The Progress Index - Not only how they work but why and when

Michael Lonski

President - Allegro

AllegroConsultants.com

Who I Am…

- Started working with Progress® v3 (1986)
- Founded Allegro in 1993
- Internationally recognized speaker
- Author of “Coding Smart” book on ADM2
- Punster and Frequent Talker
- Ask me about Capoferro or Giganti

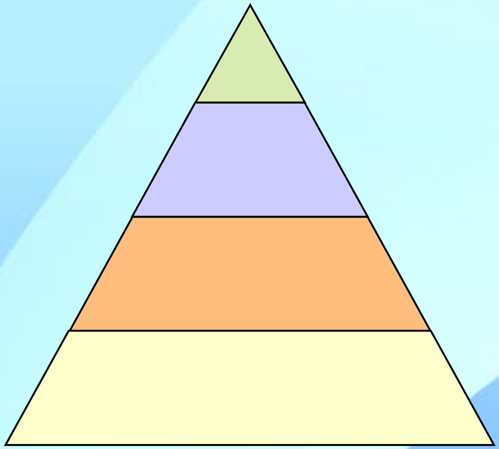
…And Why I Am Here

- Overview index rules
- Understand what survives
- Don’t get too clever
- Avoid the unknown
- When to do and not to do

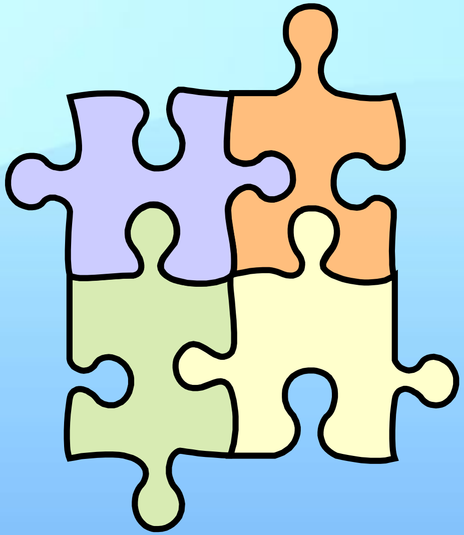
What’s Next…

- Laying the ground work

Index Rules Engines



VS



“Rules based”

“Context based”

Don’t Worry About Details

- At least, that’s what Progress said in a whitepaper on triggers and indexes

The Compiler constructs a logical tree from a query and evaluates both sides of each AND or OR, looking for index criteria. ABL counts equality, range, and sort matches (for OR) and uses them to select and bracket indexes. The precise rules are numerous and complex, and it is not important to fully understand their details.

“ABL Triggers and Indexes” - published for OE10 in 2011

Database Components

- Index Cursor
  - Maintained on behalf of client
  - Maintains position within an index
  - Can have multiple open at same time
  - One curser per record buffer per bracket
- Index Bracket
  - Set of consecutive entries in an index
  - Equality and range brackets

Bracketing

LastName	FirstName
Baker	Anita
Baker	Daniel
Baker	Thomas
Drew	Nancy
Gaston	Daniel
Gaston	Sherri
Marcus	Anita
Smith	Betty
Smith	Bob
Smith	Nancy

LastName LT “H”

LastName EQ “Smith” AND

FirstName BEGINS “B”

More Bracketing

LastName LT “H” AND

FirstName EQ “Daniel”

FirstName BEGINS “B”

LastName	FirstName
Baker	Anita
Baker	Daniel
Baker	Thomas
Drew	Nancy
Gaston	Daniel
Gaston	Sherri
Marcus	Anita
Smith	Betty
Smith	Bob
Smith	Nancy

3 Types of ABL Queries

- FIND
  - V6 and earlier
  - FIRST/LAST/NEXT/PREV/CURRENT/CAN-FIND
  - Doesn’t support multiple-index selection
- FOR
  - V6 and earlier
  - EACH/FIRST/LAST
- GET
  - V7 and later
  - Works with QUERY
  - Expected to replace FIND usage

What’s Next…

- ❖Laying the ground work
- ❖Understanding the rules

Thinning the herd

Expecting the right results

Thinning the Herd

- Every index starts as a candidate
- "Tokens" in WHERE evaluated
- Index candidates are removed as rules are applied
- Elimination rather than selection
- Think "last one standing" instead of "first one chosen"

Hierarchy For A Single Index

1. If “CONTAINS”, use word-index
   - Sometimes put after #4
2. Unique index with all equality matches
   - # of index fields doesn’t matter
3. Most active equality matches
   - Full matches trump partial matches
4. Most active range matches
5. Most active sort matches
6. The primary index
7. First index alphabetically by name
   - Temp-tables go by order of definition

Which Index Survived?

Multiple Index Usage

- If indexes are available for both sides of ***WHERE***... ***AND/OR***, more than one index can be used
- Multiple indexes will ***only**** be used to assist in bracketing* records
- *Can still be a bracket of 1 record
- Return order ***not**** guaranteed

WHERE…AND…

- ***WHERE*** clause includes the use of ***AND***
- All components of each index are involved in equality matches
- No unique index candidates are left

WHERE…OR…

- ***WHERE*** clause includes the use of ***OR***
- Both the left and right side of the OR contain at least the lead component of an index
- These lead components are involved in either equality or range matches

What’s Next…

- Laying the ground work
- Understanding the rules
- How it can all go wrong

Ignoring The Engine

USE-INDEX …

… TABLE-SCAN

FIND … WHERE ROWID ( ) EQ …

Ignoring The Rules

```
WHERE NOT …

                WHERE <non-indexed field> EQ …

 WHERE … MATCHES …
                      WHERE IF …

                          THEN … EQ vcInput

                          ELSE TRUE

WHERE SUBSTRING (<database field>) EQ “A”
```

Cleverness Kills

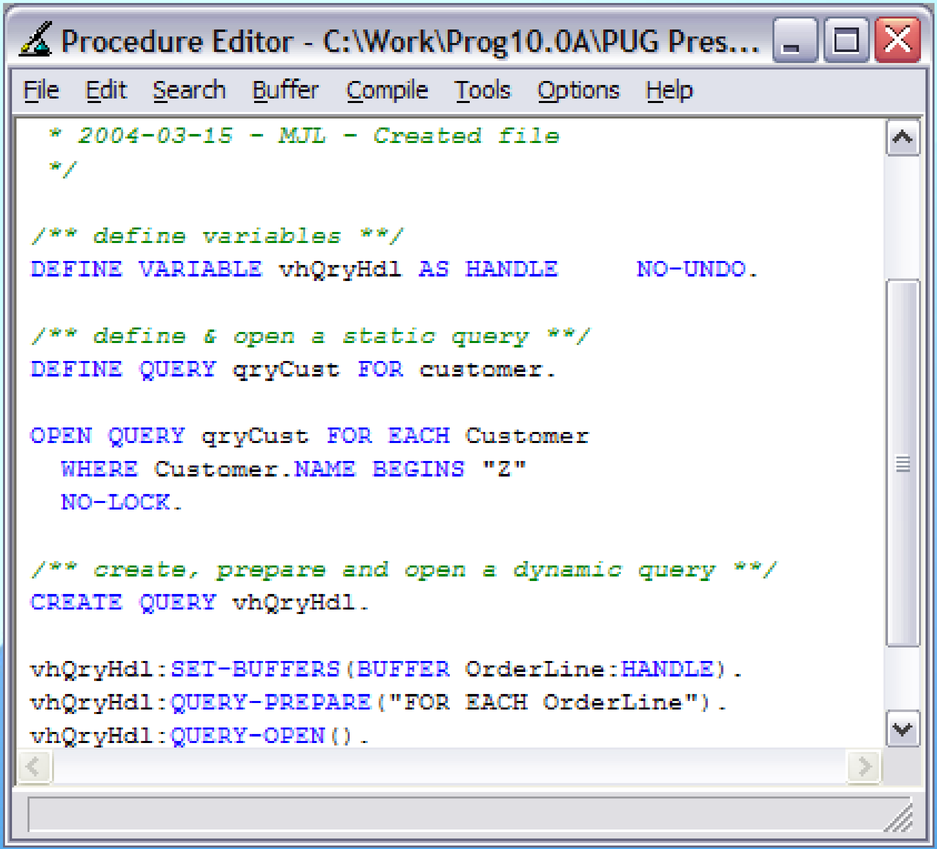
```
FOR EACH table WHERE unique-field EQ ?
  BY non-unique-field:
```

- ❖Equality queries on unique indices ignore sorting (since only 1 record should be returned)
- ❖Use “unique-field GE ?” to change to range match

Breaking Data

- Ability to have multiple records with “?” unknown value in unique indices is a side effect
- Sorts differently when in an index field vs. non-index field
- Use only EQ and NE in comparisons or face frustration
- See KB 15969 and P4130 for more details

Demo – UniqueProblems.p



Procedure Editor - C:\Work\Prog10.0A\PUG Pres...
File Edit Search Buffer Compile Tools Options Help

```
 * 2004-03-15 - MJL - Created file
 */

/** define variables **/
DEFINE VARIABLE vhQryHdl AS HANDLE     NO-UNDO.

/** define & open a static query **/
DEFINE QUERY qryCust FOR customer.

OPEN QUERY qryCust FOR EACH Customer
  WHERE Customer.NAME BEGINS "Z"
  NO-LOCK.

/** create, prepare and open a dynamic query **/
CREATE QUERY vhQryHdl.

vhQryHdl:SET-BUFFERS(BUFFER OrderLine:HANDLE).
vhQryHdl:QUERY-PREPARE("FOR EACH OrderLine").
vhQryHdl:QUERY-OPEN().
```

PUG Challenge Americas 2018

UDF Caused Failures

- ASSIGN with index fields *before* a UDF reference caused corruption or error.
- Cannot execute user defined function '<function>' in an ASSIGN statement after a key field change. (7954)
- Older KB says it was fixed in 8.3C
- Recent Progress tech says 10.2B

Tools To Tell

- COMPILE … XREF …
- vhQry:INDEX-INFORMATION()
- LOG-MANAGER
  - Command line controls
  - Run time controls

What’s Next…

- Laying the ground work
- Understanding the rules
- How it can all go wrong
- To index or not to index

Index Pros

- Fast bracket access
- Sorted access
- Foreign key links
- Enforce uniqueness
- Consider for common queries that need few columns

Index Cons

- Takes up space
- Can “break” existing code
- More updates required
- Smaller index may block larger

Use To ID And Control

- ❖ Primary keys
- ❖ Unique keys
- ❖ Foreign keys
- ❖ Enforce 1-to-1 versus 1-to-many
- ❖ Commonly needed small brackets

Be Careful With…

- High transaction tables
- Numerous small indices
- Nearly identical multi-field indices

Useful or Not?

- Index 1 (p)
  - Part-num
- Index 3
  - Ship-date
- New Index
  - Part-num
  - Cust-num
  - Ship-date

- Index 2
  - Cust-num

Query

WHERE part-num EQ…

AND cust-num EQ…
BY ship-date

Useful or Not?

- Index 1 (pu)
  - Cust-num
  - Inv-num
  - Ar-seq
- Index 3
  - Cash-cknum
  - Cust-num
  - Inv-num
  - Ar-seq
- Index 2 (u?)
  - Inv-num
  - Cust-num
  - Ar-seq
- Index 4
  - Inv-num

Useful or Not?

- Index 1 (pu)
  - Company
  - Vendor-num
  - Voucher
  - Trans-no
  - Payment-#
  - …

- Index 2
  - Company
  - Voucher
  - Vendor-num
- Index 3
  - Company
  - Vendor-num
  - Voucher

Useful or Not?

- Index 1 (pu)
  - Batch-num

- Index 2 (u?)
  - Batch-userid
  - Batch-num

Useful or Not?

- Index 1 (pu)
  - Location
  - Bin#
- Index 3 (non-u)
  - Part-num

- Index 2 (u?)
  - Part-num
  - Location
  - Bin#

Useful or Not?

- Index 1 (pu)
  - Sales-rep
  - Comm-cat
  - Cust-num
  - From-date
- Index 3 (u?)
  - Comm-cat
  - Sales-rep
  - Cust-num
  - From-date
- Index 2 (u?)
  - Cust-num
  - Comm-cat
  - Sales-rep
  - From-date

When To Stop?

- When do additional fields stop helping?
  - Field 1
  - Field 2
  - Unique field(s)
  - Field 4
  - Field 5

Cleaning Up

- Application-wide XREF
- Index logging over ~15 months
- Disable and wait for pain

Now that I’ve rambled on, are there any questions?

Final Notes

- Examples bundled with the presentation.
  - AllegroConsultants.com/about/downloads
- Thanks for attending!
- Please fill out your evaluations!