Property of the “many” side of the relationship
- ***Only*** impact of this property is the type of lock placed on a READ for:
  - ASSOCIATE
  - DISASSOCIATE
  - TRANSFER

Choose Referencing for Relationship




```
READ order
  WHERE ...

CREATE order_line
  SET ...
  ASSOCIATE WITH order THAT contains IT
```


Separate READs for Conditional Update

- If an update is performed on an entity action view, then all READs against that view will be “FOR UPDATE OF”
- If the update is rarely performed or it is an extended READ then:
  - Create one entity action view for the READ
  - Create a second entity action view for the update
- Review with DBA before adding the extra view

Separate READs for Conditional Update

- Non-optimized example:

```
READ EACH customer
          order

  MOVE order TO export order

  IF order date_filled IS LESS THAN CURRENT DATE

  UPDATE order
      SET ...
```

codegen →

```
select customer
    order

select customer for update of
select order for update of
```


Separate READs for Conditional Update

- Optimized example:

```
READ EACH customer
          order

  MOVE order TO export order

  IF order date_filled IS LESS THAN CURRENT DATE

    READ existing order

    UPDATE existing order
      SET ...
```

codegen →

```
select customer
    order
```

codegen →

```
select order for update of
```


READ Performance Summary

- Minimize execution of READ statements
- Optimize the READ statement
- Minimize READ locking

READ Performance for Developers

Session 740

Alan Bartholomew
Texas Instruments