QAD Enterprise Applications

Enterprise & Standard Edition

Training Guide

Progress Programming for Open Edge

70-2931A
QAD Enterprise & Standard Edition
April 2009

QAD Inc.
100 Innovation Place
Santa Barbara, California 93108
Phone (805) 684-6614
Fax (805) 684-1890
http://www.qad.com

Contents

About This Course

Course Description

QAD designed this course for programmers working with QAD Enterprise Applications code. The intent of this course is to introduce basic database and programming concepts as they apply to QAD Enterprise Applications. Additionally, basic Progress Programming syntax will be covered to include custom report writing or changes to existing reporting capability. A working view of the data structures is also presented. Course includes hands-on programming to understand key concepts, as well as sample program takeaways for later reference.

Course Objectives

By the end of this class, students will be able to understand and work with:

- Database programming principles
- Database records
- Transactions and record scoping
- Predicates, indexes, and field lists
- Variables, messages, and conversions
- Validation and help
- Frames
- Reporting and Input/Output

Audience

This course is intended for:

- Beginning programmers who plan to customize reports within QAD Enterprise Applications
- Project Managers who want to understand the fundamentals of QAD Enterprise Applications as it pertains to Progress
- Members of QAD Enterprise Applications implementation teams

Prerequisites

A general understanding of QAD Enterprise Applications functions and an interest in how they work. Prior programming or database admin is suggested, but not required.

Course Credit and Scheduling

This course is valid 18 credit hours. It is designed to be taught in 3 days

System Requirements

This course uses a Progress OpenEdge version 10.1.C QAD training database. The training environment for the course is hosted on a Surgient system for virtual images. A student account is given to each trainee, and each student works in their own environment.

QAD Resources

If you encounter questions or problems on QAD software that are not addressed in this book, several resources are available.

Product Help

All QAD products ship with integrated help systems. A properly installed QAD application will display help when you press the Help key (F1), or access it through the menu. The help covers the normal use of the product.

QAD Web Resources

The QAD website provides product and company overviews. The Print Solution option on the opening page provides a means of compiling desired content into a document specialized to your industry, business implementation, and needs.

http://www.qad.com/

From QAD's main site, you can access QAD's Learning or Support sites.

QAD Learning Portal for Training Opportunities

To view available training courses, locations, and materials, use the QAD Learning Portal. Choose Learning under the Global Services tab to access this resource.

QAD Support for Product Documentation and the QAD Knowledgebase

To access release notes, user guides, installation and conversion guides by product and release, visit the Support website. Support also offers an array of tools depending on your company's maintenance agreement with QAD. These include the Knowledgebase and direct links to QAD Support experts.

Choose Support under the Global Services tab.

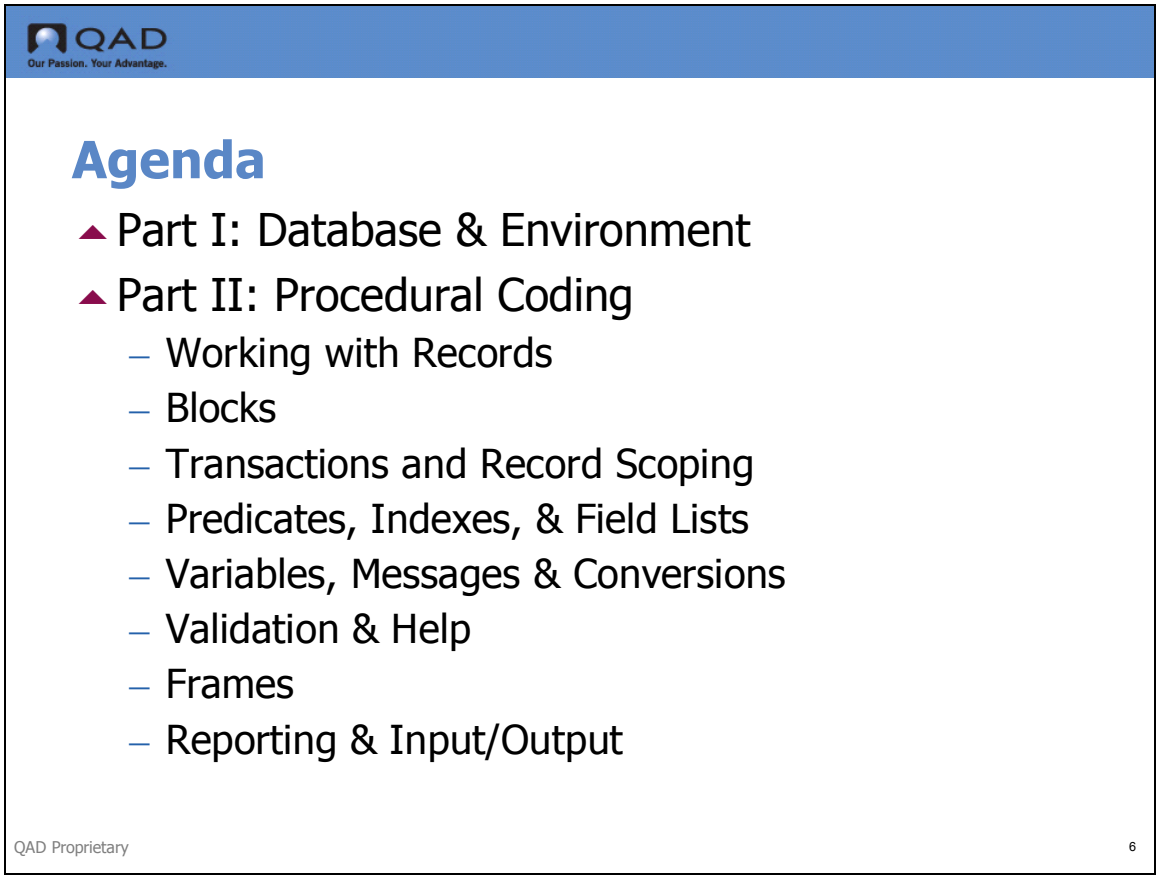
Any QAD customer can register for a QAD web account by accessing the Support web site and clicking the Accounts link at the top of the screen. Your customer ID number is required. Access to certain areas is dependent on the type of agreement you have with QAD.

SECTION 1

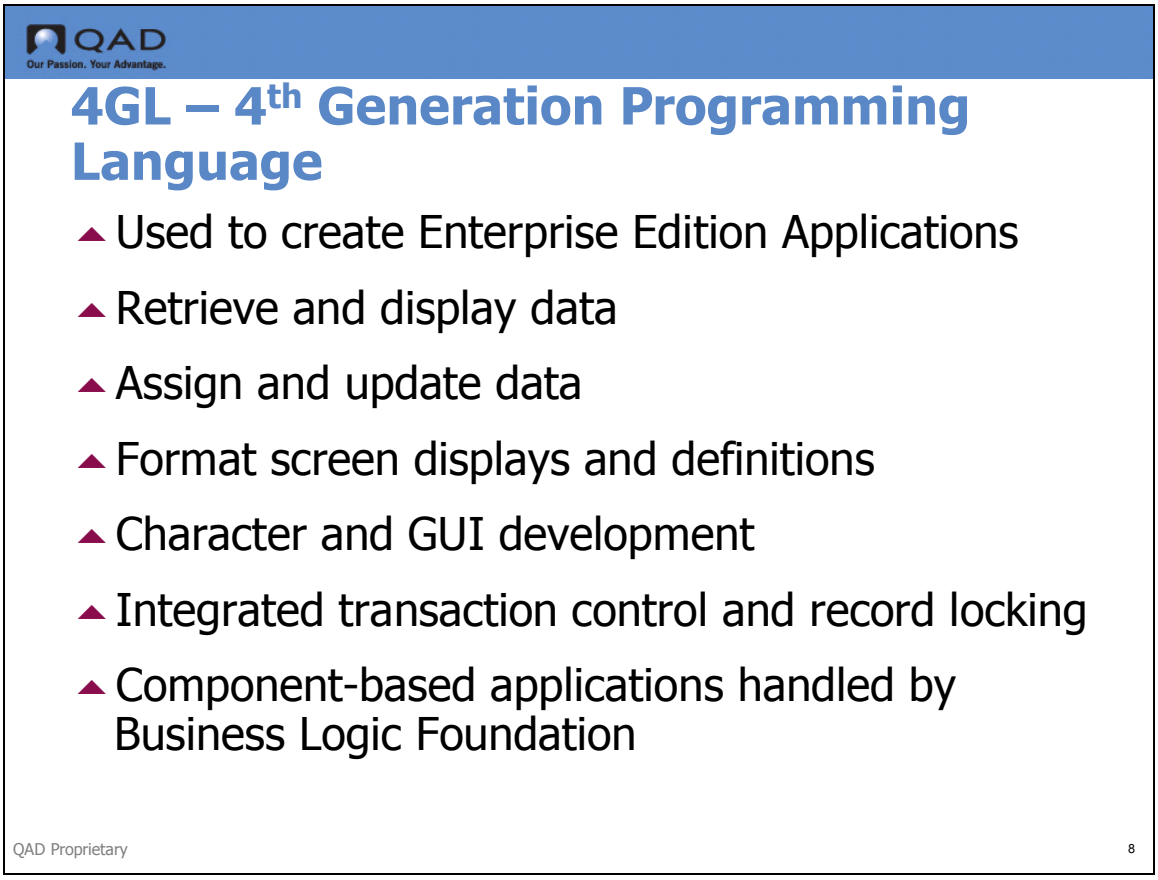
Database and Environment

CHAPTER 1

Database and Programming


QAD
Our Passion. Your Advantage.
Agenda
Part I: Database & Environment
Part II: Procedural Coding
Working with Records
Blocks
Transactions and Record Scoping
Predicates, Indexes, & Field Lists
Variables, Messages & Conversions
Validation & Help
Frames
Reporting & Input/Output
QAD Proprietary
6

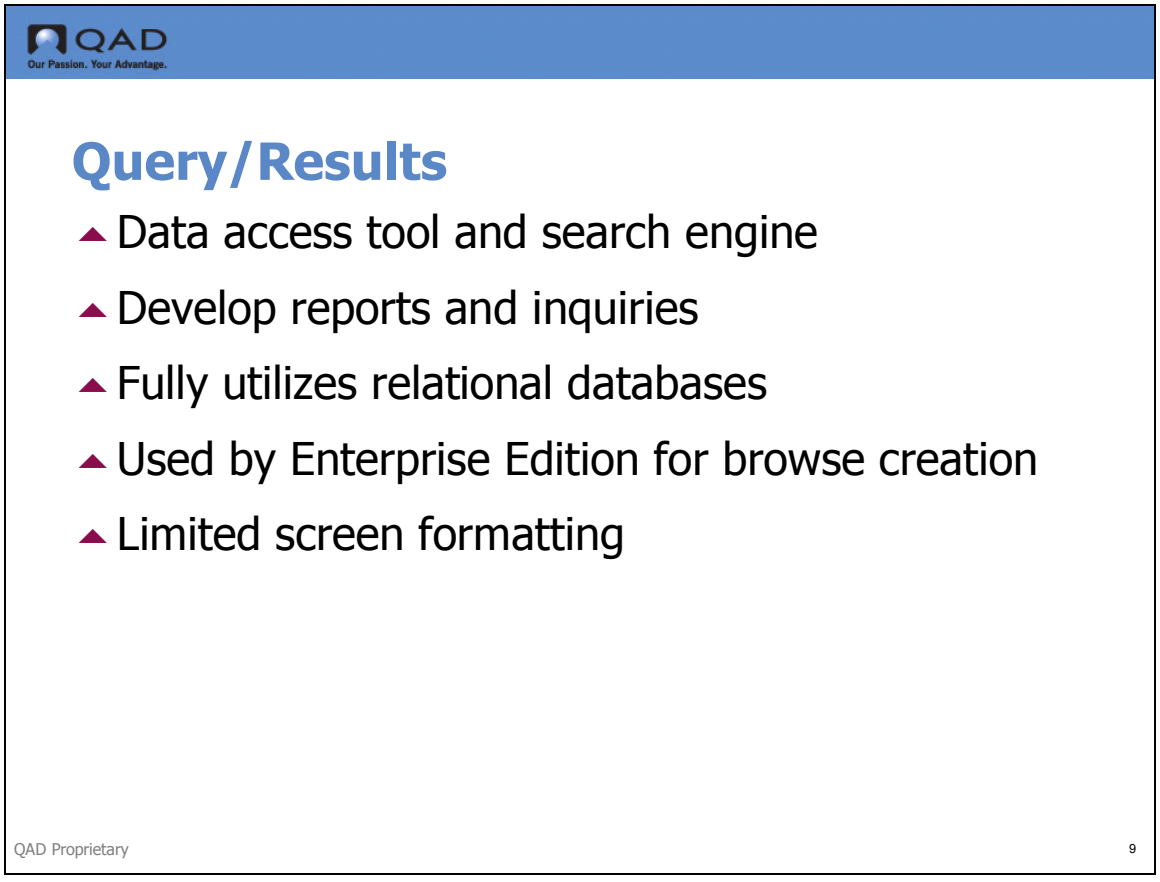
Agenda



4GL: 4th Generation Programming Language

Progress 4GL is the programming language in which Enterprise Edition is written. There are more than 9,000 programs within Enterprise Edition.

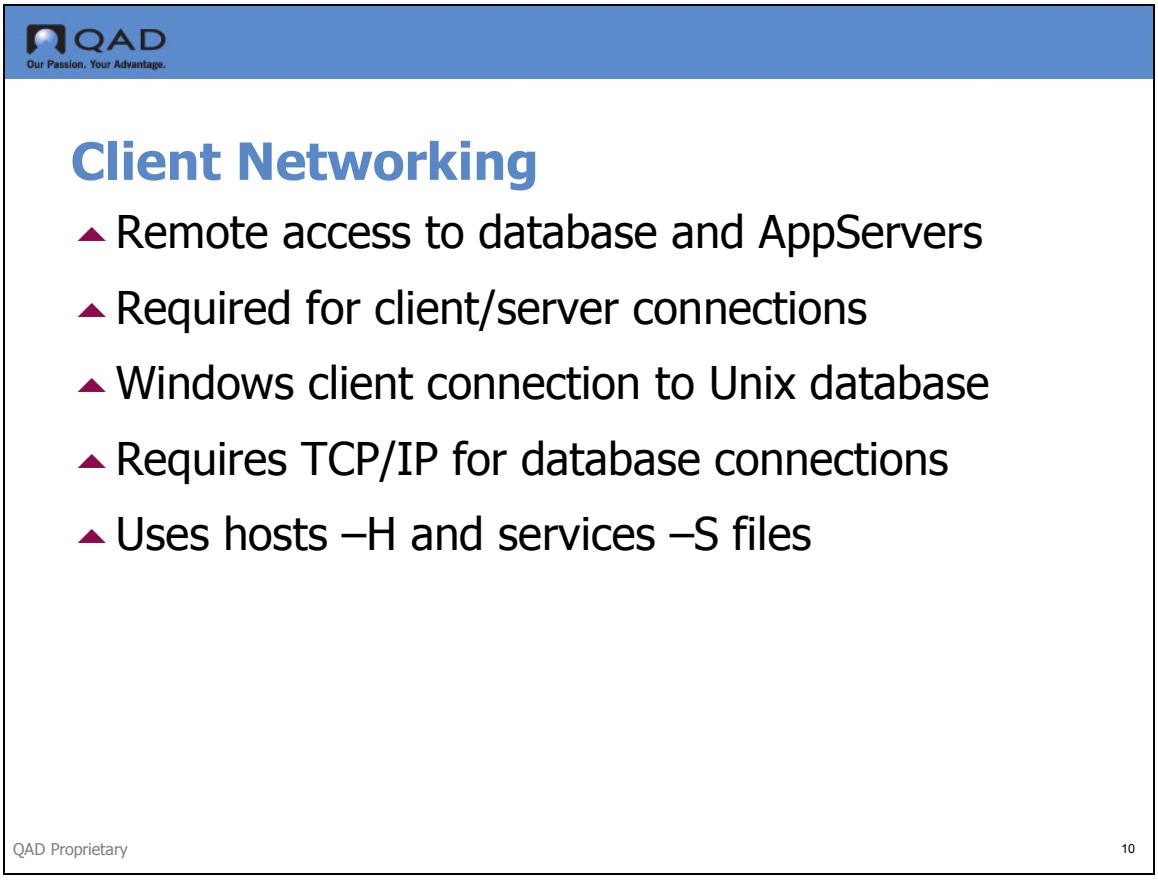
Note Component-based programs, such as Enterprise Edition Financials, are maintained using the Component Builder tool.



Query/Results

A Progress tool to create queries and reports

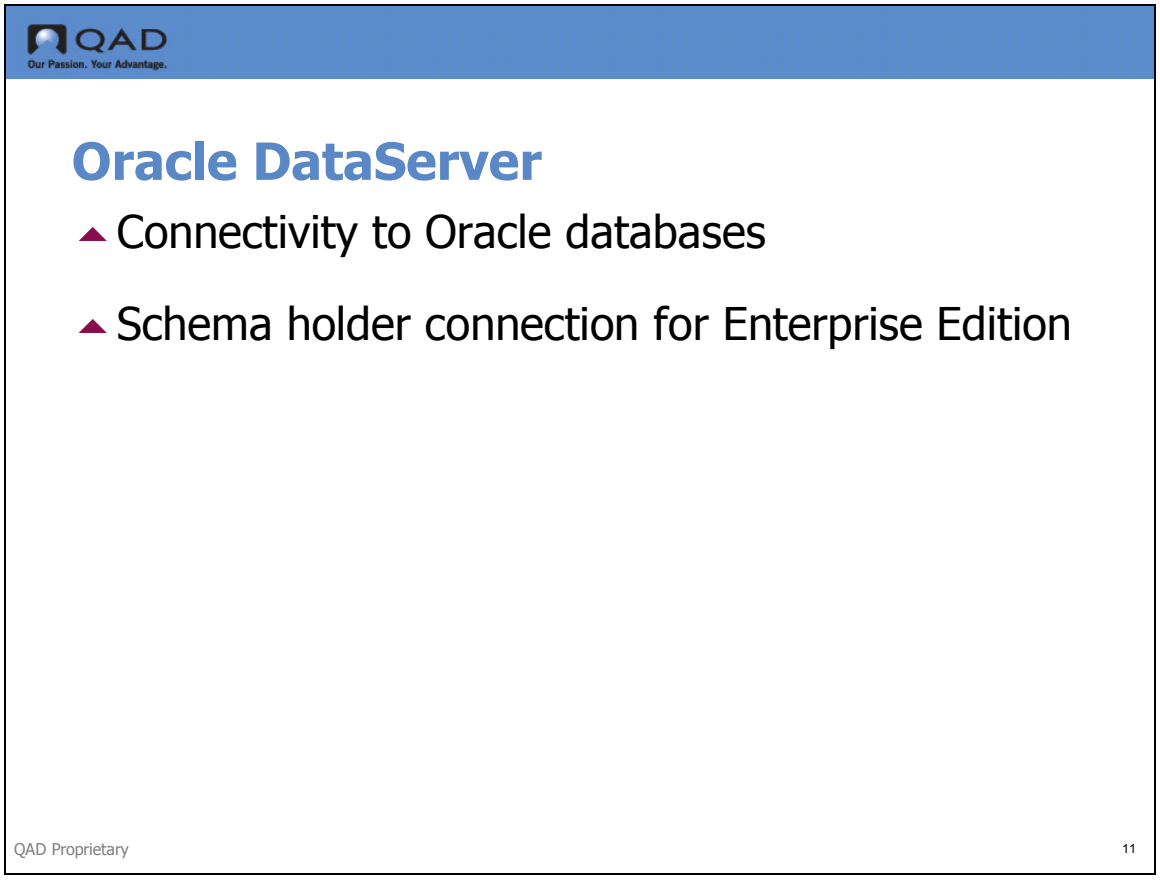
One license per user is required because the query engine is used to create the browses.



Client/Networking

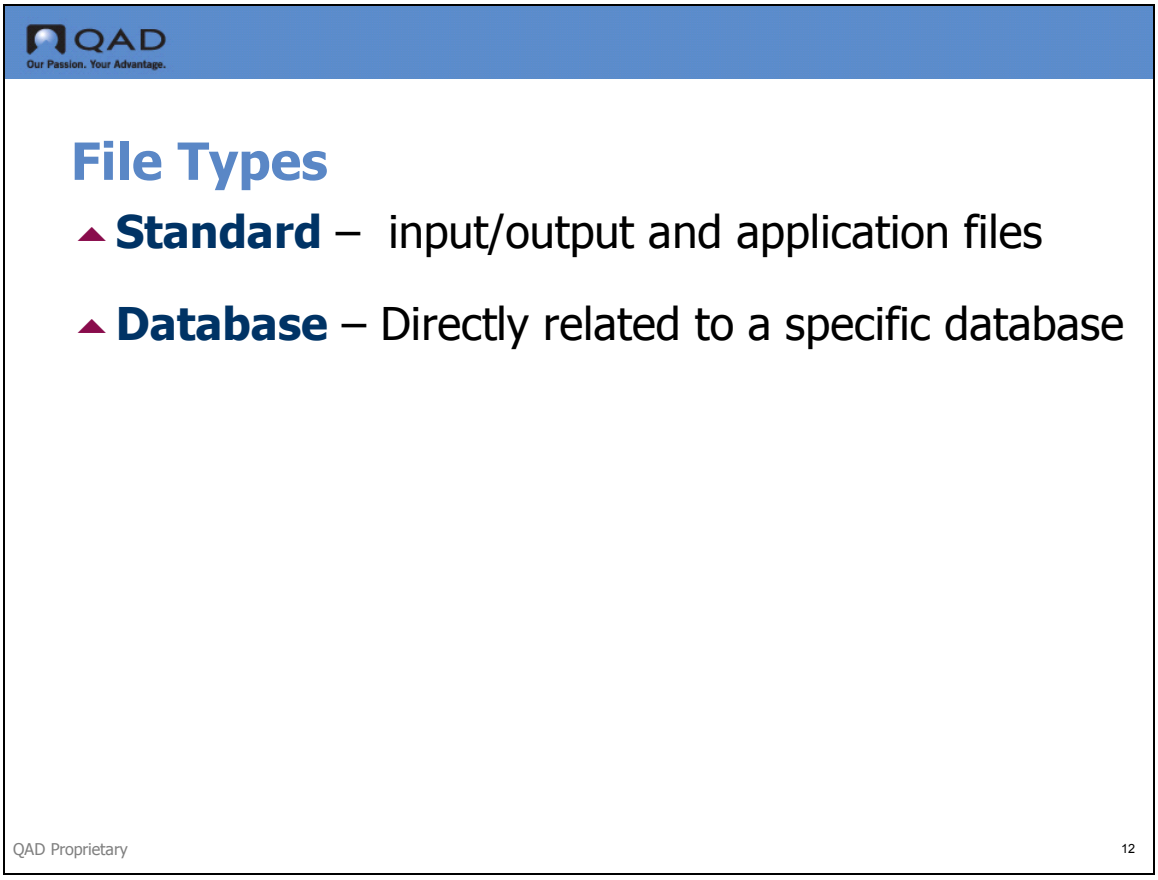
This is a free Product that may need to be specifically ordered.

Multi-Platform connectivity – connection between unlike operating platforms



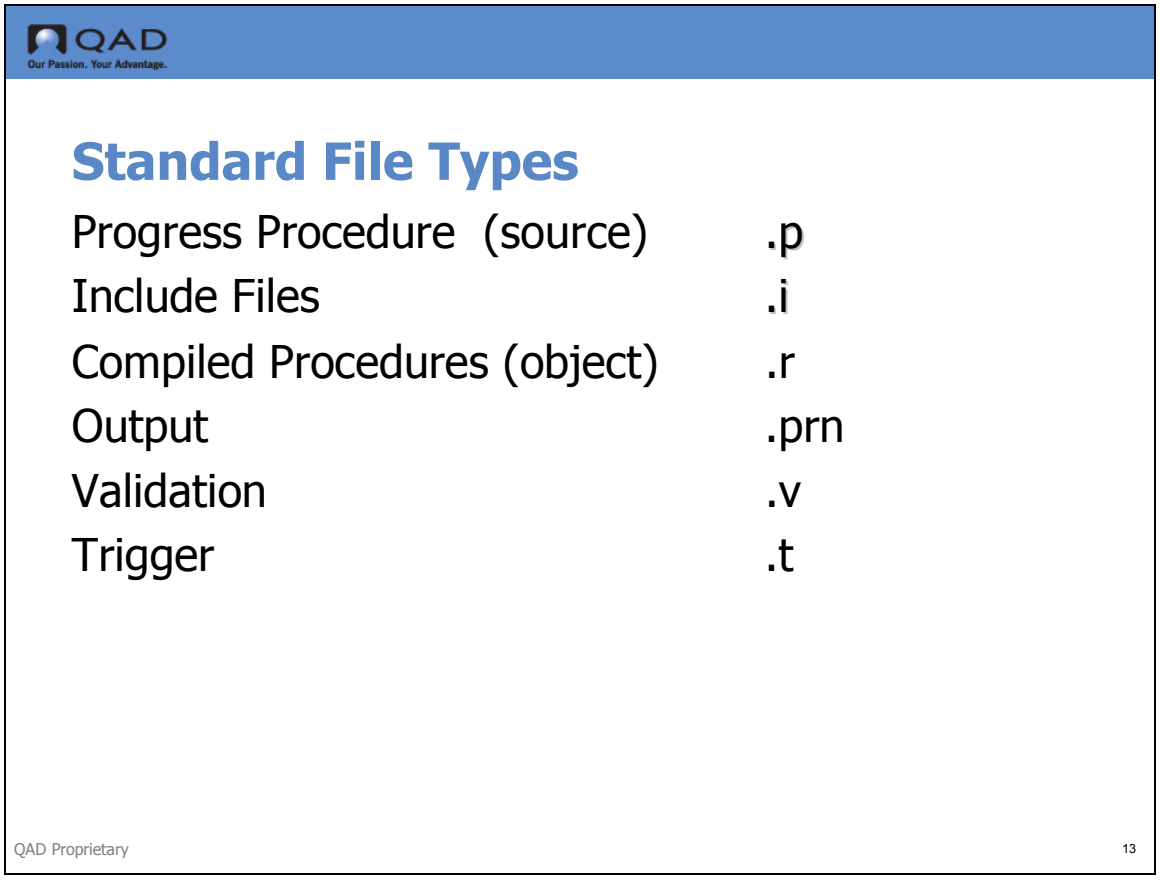
Oracle DataServer

Note This DataServer product is used to connect to the Oracle database through the schema holder.



File Types

- Standard – Files that are not required or necessary for the database. They are necessary for applications to run.
- Database - Usually identified by the database name, they are the files that physically hold the database and it definitions.



Standard File Types

- .p – Uncompiled code
- .i – Incomplete code segments used in multiple other files
- .r – Compiled code
- .prn – QAD Enterprise Applications report output file
- .v – Field testing and validation
- .t – Used for database integrity checking/Triggers

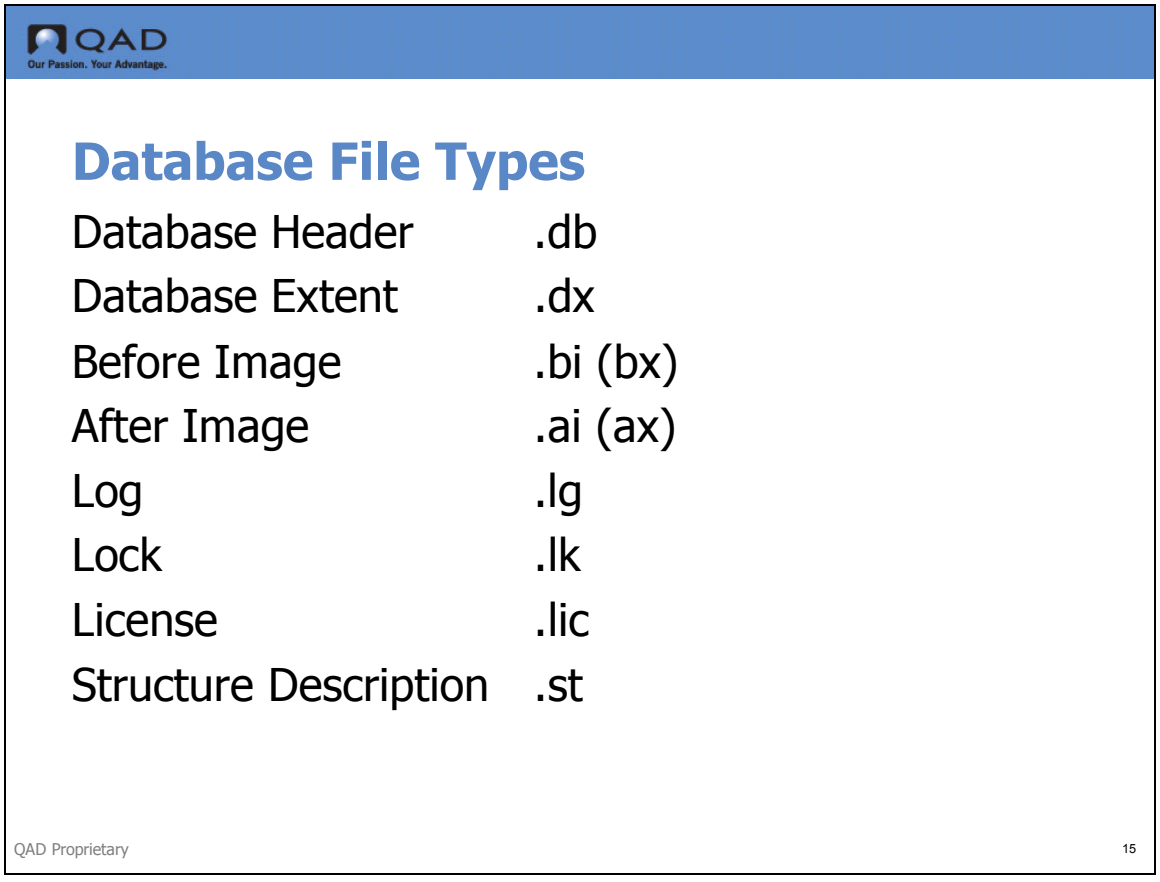
QAD
Our Passion. Your Advantage.

Standard File Types

OBCM	.w
Data (Input)	.d
Database Schema Definition	.df
Bulk Load File Description	.fd
Results Table Join	.qc (.qc7)
Procedure Library	.pl

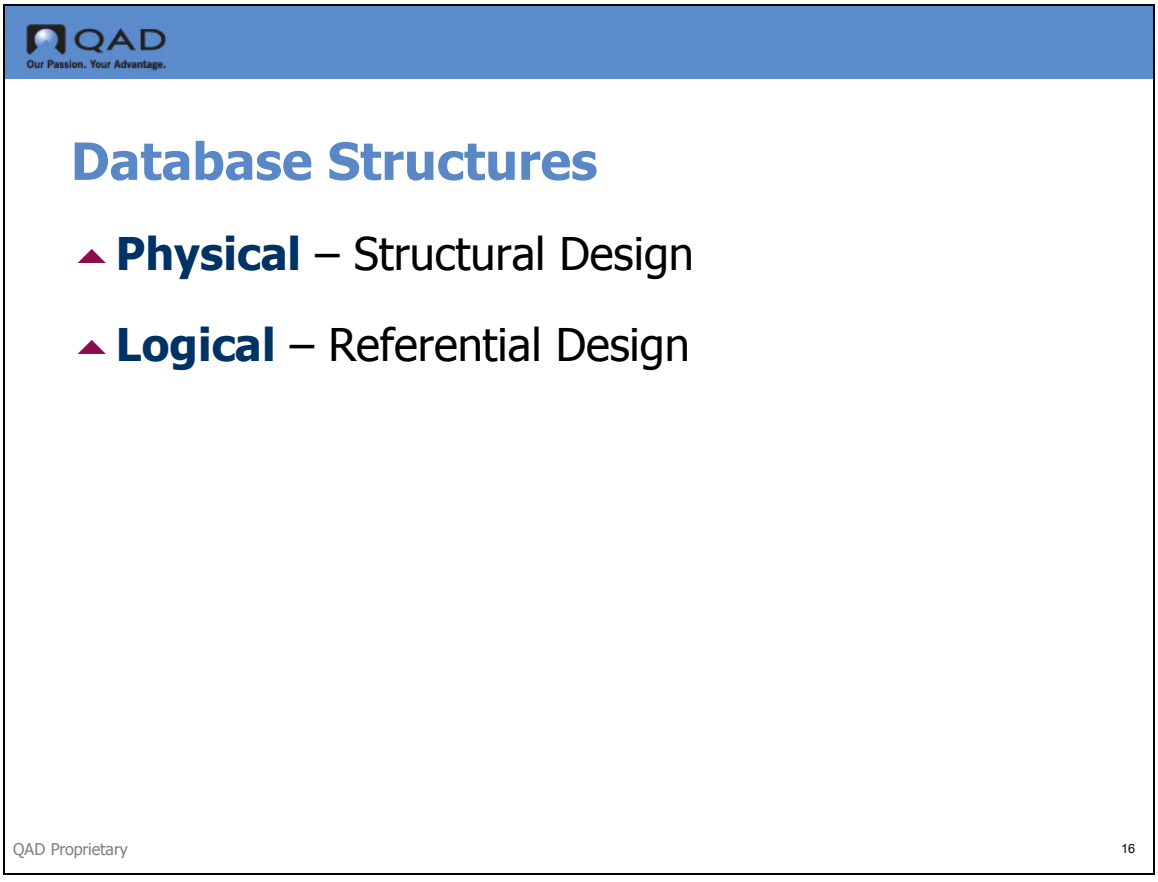
QAD Proprietary 14

- .w – Object Bases Component Module
- .d – import/export files containing specific table data
- .df – commands for creating a database; the database structure itself
- .fd – table load instructions; field descriptions within a table
- .qc – Contains the table relationships within the database
- .pl – GUI button images



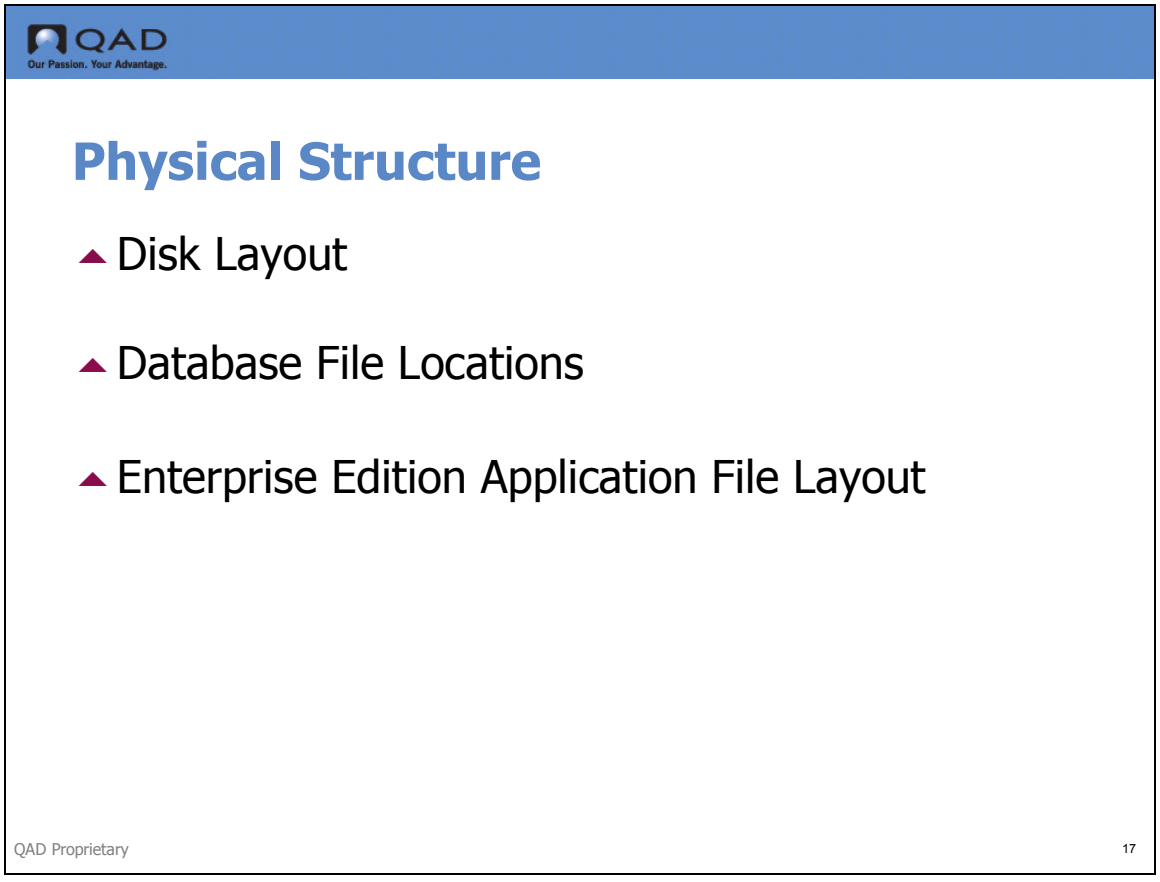
Database File Types

- db – Single volume database file
  or - Roadmap to multi-volume database files and information
- .dx – Database extent number (dbname.d5)
- .bi/.bx – The Before Image file or extent number file
- .ai/.ax - The After Image file or extent number file
- .lg – reporting log of database activity
- .lk – Existence of this file indicates the database in use updateable mode
- .lic – Keeps track of number of users logins
- .st – a map for creating a database



Database Structures

- Physical – Where the files are located
- Logical – Where the data is located

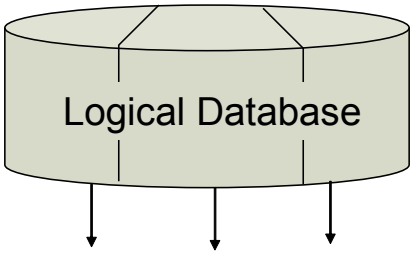


Physical Structure

QAD
Our Passion. Your Advantage.

Multi-Volume Overview

- One logical database comprised of several physical files
- Physical files placed across several physical disk drives for maximize I/O
- Likewise for After Image and Before Image files

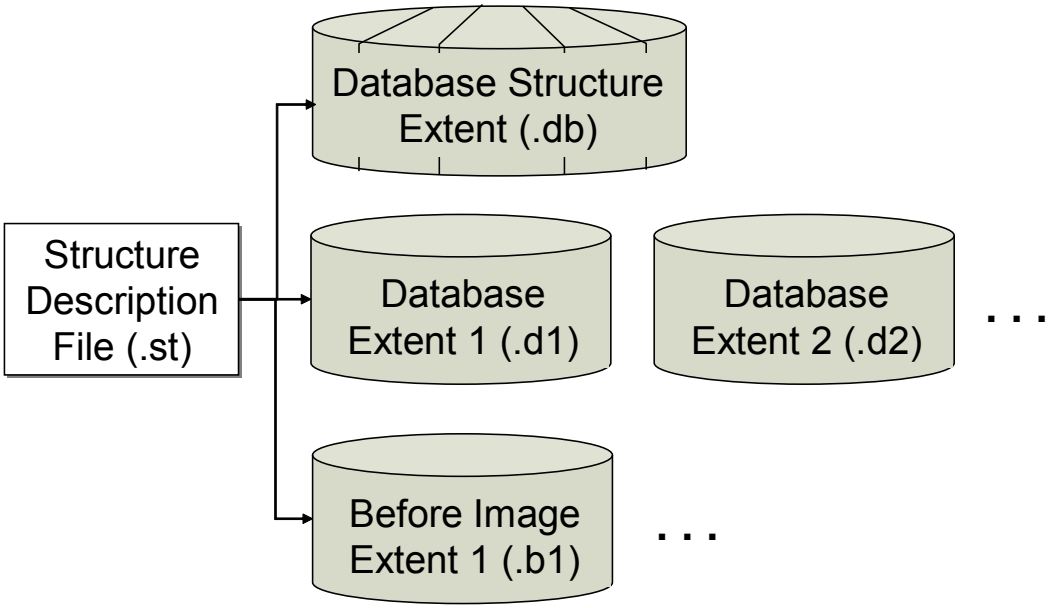


Physical Files (extents)

Multi-Volume Overview

QAD
Our Passion. Your Advantage.

Multi-Volume Components

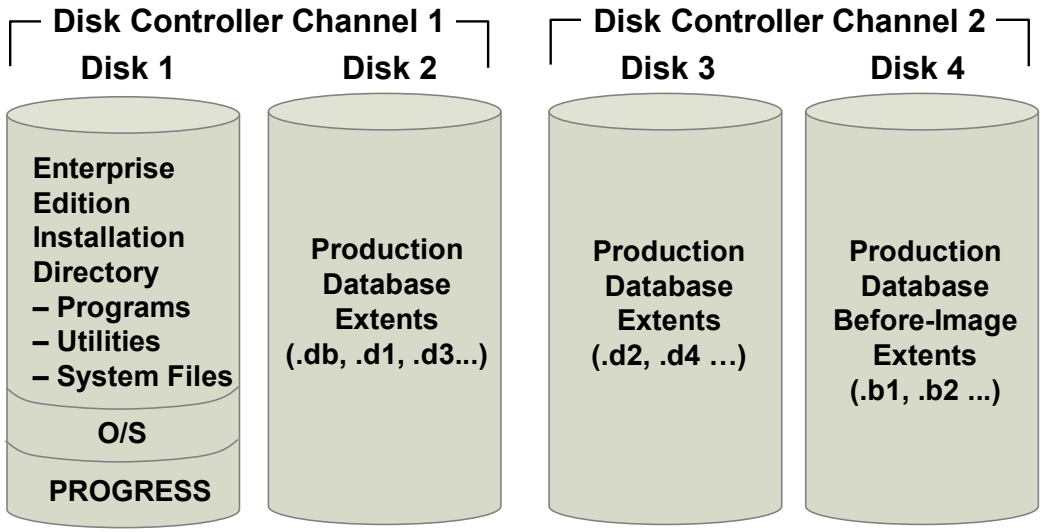


QAD Proprietary 19

Multi-Volume Components

QAD
Our Passion. Your Advantage.

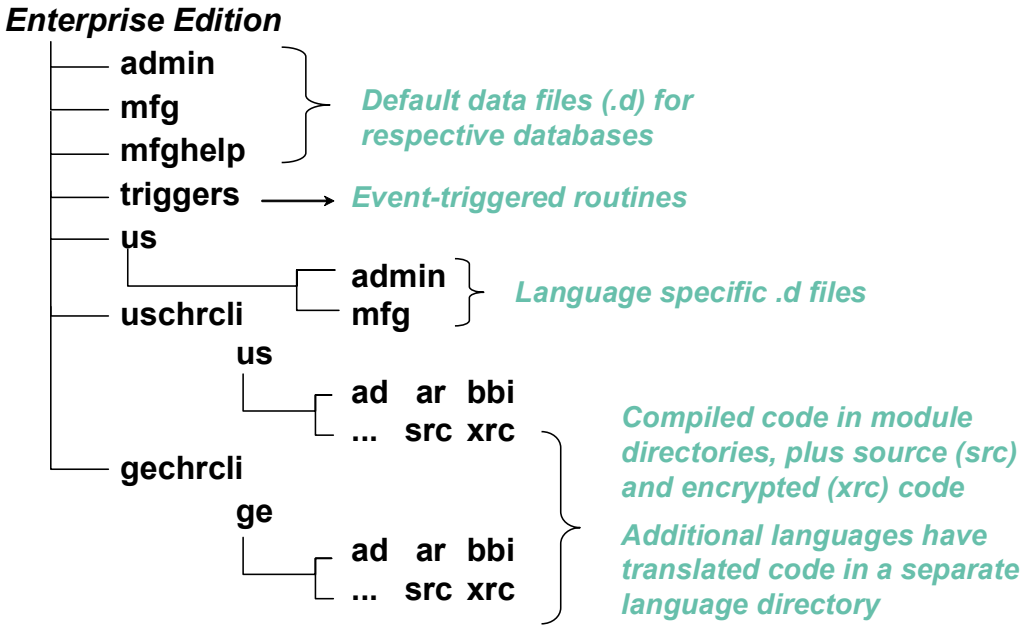
Disk Layout Example



Disk Layout Example

QAD
Our Passion. Your Advantage.

Enterprise Edition Directory Structure

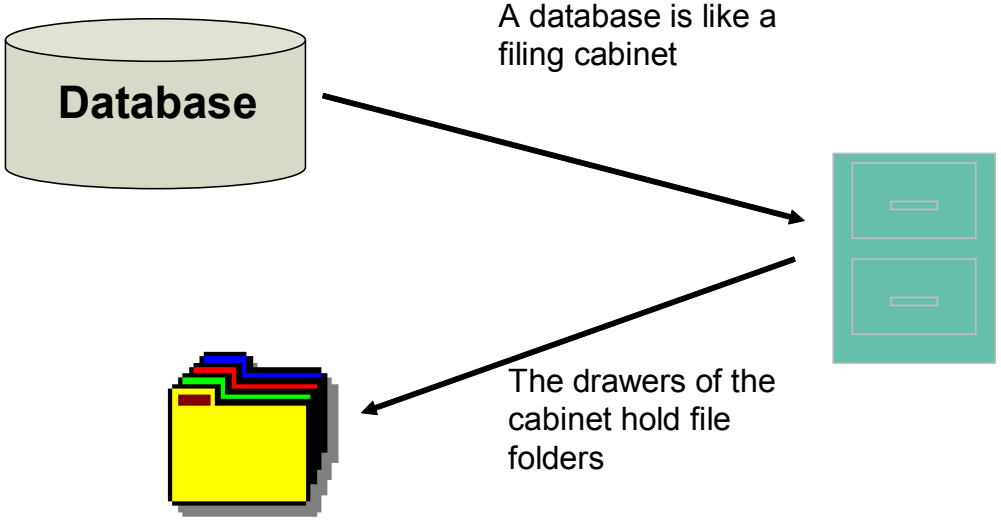


QAD Proprietary 21

Directory Structure

QAD
Our Passion. Your Advantage.

Logical Structure



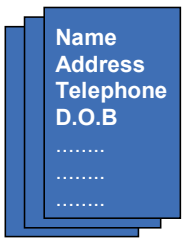
QAD Proprietary 22

Logical Structure

QAD
Our Passion. Your Advantage.

Logical Structure

A file holds a group of related information such as Customer Names and Addresses. These files are called Tables.



A particular Table holds many records, each of which details a particular object such as Employee.

Record segments, like Name, Address and Date of Birth, are known as *FIELDS*.

QAD
Our Passion. Your Advantage.

Logical Structure

Database

Each file in the cabinet represents a Table

Tables

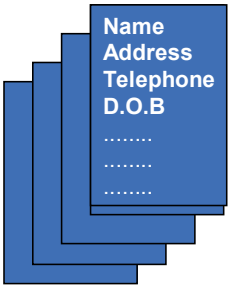
Record

Each entry in the Table represents a Record

QAD
Our Passion. Your Advantage.

Logical Structure

Records



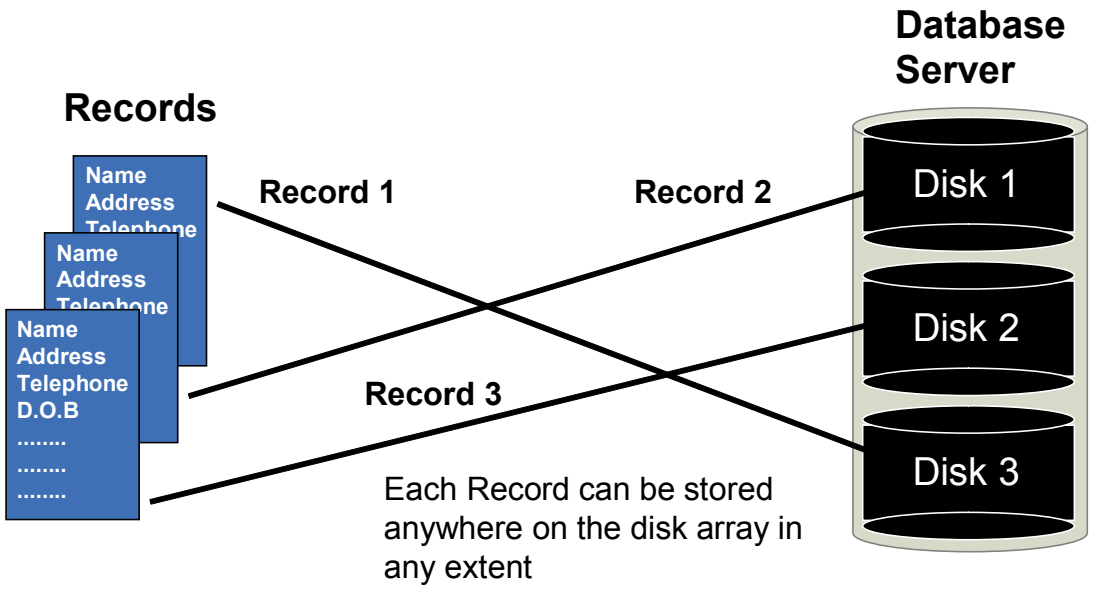
Each segment of the
Record represents a Field

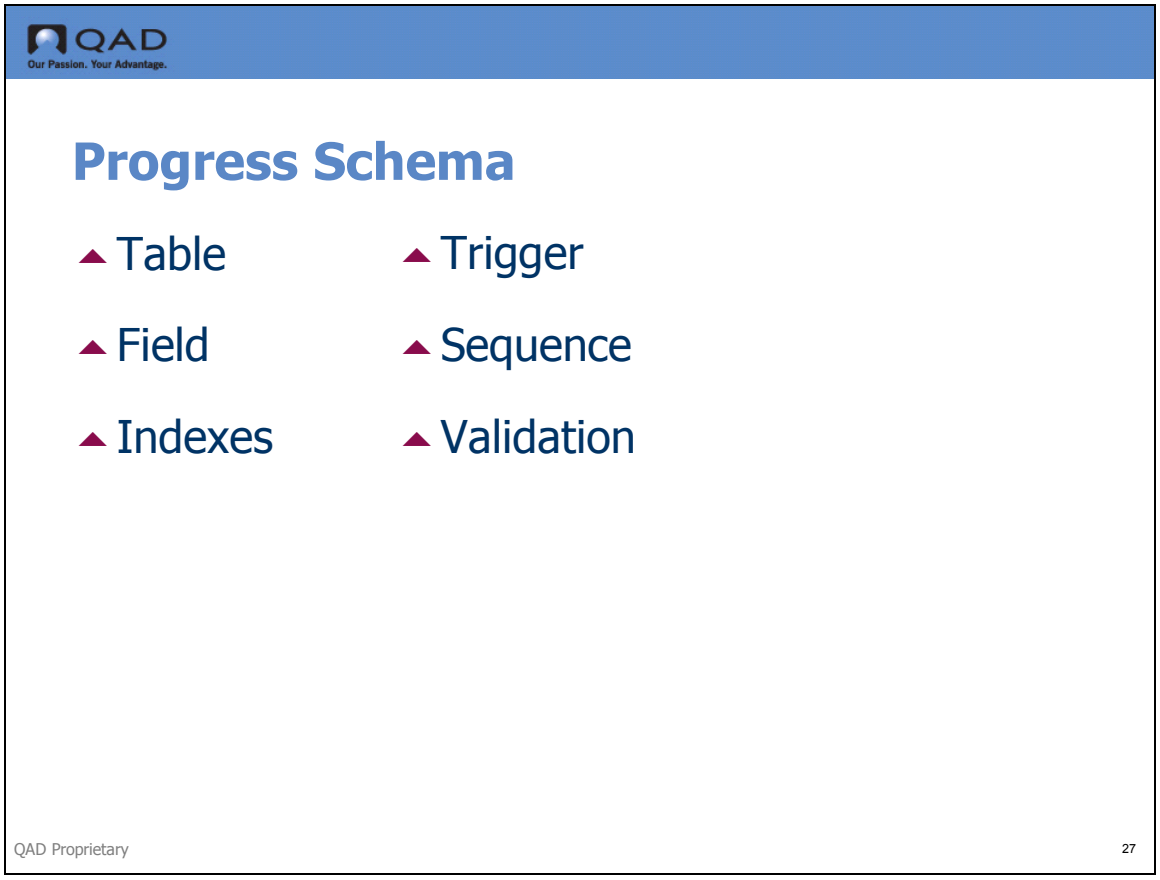
Fields

Nancy	QA Dr	x 7645	6/21/80	xxxx	xxxx

QAD
Our Passion. Your Advantage.

Logical Structure





Progress Schema

QAD
Our Passion. Your Advantage.

Progress Schema

- **Table** – Collection of similar data
- **Field** – One item of information in a Table
- **Indexes** – Structure to locate a specific record

QAD
Our Passion. Your Advantage.

Progress Schema

- **Trigger** – Event-driven procedure
- **Sequence** – Pointer to next occurrence
- **Validation** – Security and data integrity

QAD
Our Passion. Your Advantage.

Tables

- Collection of similar data
- Contains records or rows of data
- Contains fields or columns of data
- Indexed for performance

QAD Proprietary 30

Tables

QAD
Our Passion. Your Advantage.

Tables

	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
	Name	Addr	Phone	DOB	….	….
Rec 1	Bill	Main St	x 2887	5/11/67	xxxx	xxxx
Rec 2	Bob	1st Ave	x 9845	8/19/60	xxxx	xxxx
Rec 3	Nancy	QA Dr	x 7645	6/21/80	xxxx	xxxx
Rec 4	Sam	10th St	x 8347	9/18/45	xxxx	xxxx
Rec 5	Mary	Box 32	x 8934	2/12/77	xxxx	xxxx

Each Row is a Record in the Table

Each column is a Field of the Table

QAD
Our Passion. Your Advantage.

Records

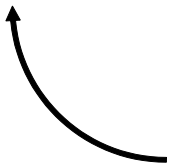
- Data in one row of the table
- Contain data in multiple fields
- Identified by unique recid in Database

Records

QAD
Our Passion. Your Advantage.

Record

	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
	Name	Addr	Phone	DOB	….	….
Rec 3	Nancy	QA Dr	x 7645	6/21/80	xxxx	xxxx



Each Row is a Record in the Table

QAD
Our Passion. Your Advantage.

Fields

- One segment of data in a record
- Stores data of particular type
- Multiple fields within a record
- One column in a Table

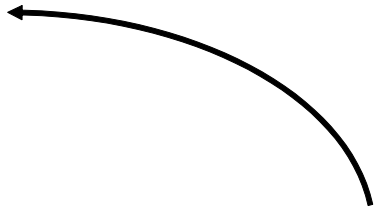
QAD Proprietary 34

Fields

QAD
Our Passion. Your Advantage.

Fields

	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
	Name	Addr	Phone	DOB	….	….
Rec 1		Main St				
Rec 2		1st Ave				
Rec 3		QA Dr				
Rec 4		10th St				
Rec 5		Box 32				



Each column is a Field in the Table
This column is the Addr field

QAD
Our Passion. Your Advantage.

Data Types: Default Display Formats

- Character "x(8)"
- Date "99/99/99"
- Decimal "->>,>>9.99"
- Integer "->,>>>,>>9"
- Logical Yes/No

QAD Proprietary 36

Data Types: Default Display Formats

QAD
Our Passion. Your Advantage.

Indexes

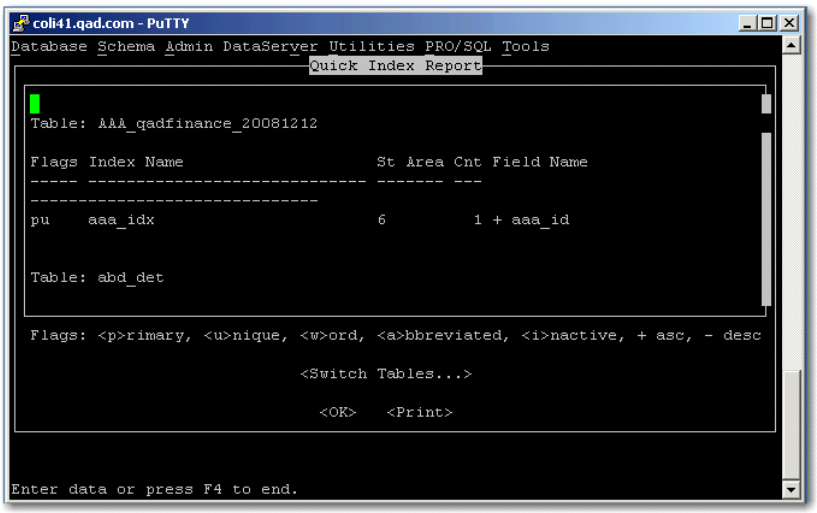
- An index sorts the records of a Table
- Index is based on one or more fields in the Table
- Value in the field decides place in the index
- Unique index – only one record in the Table with a specific combination of values

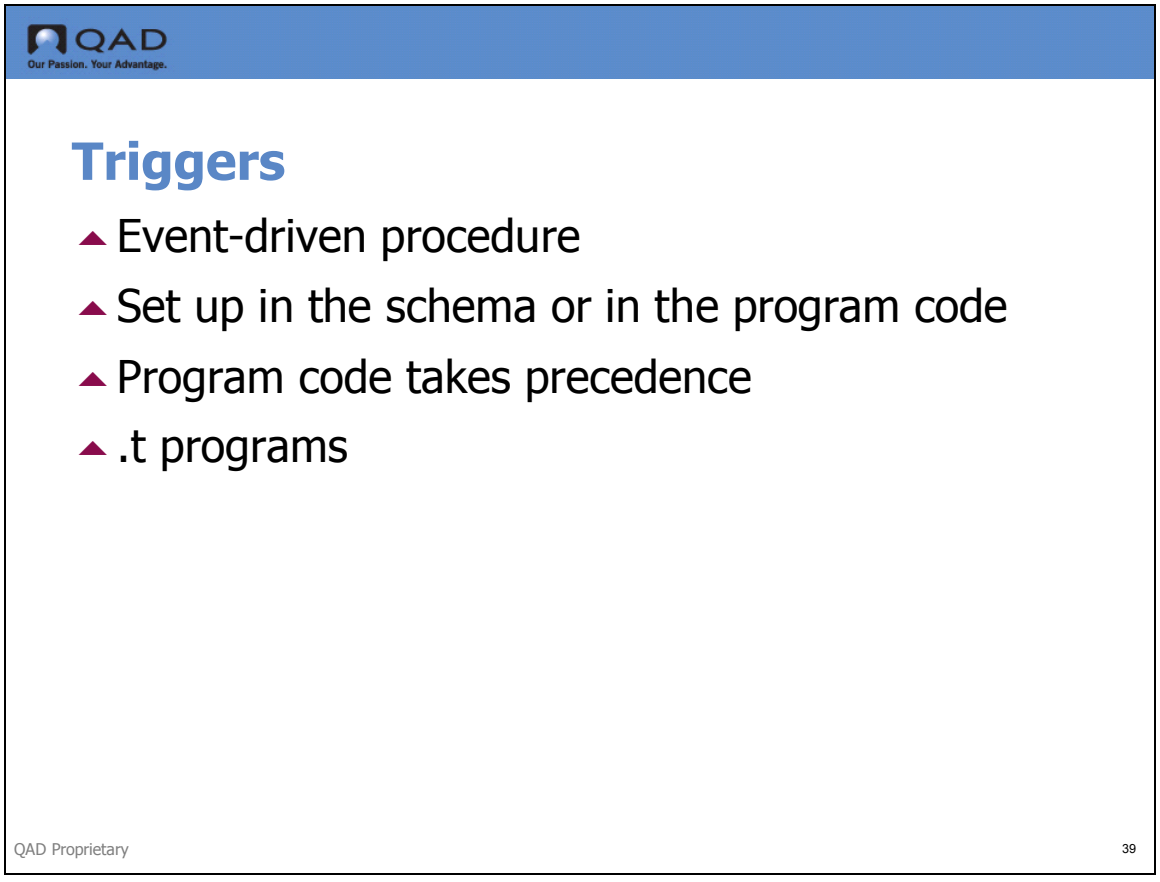
QAD Proprietary 37

Indexes

QAD
Our Passion. Your Advantage.

Indexes




QAD
Our Passion. Your Advantage.
Triggers
Event-driven procedure
Set up in the schema or in the program code
Program code takes precedence
.t programs
QAD Proprietary
39

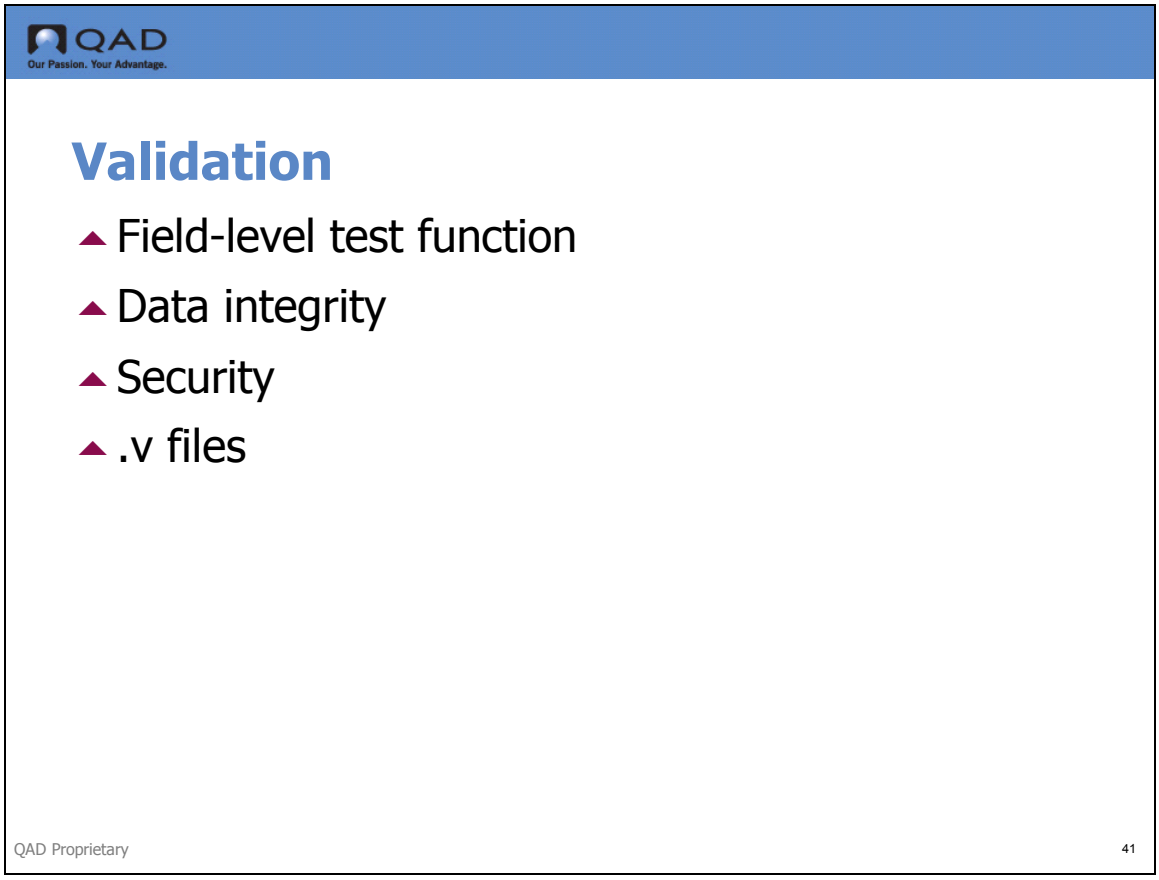
Triggers

QAD
Our Passion. Your Advantage.

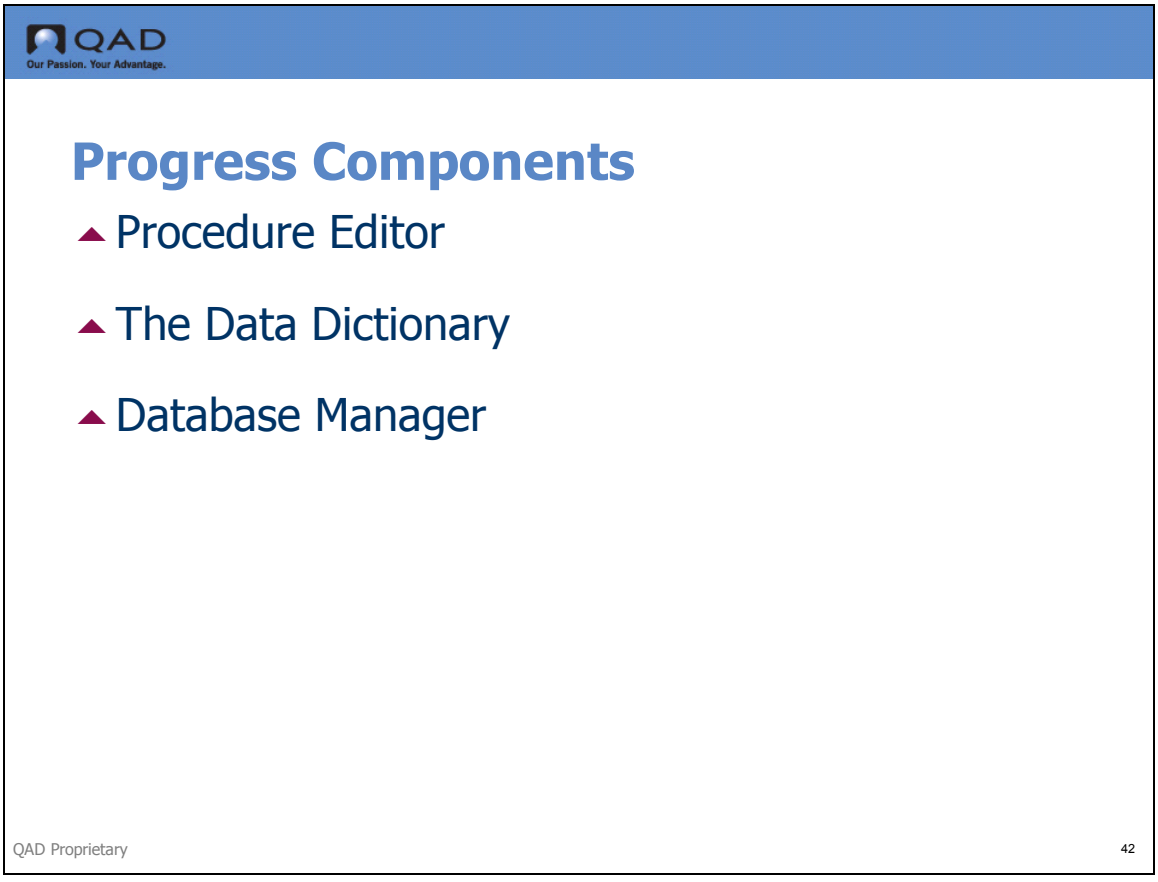
Sequences

- Auto-generated incremental integer values
- Stores the next occurrence
- Separate from the database Tables
- Definable increment values

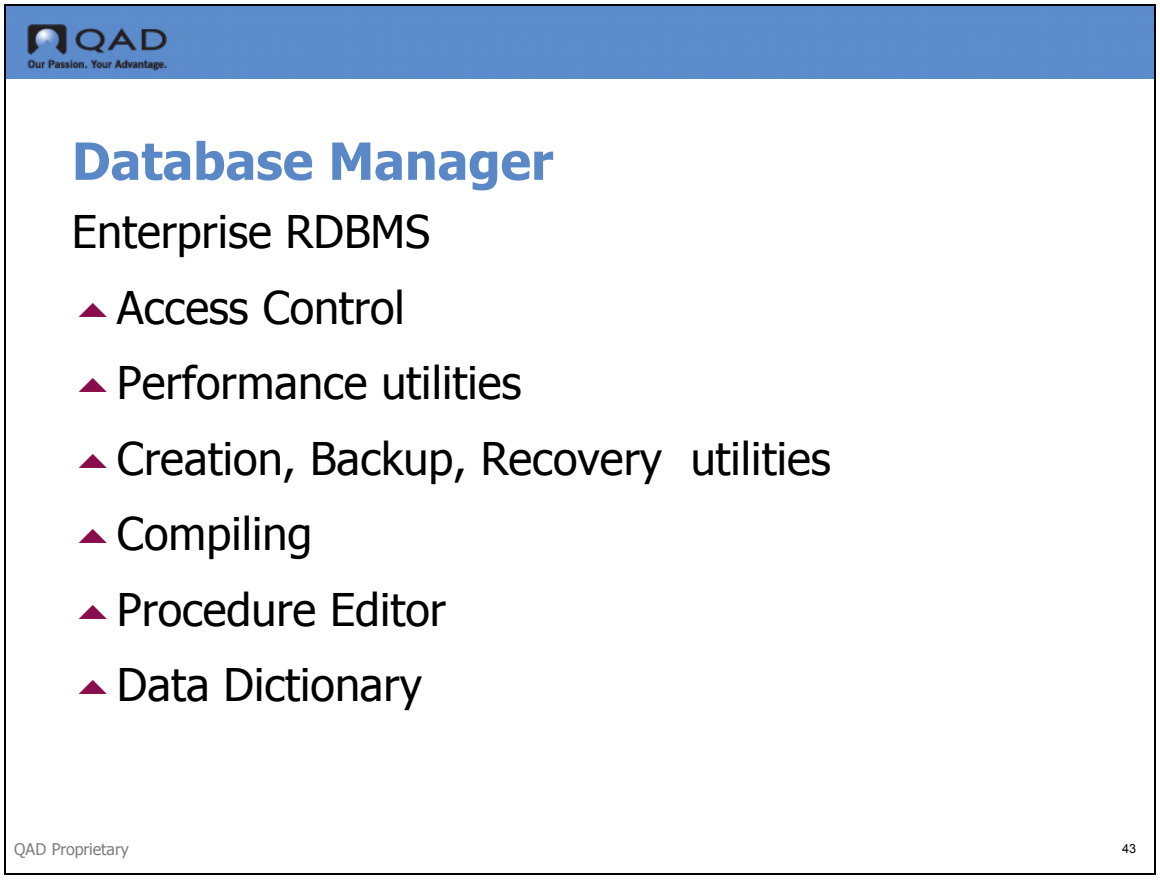
Sequences


QAD
Our Passion. Your Advantage.
Validation
Field-level test function
Data integrity
Security
.v files
QAD Proprietary
41

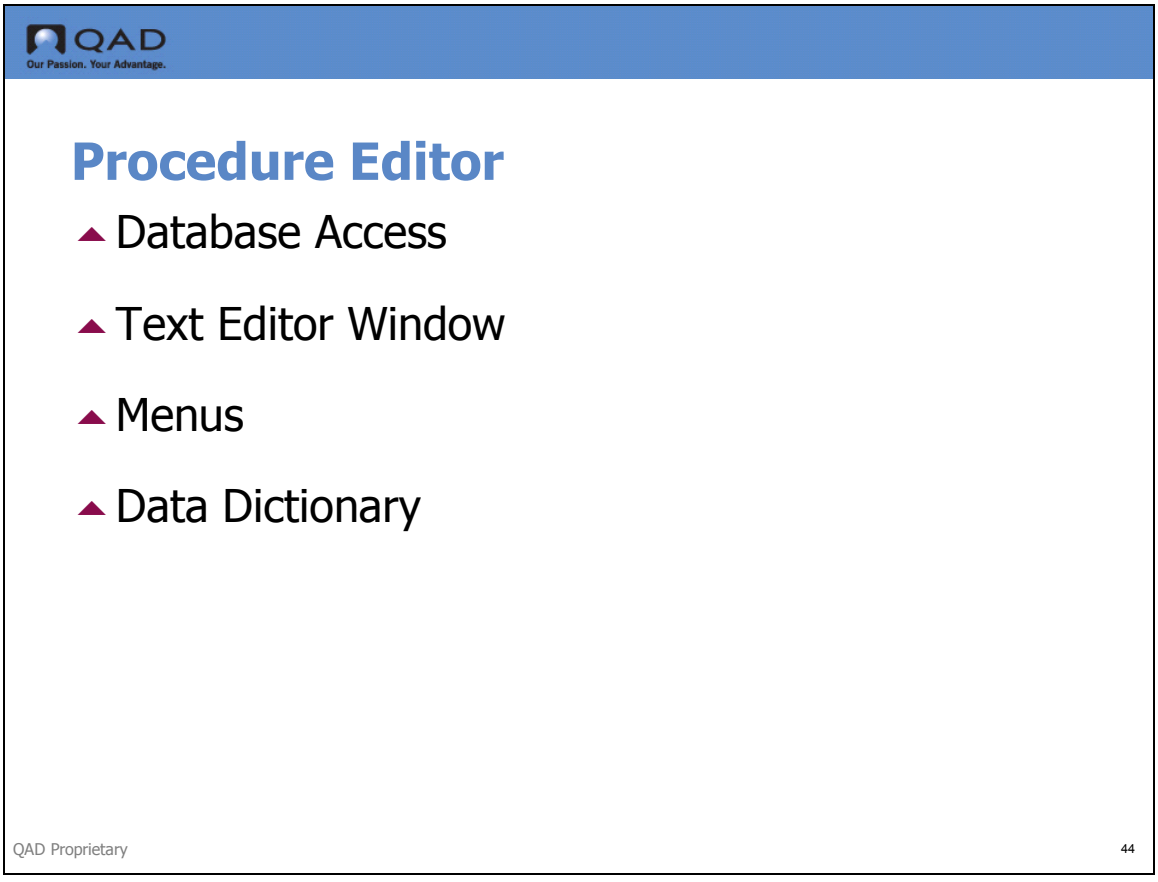
Validation



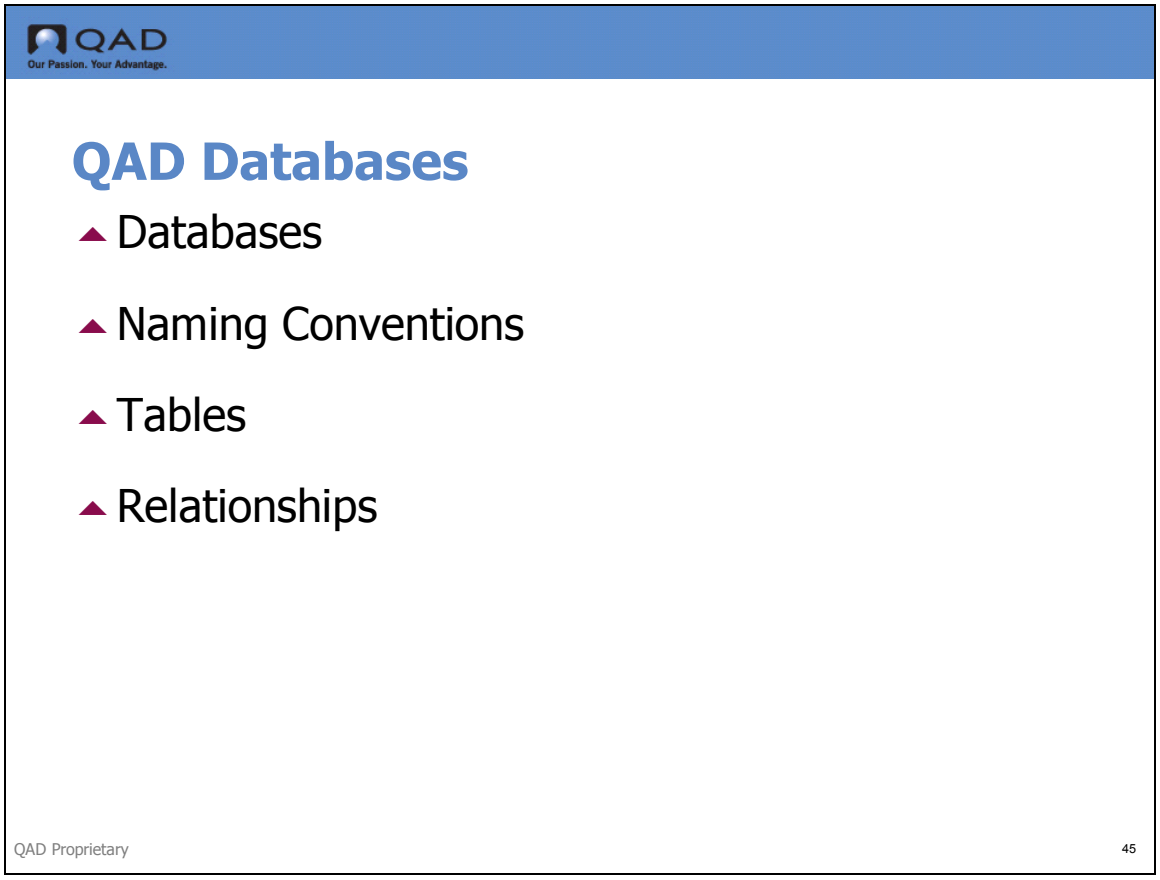
Progress Components



Database Manager



Procedure Editor



QAD Databases

QAD
Our Passion. Your Advantage.

QAD Databases

9.2	9.2b	93
Admin	Admin	Admin
Help	Help	Help
Mfg	Mfg	Mfg
	Audit	

QAD
Our Passion. Your Advantage.

Naming Conventions

- Underscore in field and file names
- Master files (Headers and Footers) are named xxxx_mstr and detail records are named xxxd_det (note the d!)
- Field names are prefixed by the filename prefix; that is, so_mstr has a field called so_nbr
- Control files (holding default system values) are named xxxc_ctrl

QAD Proprietary 47

Naming Conventions

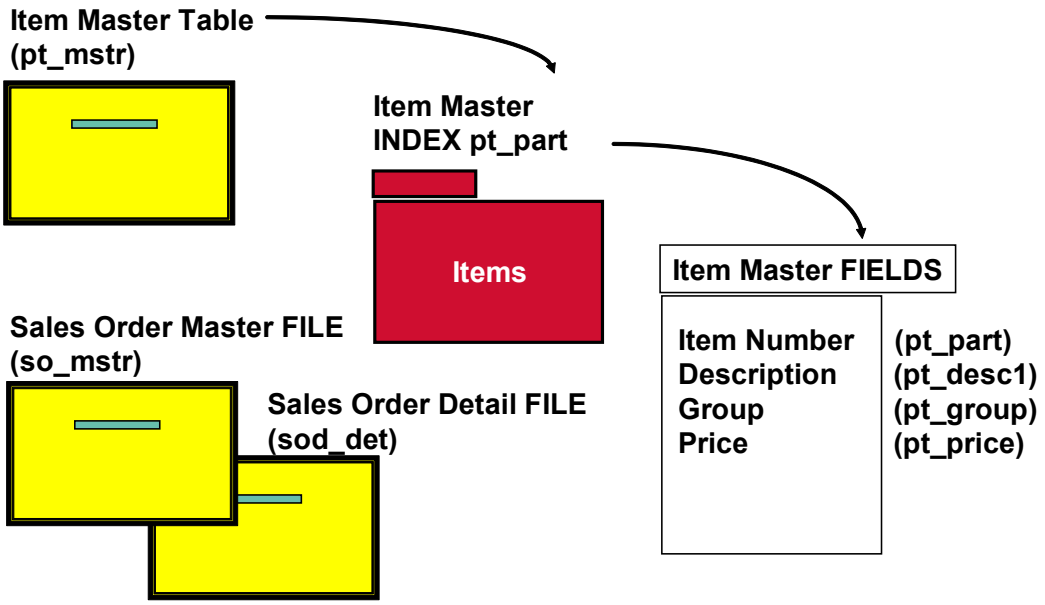
QAD
Our Passion. Your Advantage.

Naming Conventions

- History Files (used to record data after certain events, such as invoice posting) are named xxxx_hist
- Internal system workfiles are named xxxx_wkfl. The qad_wkfl is used by QAD development staff *ONLY!*
- User fields can either be *Double or Single Underscore Userx*; that is, xx__ or xx_user1 etc.
- Index names usually mimic field names

QAD
Our Passion. Your Advantage.

Naming Conventions



QAD
Our Passion. Your Advantage.

Tables

- so_mstr – Sales Order Master
- cm_mstr – Customer Master
- ca_mstr – Call Master
- tr_hist – Transaction History
- mnd_det – Menu Detail
- arc_ctrl – Accounts Receivable Control File
- qad_wkfl – QAD Workfile

Tables

QAD
Our Passion. Your Advantage.

Indexes

- so_mster has 10 indexes:

```
so_bill
      so_bill
      so_nbr
      so_ord_date
  so_nbr (primary, unique)
      so_nbr
  so_to_inv(unique)
      so_to_inv
      so_nbr
```

QAD Proprietary 51

Indexes

QAD
Our Passion. Your Advantage.

Indexes

- ca_mstr

```
ca_area
        ca_category
        ca_area
        ca_eu_nbr
    ca_assign
        ca_assign
        ca_category
    ca_nbr  (primary, unique)
        ca_category
        ca_nbr
```

QAD
Our Passion. Your Advantage.

Database Relationships

Sales Order File (so_mstr)

so_nbr	*so_cust*
SO00011	A111
SO00022	B111
SO00033	B111

Customer File (cm_mstr)

cm_addr	*cm_name*
A111	Smith Ltd
B111	Jones Ltd
C003	Thompson Plc

Part File (pt_mstr)

pt_desc1	*pt_part*
Front Seat	001
Door	002
Wheel	003

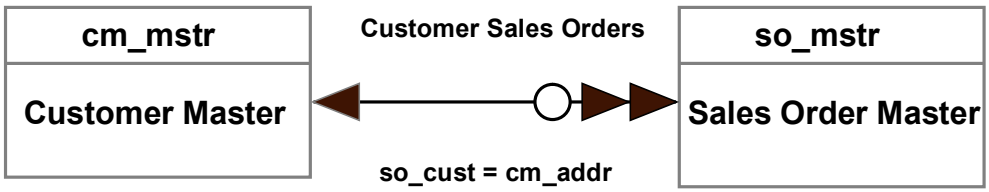
Sales Order Detail File (sod_det)

sod_part	*sod_nbr*	*sod_qty_item*
001	SO00011	3
001	SO00022	6
002	SO00022	10
003	SO00022	20
003	SO00033	4
002	SO00033	2

Database Relationships

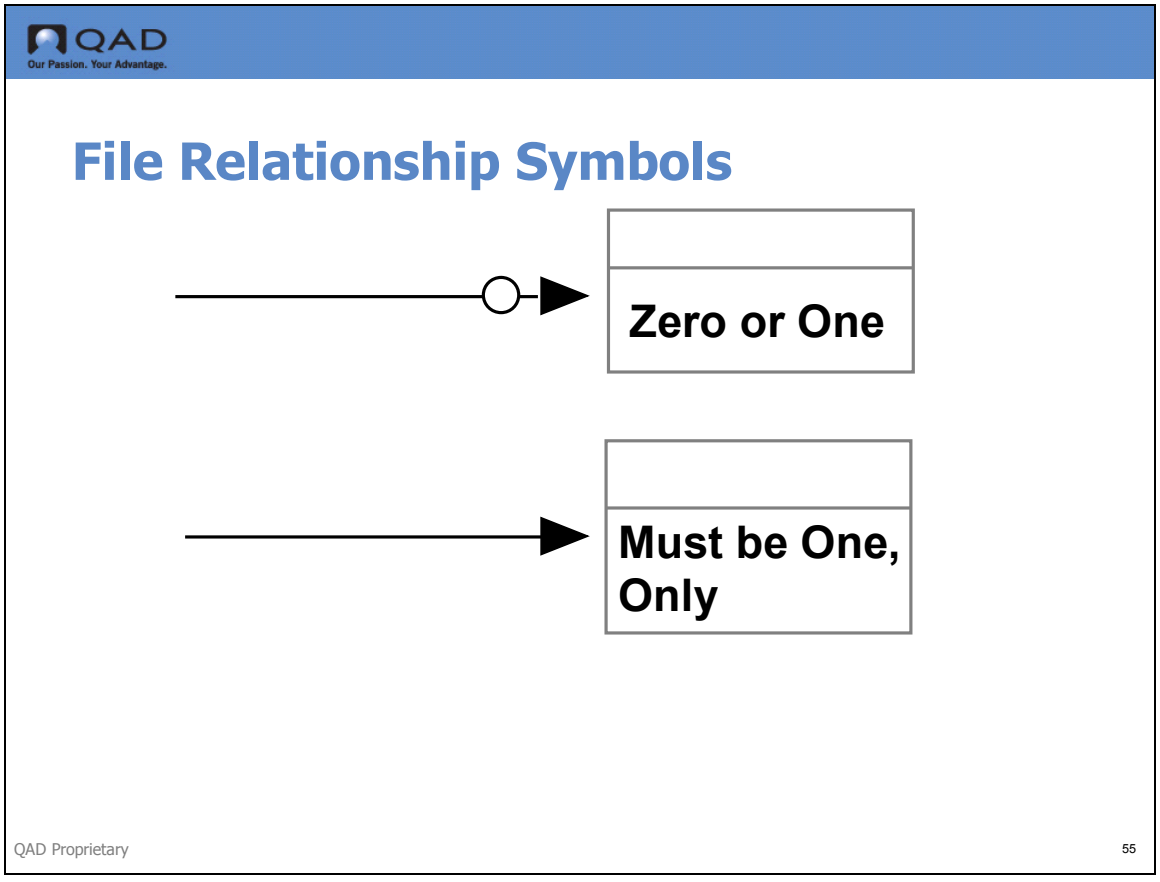
QAD
Our Passion. Your Advantage.

File Relationships Manual




```
FOR EACH cm_mstr :
    FOR EACH so_mstr WHERE so_cust = cm_addr:
        DISPLAY cm_addr so_nbr.
    END.
END.
```

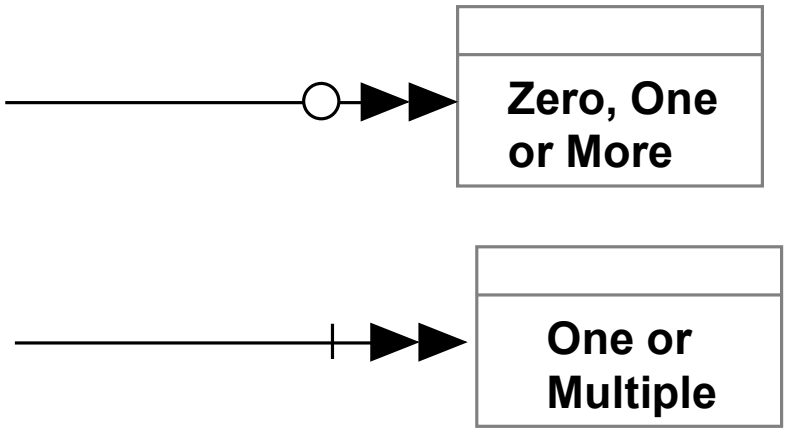

File Relationships Manual

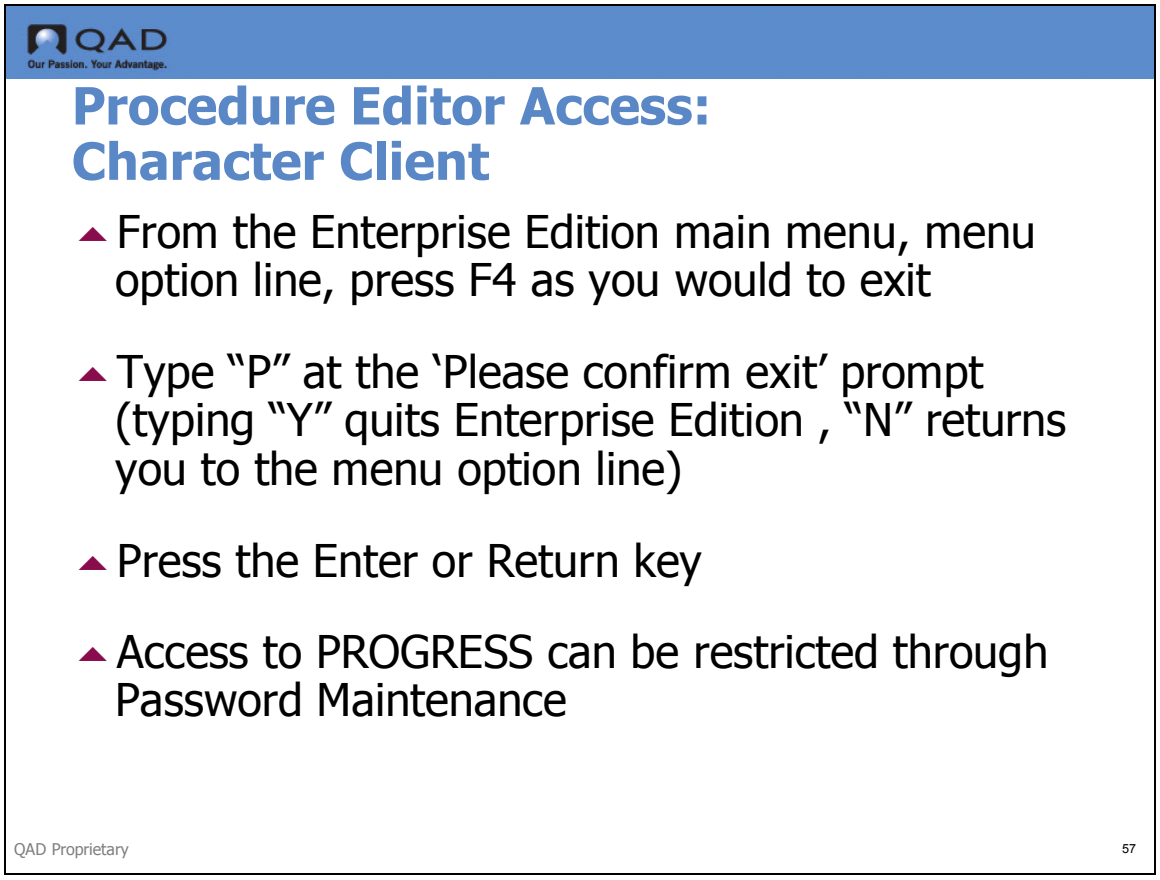


File Relationships Symbols

QAD
Our Passion. Your Advantage.

File Relationship Symbols

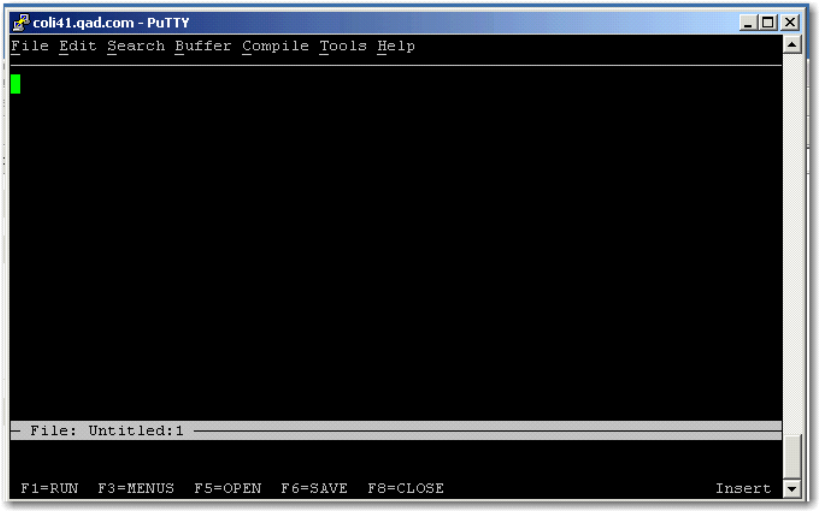




Procedure Editor Access: Character Client

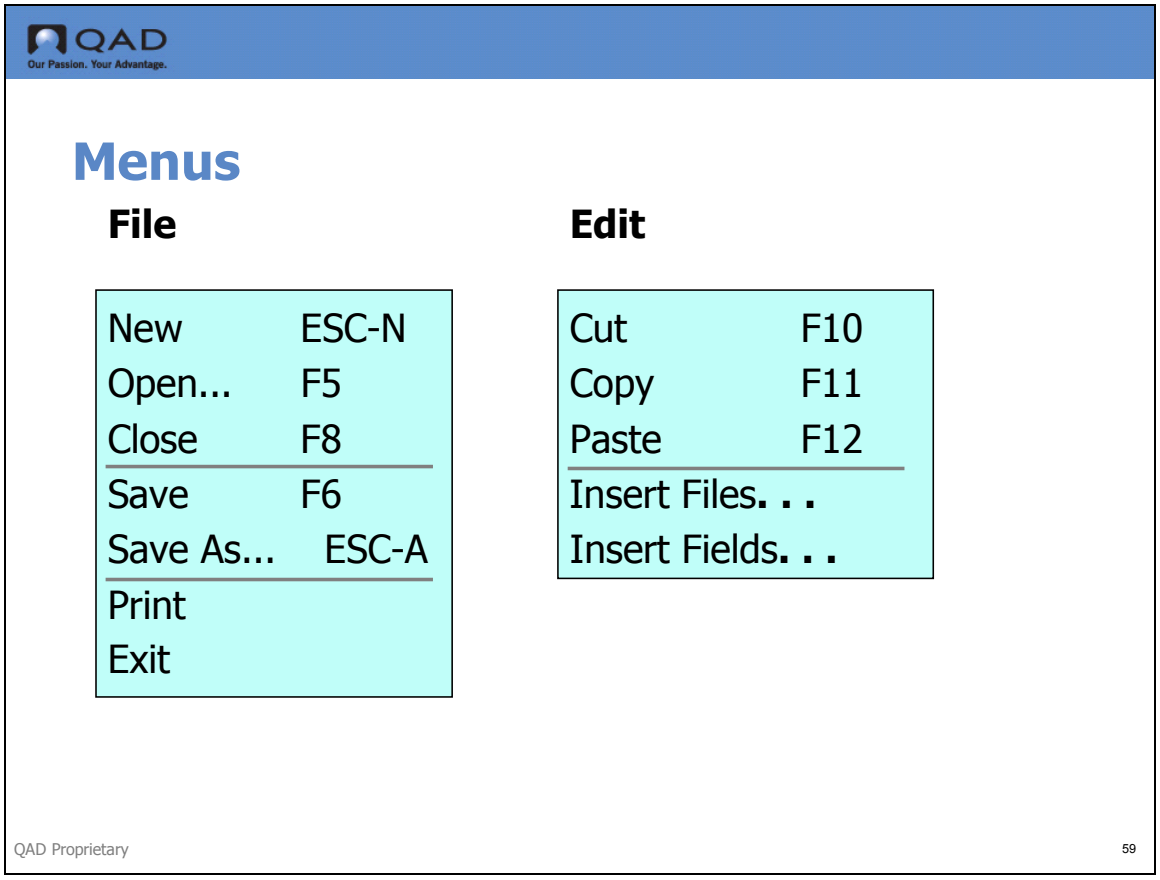
QAD
Our Passion. Your Advantage.

Procedure Editor Access: Character Window



QAD Proprietary 58

Procedure Editor Access: Character Window


QAD
Our Passion. Your Advantage.
Menus
File
New ESC-N
Open... F5
Close F8
Save F6
Save As... ESC-A
Print
Exit
Edit
Cut F10
Copy F11
Paste F12
Insert Files. . .
Insert Fields. . .
QAD Proprietary
59

Menus

QAD
Our Passion. Your Advantage.

Menus

Search

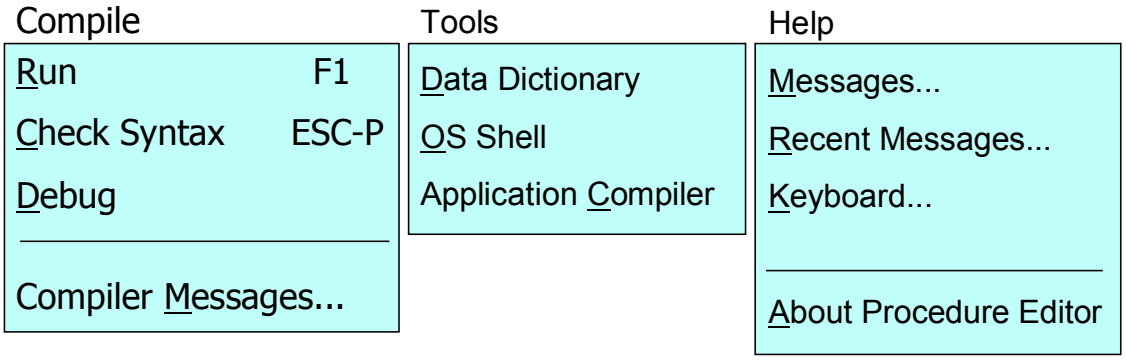
Find. . .	CTRL-F
Find Next	ESC-F
Find Previous	ESC-I
Replace. . .	ESC-R
Goto Line. . .	ESC-G

Buffer

List. . .	ESC-B
Next Buffer	ESC-TAB
Previous Buffer	ESC-CTRL-U
Information. . .	

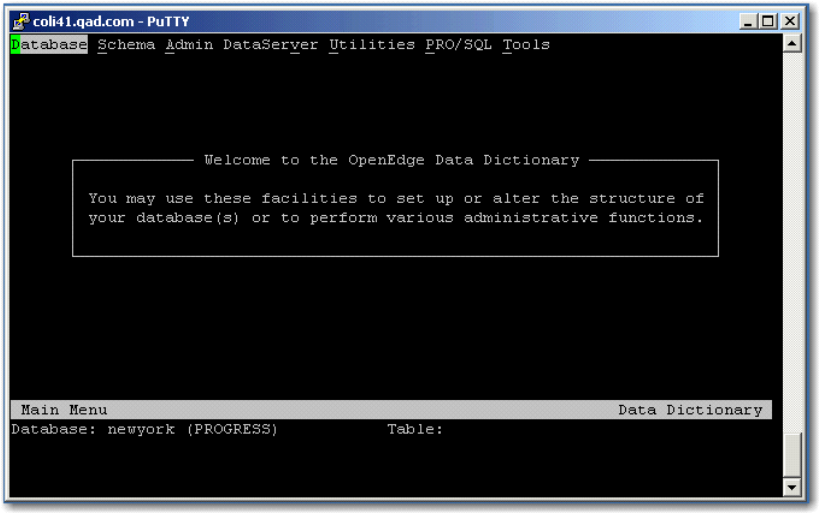
QAD
Our Passion. Your Advantage.

Menus

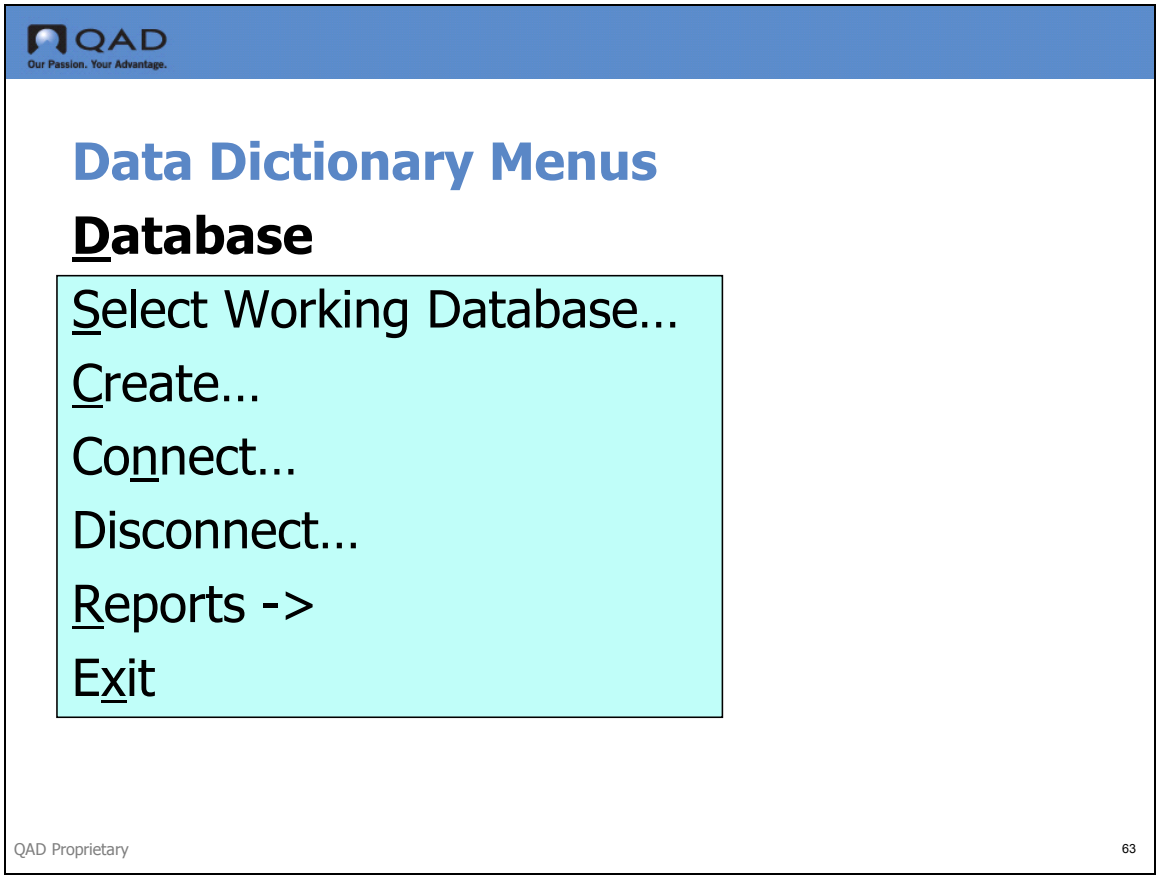


QAD
Our Passion. Your Advantage.

Data Dictionary: Character Window



Data Dictionary: Character Window



Data Dictionary Menus

QAD
Our Passion. Your Advantage.

Data Dictionary Menus

Schema

Modify Table...
Add New Table...
Delete Table(s)...
Field Editor...
Reorder Fields...
Global Field Name Change...
Index Editor...
Sequence Editor...
Adjust Field Width...

QAD
Our Passion. Your Advantage.

Data Dictionary Menus

Admin

Dump Data and Definitions ->
Load Data and Definitions ->
Database Identification->
Security ->
Export Data ->
Import Data ->
Create Bulk Loader Description File…
Database Options…
Enable Large Key Entries

QAD
Our Passion. Your Advantage.

Data Dictionary Menus

DataServer

MS SQL Server Utilities->
ODBC Utilities->
ORACLE Utilities->

QAD
Our Passion. Your Advantage.

Data Dictionary Menus

Utilities

Editor for Parameter Files...
Quoter Functions ->
Generate Include Files ->

Edit PROGRESS Auto-Connect List...
Freeze/Unfreeze...
Index Deactivation...

Information...

QAD
Our Passion. Your Advantage.

Data Dictionary Menus

PRO/SQL

PRO/SQL View Report
Dump as CREATE VIEW Statement…
Dump as CREATE TABLE Statement…

QAD
Our Passion. Your Advantage.

Data Dictionary Menus

Tools

Procedure Editor
OS Shell
Application Compiler

QAD
Our Passion. Your Advantage.

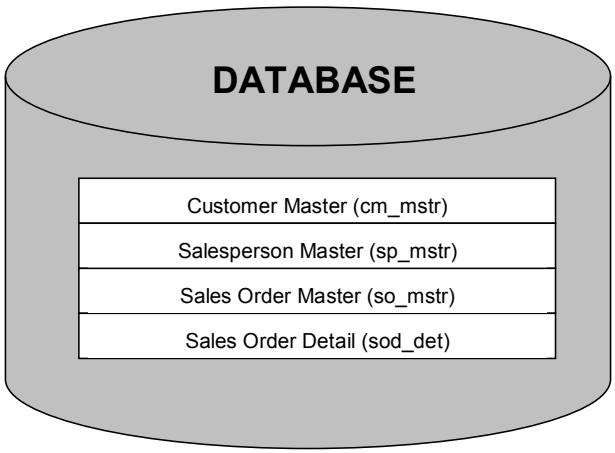
Recent Progress Versions

- 8.2: First 32-bit version of Progress (ADM 1)
- 8.3: Marketing version of 8.2D (ADM 1)
- 9.0: First implementation of ADM 2
  - Separated the UI from the business logic
  - Many new features added
- 9.1B: significant WebClient is added
- 10.0: introduction of OpenEdge Studio
- 10.1B: support for 64-bit data formats
- 10.1C: current version, supports IP v6

Recent Progress Versions

QAD
Our Passion. Your Advantage.

What is a Relational Database ?

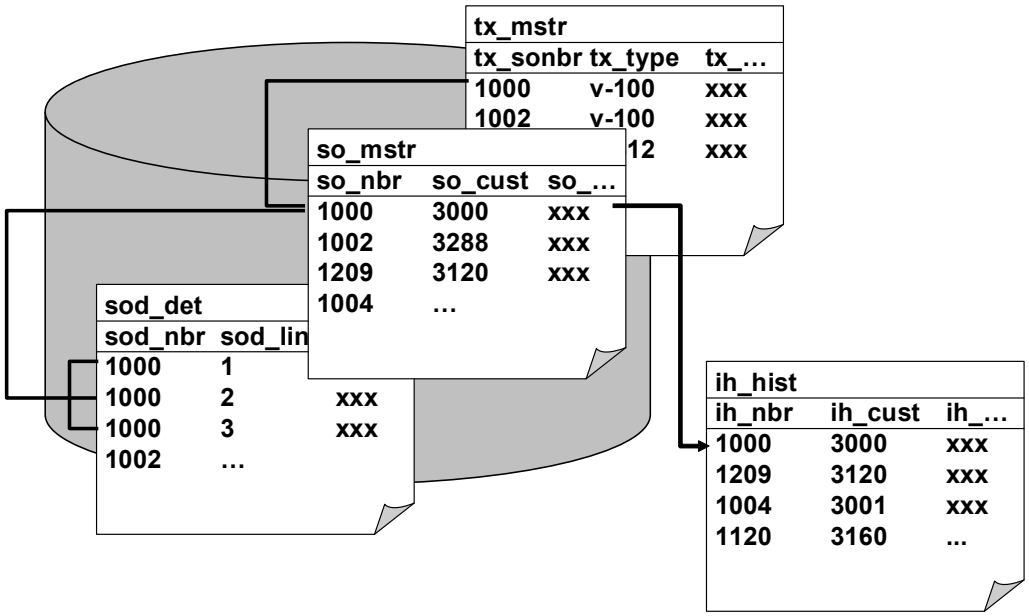


QAD Proprietary 71

What is a Relationship Database?

QAD
Our Passion. Your Advantage.

What is a Relational Database ?



QAD
Our Passion. Your Advantage.

What is a Database File ?

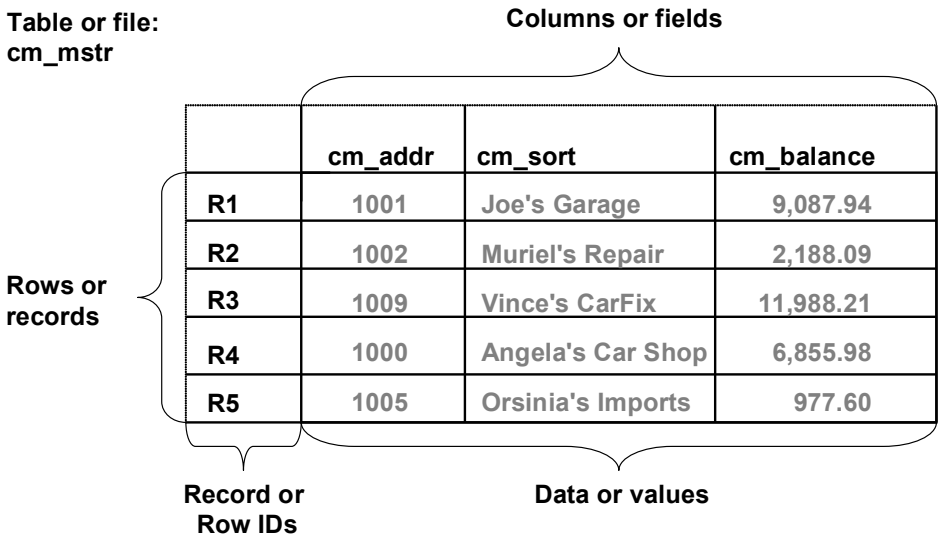
cm_mstr cm_addr	cm_sort	cm_balance
1001	Joe's Garage	9,087.94
1002	Muriel's Repair	2,188.09
1009	Vince's Car Fix	11,988.21
1000	Angela's Car Shop	6,855.98
1005	Orsinia's Imports	977.60

QAD Proprietary 73

What is a Database File?

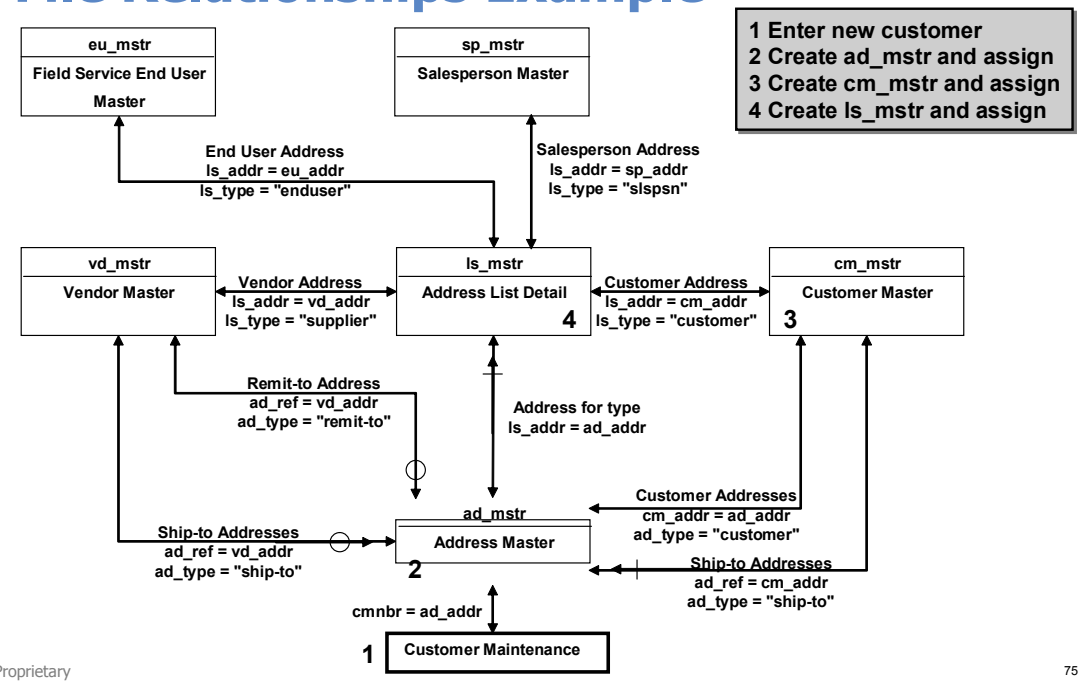
QAD
Our Passion. Your Advantage.

What is a Database File?



QAD
Our Passion. Your Advantage.

File Relationships Example



QAD Proprietary 75

File Relationships Example

QAD
Our Passion. Your Advantage.

Indexing

- Fast Retrieval
  - satisfy search criteria without opening record
  - automatic record ordering
  - may contain multiple fields (compound index)
  - MUST BE ACCESSED IN FIELD ORDER
- Overhead
  - when a record is created, an index entry is created
  - when a record is deleted, an index entry is deleted
  - when an indexed field is updated, the index is updated
  - DO NOT CREATE INDEXES NEEDLESSLY

Indexing

QAD
Our Passion. Your Advantage.

Compound Indexes

- Get the best "bracket" you can
  - See PROGRESS Database Design Guide, section 4

~~•Index 1~~
~~•Field - cm_addr~~
~~•Field - cm_sort~~
~~•Field - cm_xslspsn~~

Index 1
- Field - cm_addr

Index 2
- Field - cm_sort

Index 3
- Field - cm_sort
- Field - cm_xslspsn

QAD Proprietary 77

Compound Indexes

QAD
Our Passion. Your Advantage.

Qualifying Field and Table Names

- When to use more than just the field name ?
  - Filename.Fieldname
    - Used when the field name is shared among other files, buffers, workfiles, work-tables, or temp-tables
  - Database.Filename.Fieldname
    - Used when the file name is shared among other connected databases

QAD Proprietary 78

Qualifying Field and Table Names

QAD
Our Passion. Your Advantage.

Data Types

Data Type	Description
Character	contains any type of data (letters, numbers, and special characters)
Integer	holds whole numbers, positive or negative, ranging from -2,147,483,648 to 2,147,483,647
Decimal	contains decimal numbers up to 50 digits in length, including up to ten places to the right of the decimal
Date	holds dates ranging from 1/1/32768 B.C. to 12/31/32767 A.D. (Enterprise Edition 's hi_date = 12/31/3999)
Logical	values that evaluate to a Yes/No or True/False

Data Types

QAD
Our Passion. Your Advantage.

Default Initial Values

Data Type	Initial Value
Character	null string
Integer	0
Decimal	0
Date ?	(unknown value)
Logical	no/false
RECID ?	(unknown value)

- Null strings and unknown values display as blanks

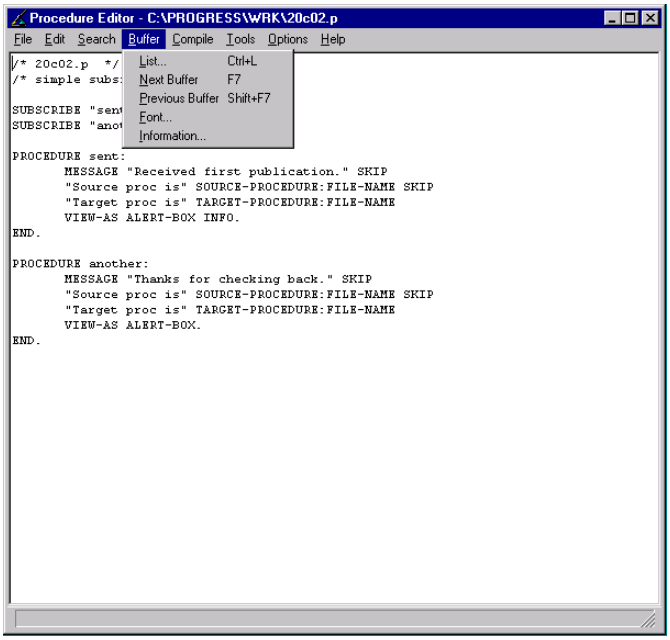
QAD Proprietary 80

Default Initial Values

QAD
Our Passion. Your Advantage.

Progress Editor

- File Management
- Cut & Paste
- Searching
- Multiple buffers
- Compiles & debugging
- Data dictionary
- 4GL Help & messages help



QAD Proprietary 81

Progress Editor

QAD
Our Passion. Your Advantage.

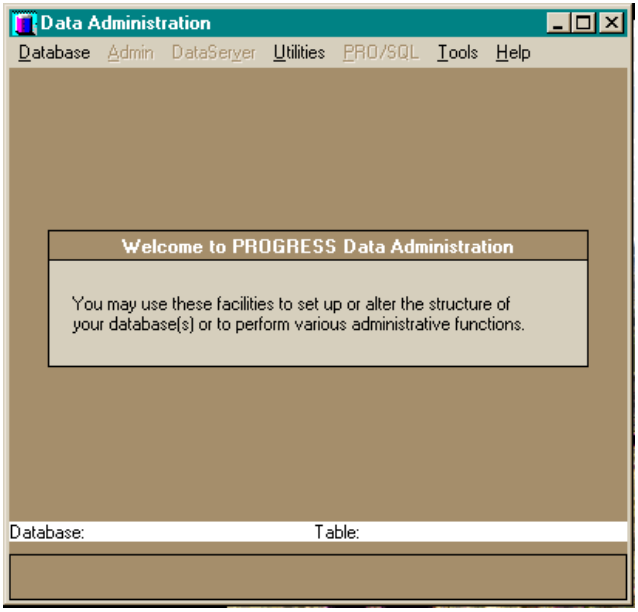
Progress Key Functions

Key	Function	Key	Function
F1	HELP	F8	CLOSE
F2	GO	CTRL+F8	FIND
SH+F2	Check Syntax	F9	FIND NEXT
F3	File Open	SH+F9	FIND PREV.
SH+F3	New Proc. File	CTRL+R	REPLACE
CTRL+F3	New Buffer	CTRL+Z	UNDO
SH+F4	Debug	CTRL+X	CUT
F6	SAVE	CTRL+C	COPY
SH+F6	SAVE AS	CTRL+V	PASTE
F7	Next Buffer	CTRL+G	Go To Line
SH+F7	Prev. Buffer	CTRL+E	Compiler Messages

Progress Key Functions

QAD
Our Passion. Your Advantage.

The Data Administration Tool

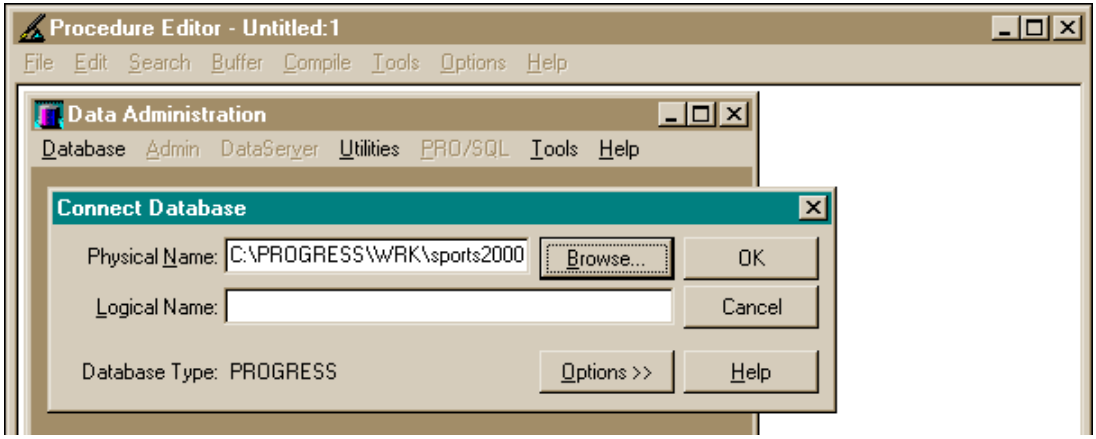


QAD Proprietary 83

Data Administration Tool

QAD
Our Passion. Your Advantage.

Connecting to a Database

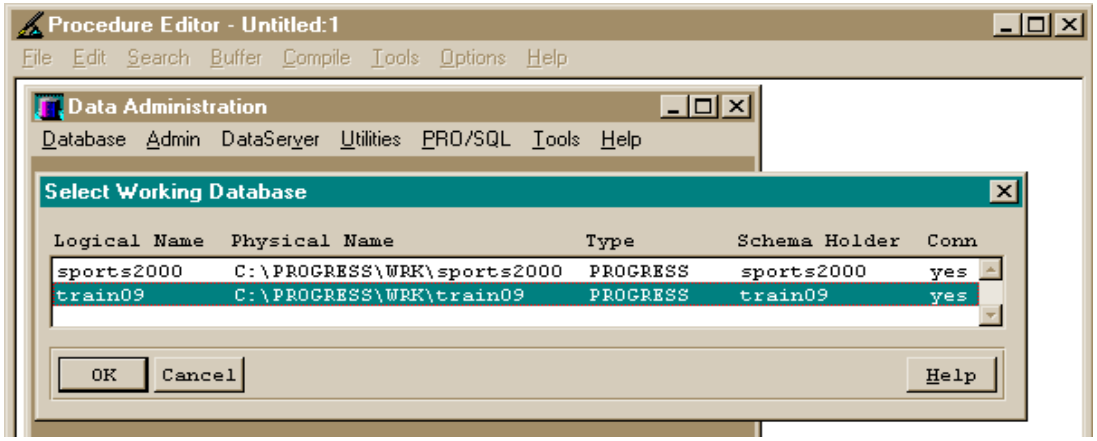


QAD Proprietary 84

Connecting to a Database

QAD
Our Passion. Your Advantage.

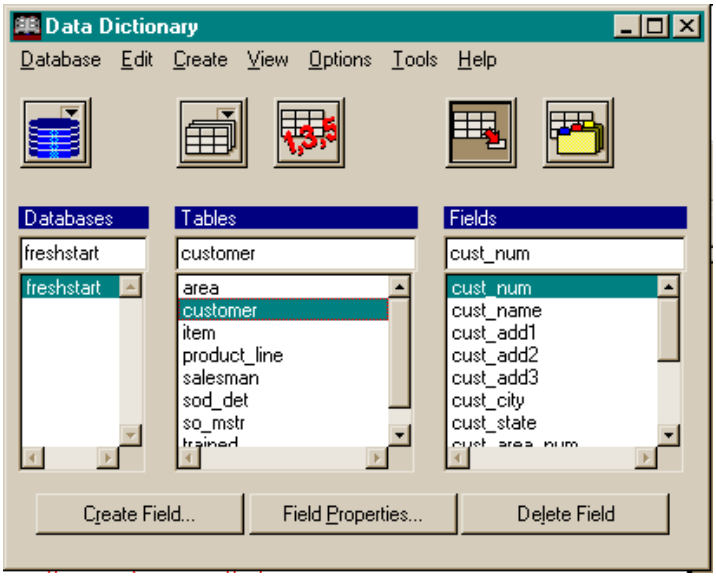
Selecting a Working Database



Selecting a Working Database

QAD
Our Passion. Your Advantage.

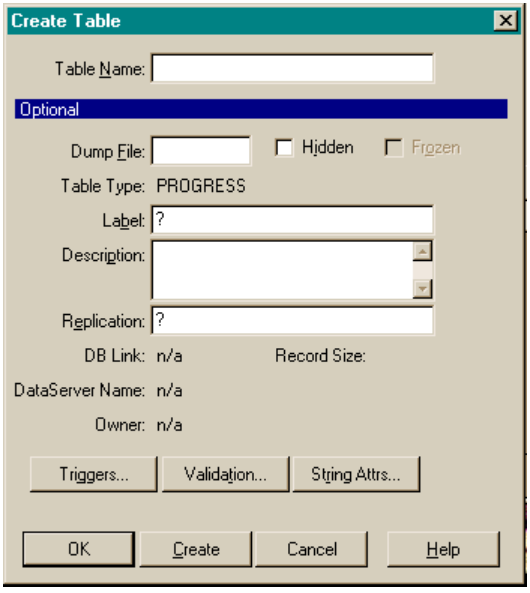
Data Dictionary



Data Dictionary

QAD
Our Passion. Your Advantage.

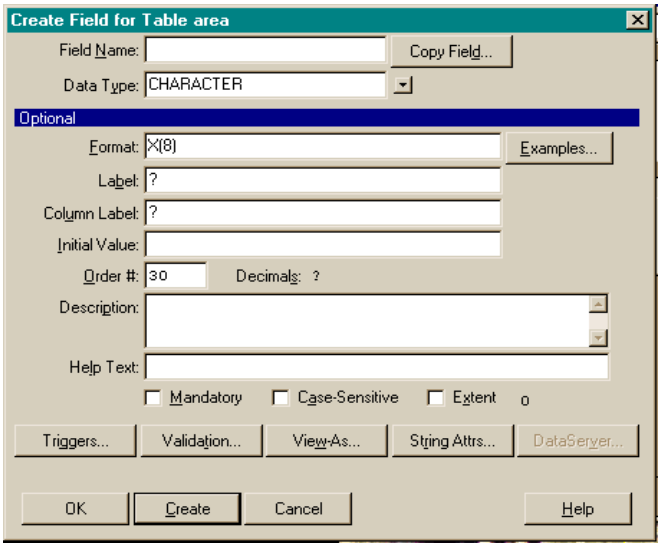
Create a Table



Creating a Table

QAD
Our Passion. Your Advantage.

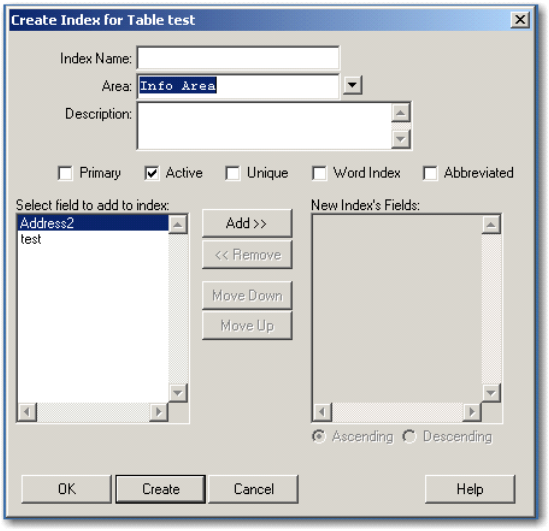
Create a Field



Create a Field

QAD
Our Passion. Your Advantage.

Create an Index



QAD Proprietary 89

Create a Index

QAD
Our Passion. Your Advantage.

Sequences

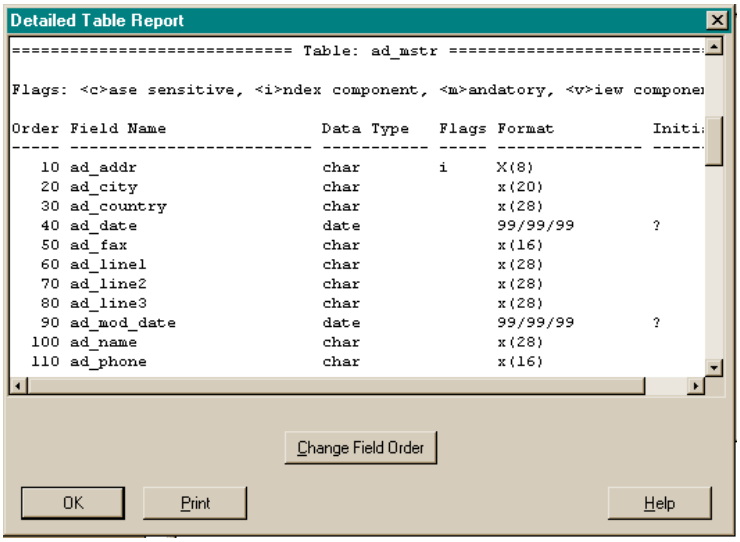
- Database objects that provide incremental values to an application
- Can generate sequential values within any range of a PROGRESS integer with your choice of increment (positive or negative)
  - nonterminating - provides continual unique values
  - terminating - terminates after it reaches a specified limit
  - cycling - restarts after it reaches a specified limit
- Limit of 250 to 2000 per database*
- Not continuous, allows for gaps
- Fast access, no lock waits
- * depends on db block size (2K to 8K)

Sequences

QAD
Our Passion. Your Advantage.

Dictionary Reports

- Detailed Table
- Quick Table
- Quick Field
- Quick Index
- PRO/SQL View
- Sequence
- Trigger
- User
- Table Relations
- Storage Areas



Dictionary Reports

QAD
Our Passion. Your Advantage.

Dumping/Loading Data and Defs

```
ADD SEQUENCE "cust_num_seq"
  INITIAL 3003
  INCREMENT 1
  CYCLE-ON-LIMIT no
  MIN-VAL 3003

ADD TABLE "cm_mstr"
  AREA "Schema Area"
  DESCRIPTION "Customer Master File"
  DUMP-NAME "cm_mstr"
  TABLE-TRIGGER "CREATE" NO-OVERRIDE PROCEDURE "crecust" CRC "29598"
  TABLE-TRIGGER "DELETE" NO-OVERRIDE PROCEDURE "delcust" CRC "19521"
  TABLE-TRIGGER "WRITE" NO-OVERRIDE PROCEDURE "custwrite" CRC "40584"

ADD FIELD "cm_addr" OF "cm_mstr" AS character
  FORMAT "X(8)"
  INITIAL ""
  LABEL "Customer"
  POSITION 2
  COLUMN-LABEL "Cust"
  ORDER 10
```

- Dump/Load:
  - Data Definitions
  - Table Contents
  - SQL Views
  - User Table Contents
  - Current Sequence
  - Values etc.

Partial Data Definitions (.df) Dump

QAD Proprietary 92

Dumping/Loading Data and Defs

QAD
Our Passion. Your Advantage.

Lab 1

1. Launch the Data Dictionary
2. Connect to the reachv9 database
3. Add the `cm_mstr` table, fields and indexes as shown on the next page

All names are important for future dependencies

4. In the Data Administration tool, use the Admin menu to load the data for the table (`cm_mstr.d`)

QAD Proprietary 93

QAD
Our Passion. Your Advantage.

```
Table: customer
Order Field Name                 Data Type   Flags Format          Initial
----- -------------------------- ----------- ----- --------------- ----------
   10 cm_addr                    char        i     X(8)
   20 cm_balance                 deci-2            ->>,>>>,>>>,>>9 0
   30 cm_class                   char              X(8)
   40 cm_cr_hold                 logi              yes/no          no
   50 cm_cr_limit                inte              >>>,>>>,>>>,>>9 0
   60 cm_cr_rating               char              X(8)
   70 cm_cr_terms                char              X(8)
   80 cm_sort                    char        i     x(28)
   90 cm_xslspsn                 char[2]           X(8)

Field Name                     Label                  Column Label
------------------------------ ---------------------- ----------------------
cm_addr                        Customer               Cust
cm_balance                     Balance                Bal
cm_class                       Class                  Class
cm_cr_hold                     Credit Hold            Hold
cm_cr_limit                    Credit Limit           Cr Limit
cm_cr_rating                   Credit Rating          Cr Rate
cm_cr_terms                    Terms                  Terms
cm_sort                        Sort Name              ?
cm_xslspsn                     Salesperson            ?

============================== INDEX SUMMARY ==============================
============================== Table: cm_mstr ==============================
Flags: <p>rimary, <u>nique, <w>ord, <a>bbreviated, <i>nactive, + asc, - desc

Flags Index Name                       Cnt Field Name
----- -------------------------------- --- ----------------------------------
pu    i_cm_addr                          1 + cm_addr

      i_cm_sort                          1 + cm_sort

** Index Name: i_cm_addr
 Storage Area: N/A
** Index Name: i_cm_sort
 Storage Area: N/A
```

SECTION 2

Procedural Coding

CHAPTER 2

Working with Records

QAD
Our Passion. Your Advantage.

Syntax Elements

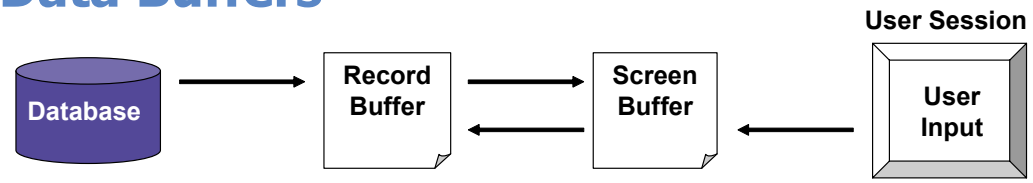
- **statement** one complete instruction to Progress
- **phrase** unit within a statement that alters or refines processing of the statement
- **variable** a variable holds data in memory
- **operator** language element used to manipulate, compare or test values in an expression
- **expression** constant, field or variable name, function, or combination of these and an operator
- **procedure** source file containing Progress 4GL code
- **block** sequence of statements treated as a unit
- **function** performs a discreet task within expressions or statements, usually returns a value

QAD Proprietary 9

Syntax Elements

QAD
Our Passion. Your Advantage.

Data Buffers

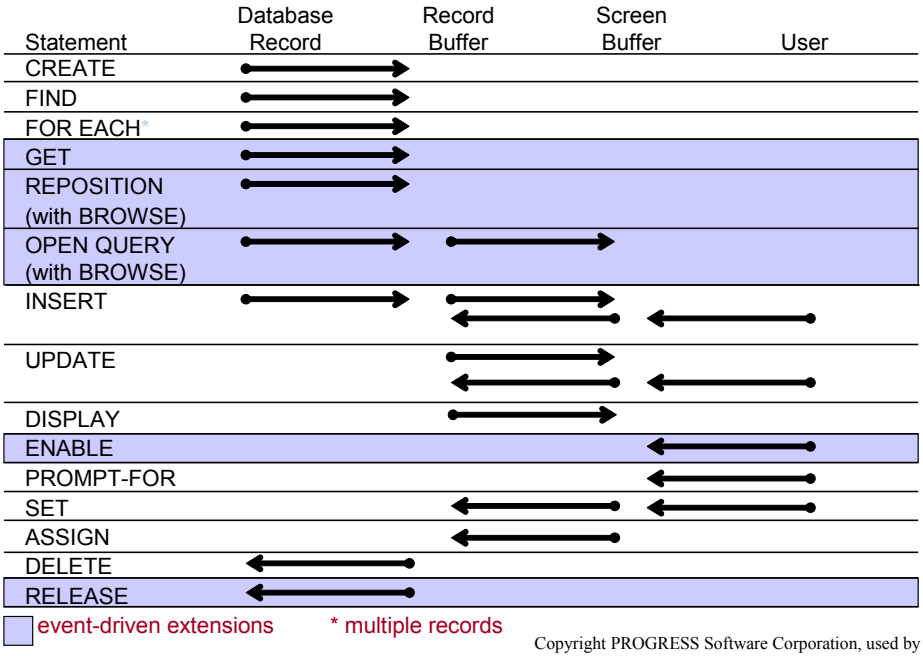


- The only connection between users is the database
- The data is protected through the 4GL:
  - All data modifications are first moved to the record buffer
  - The screen buffer allows users to see it
  - User input allows user modifications
  - All changes are written to the record buffer
- Progress manages database writes

Data Buffers

QAD
Our Passion. Your Advantage.

Data Movement



Data Movement

QAD
Our Passion. Your Advantage.

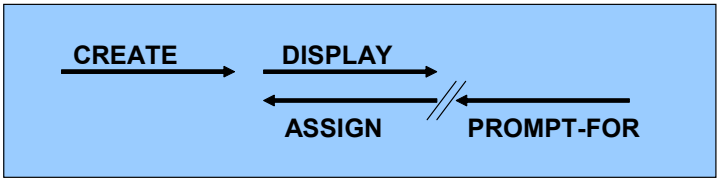
02-001.p

INSERT Statement

- Statement that performs entire record creation and update task
  - top labels
  - all fields are displayed in Data Dictionary 'order'
  - if no field label is defined, field name is used
  - PROGRESS subscripts arrays

```
INSERT pt_mstr.
```

INSERT =



QAD Proprietary 12

INSERT Statement

QAD
Our Passion. Your Advantage.

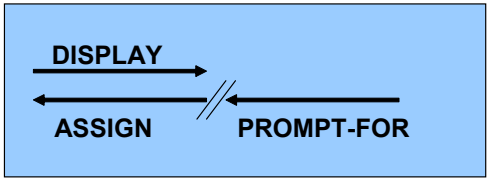
02-002.p

CREATE & UPDATE

- CREATE creates a record in a file, sets fields to default initial values, and moves a copy of the record to the record buffer
- UPDATE displays fields or variables, requests input, and then puts the input data in both the screen buffer and in the specified fields or variables
  - UPDATE controls the TAB sequence among fields

```
CREATE pt_mstr.
UPDATE pt_mstr.
```

UPDATE =



QAD Proprietary

13

CREATE and UPDATE

QAD
Our Passion. Your Advantage.

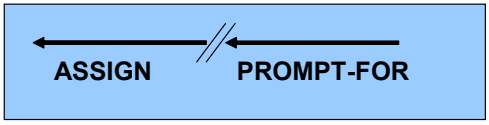
02-003.p

DISPLAY & SET

- DISPLAY moves data to the screen buffer and displays it on the screen or other output destination
- SET requests input, and then puts the data in the screen buffer frame and in the specified fields or variables

```
CREATE pt_mstr.
DISPLAY pt_mstr.
SET pt_mstr.
```

SET =



QAD Proprietary 14

DISPLAY AND SET

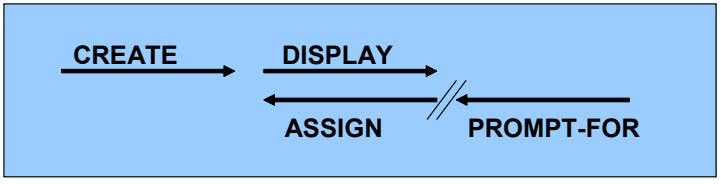
QAD
Our Passion. Your Advantage.

02-004.p

PROMPT-FOR & ASSIGN

- PROMPT-FOR makes the fields or the record available for input in the screen buffer
- ASSIGN moves data from the screen buffer to the record buffer

```
CREATE pt_mstr.
DISPLAY pt_mstr.
PROMPT-FOR pt_mstr.
ASSIGN pt_mstr.
```



QAD Proprietary 15

PROMPT-FOR and ASSIGN

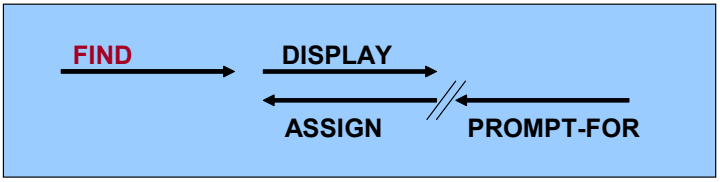
QAD
Our Passion. Your Advantage.

02-005.p

FIND Statement

- Locates a single record in a table and moves that record into a record buffer
- Raises an error if statement fails
- Optional: FIRST/LAST/NEXT/PREV

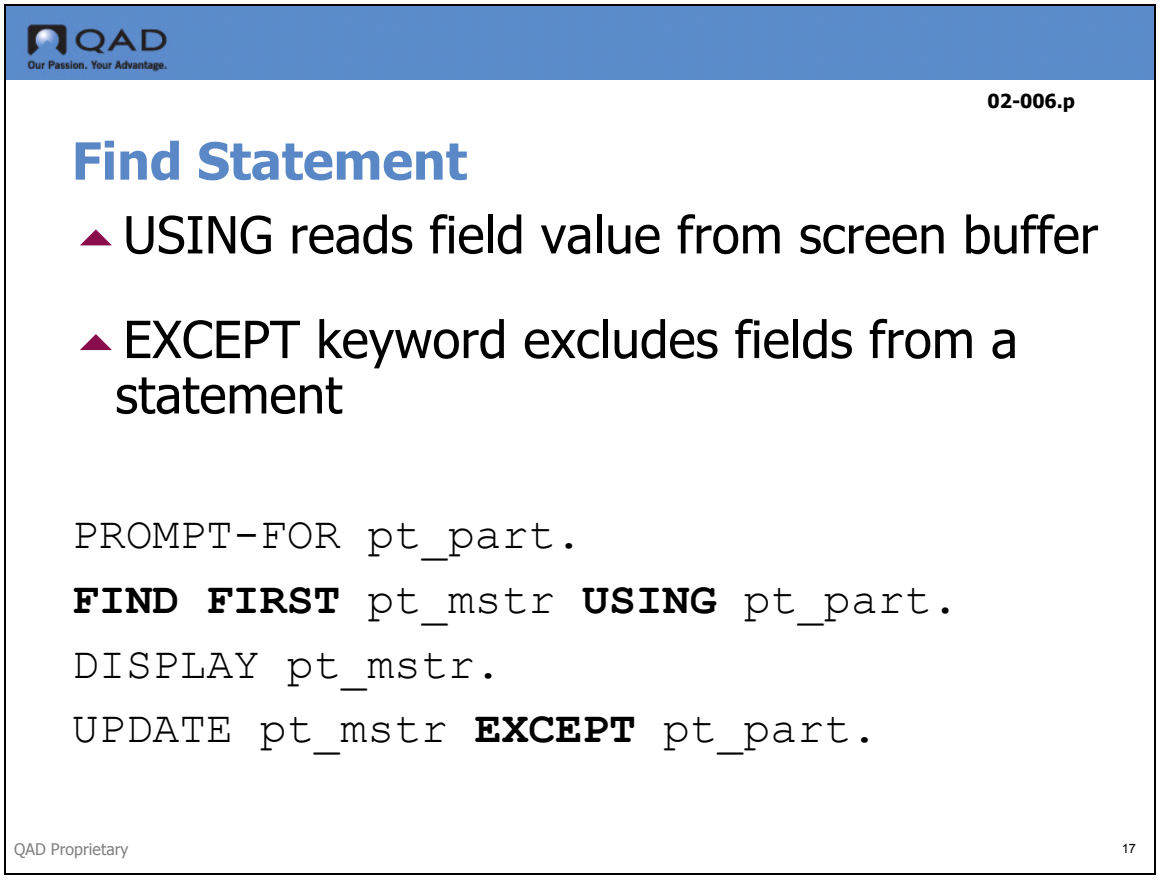
```
FIND FIRST pt_mstr.
UPDATE pt_mstr.
```



QAD Proprietary 16

FIND Statement

FIND must locate one and only one unique record. If the FIND statement finds two or more records that match the selection criteria, then the FIND actually fails.



FIND STATEMENT

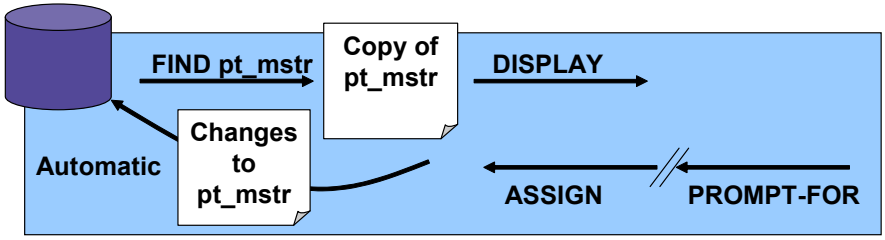
USING command can only be performed with screen input value.

The EXCEPT statement is used to update/display all fields EXCEPT the one(s) listed.

QAD
Our Passion. Your Advantage.

Record Buffers

- Every time Progress opens or creates a record, it creates a buffer in memory with the same name as the record created or opened
- Control the size of the buffer area with the startup parameter -Mr n where n is the size in bytes (default is 1012, max is 3200)
- Buffers can be named, copied, compared



QAD Proprietary 18

Record Buffers

If naming a buffer, must be explicitly called out in procedure.

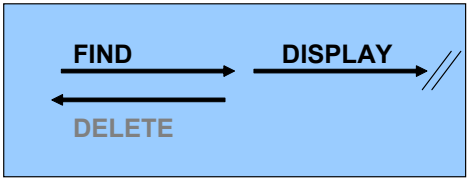
QAD
Our Passion. Your Advantage.

02-007.p

DELETE Statement

- Removes a record from a record buffer and from the database

```
FIND LAST pt_mstr.
DISPLAY pt_mstr.
DELETE pt_mstr.
```



QAD Proprietary 19

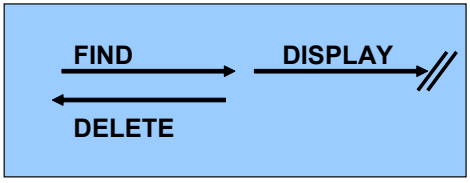
DELETE Statement

02-008.p

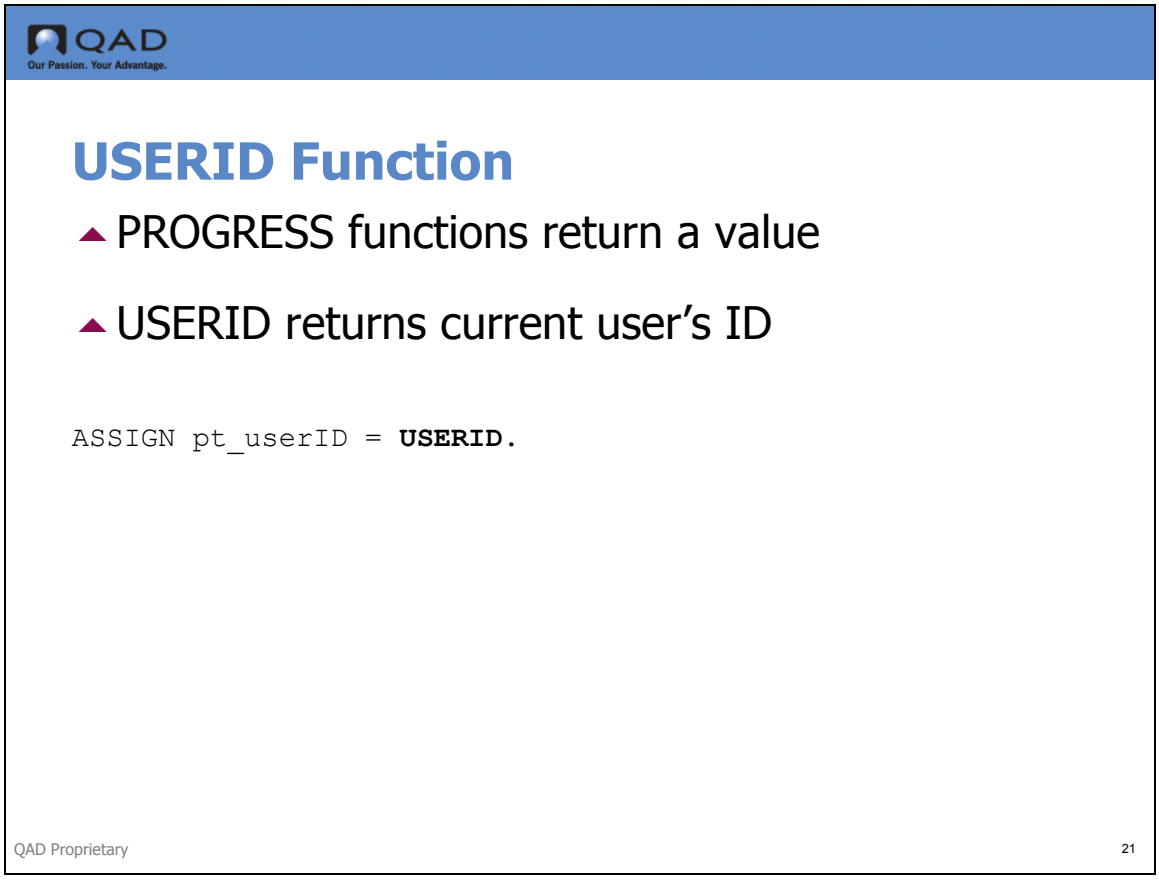
PAUSE Statement

- Suspends processing indefinitely, or for specified number of seconds, or until user presses any key
- Method to debug, or separate statements in processing

```
FIND LAST pt_mstr.
DISPLAY pt_mstr.
PAUSE.
DELETE pt_mstr.
```



PAUSE Statement



USERID Function

The USERID value is the user id that was logged into PROGRESS.

QAD
Our Passion. Your Advantage.

LAB 2

1. Add a part master record. Show the user any initial values. Display the item number first. (L0201.p)

2. Prompt the user for a part number, then find the correct record with the part number. Capture the user's ID in the record and update everything but the part number and user ID. (L0202.p)

3. Allow the user to enter a part number, then find the correct record with the part number and delete the record. (L0203.p)

Lab 2

QAD
Our Passion. Your Advantage.

LAB 2 Review

QAD Proprietary 23

Lab 2 Review

CHAPTER 3

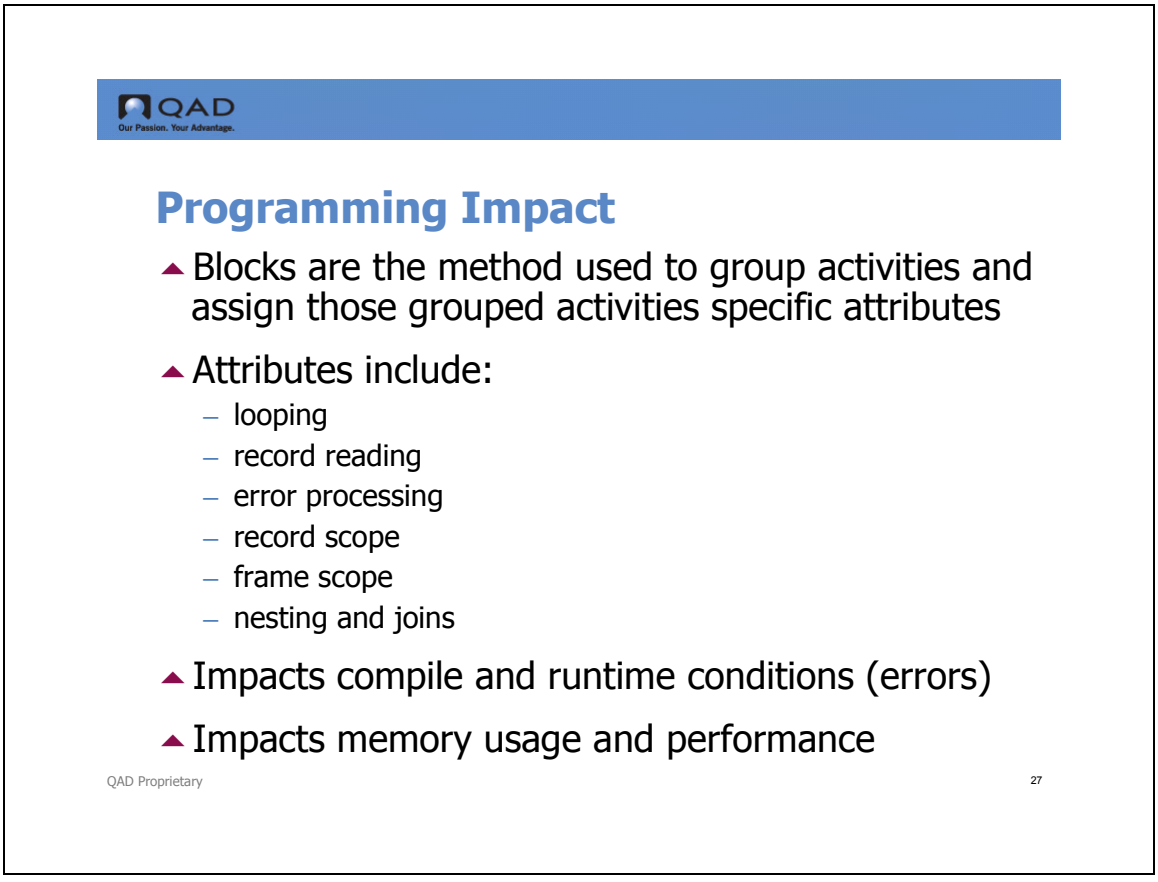
Blocks

QAD
Our Passion. Your Advantage.

Blocks

- Series of statements Progress treats as a single unit
- Each begins with a header statement and ends with an END statement
  - Except for the default block: procedure block
- Blocks in Progress include:
  - DO
  - REPEAT
  - FOR
  - Trigger
  - Procedure
  - Internal Procedure
  - User-Defined Function

Blocks

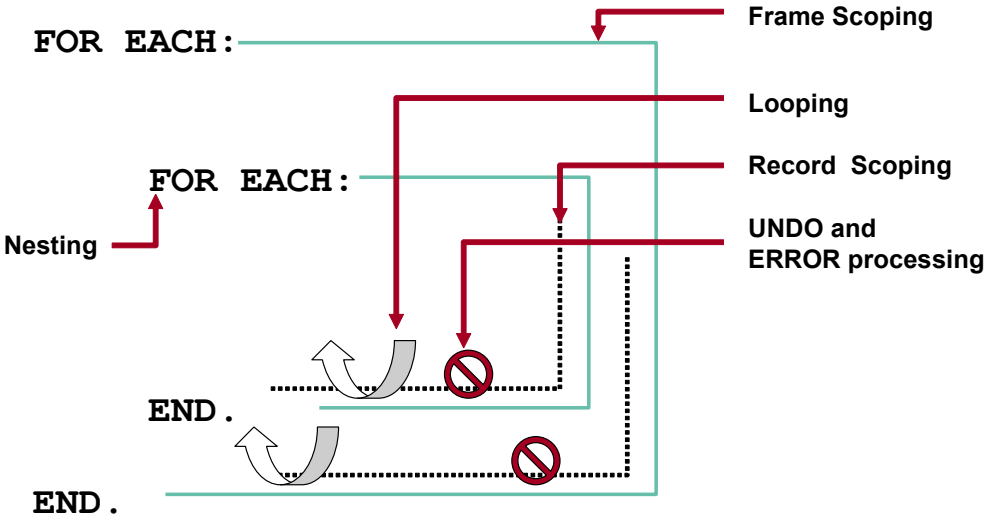


Programming Impact

- Record Scope: The time that the record is active in the record buffer
- Frame Scope: The outer most block in which a frame is referenced.
- Transaction Scope: The unit of work that is done and undone as a whole.

QAD
Our Passion. Your Advantage.

How Blocks Work



QAD Proprietary 28

How Blocks Work

QAD
Our Passion. Your Advantage.

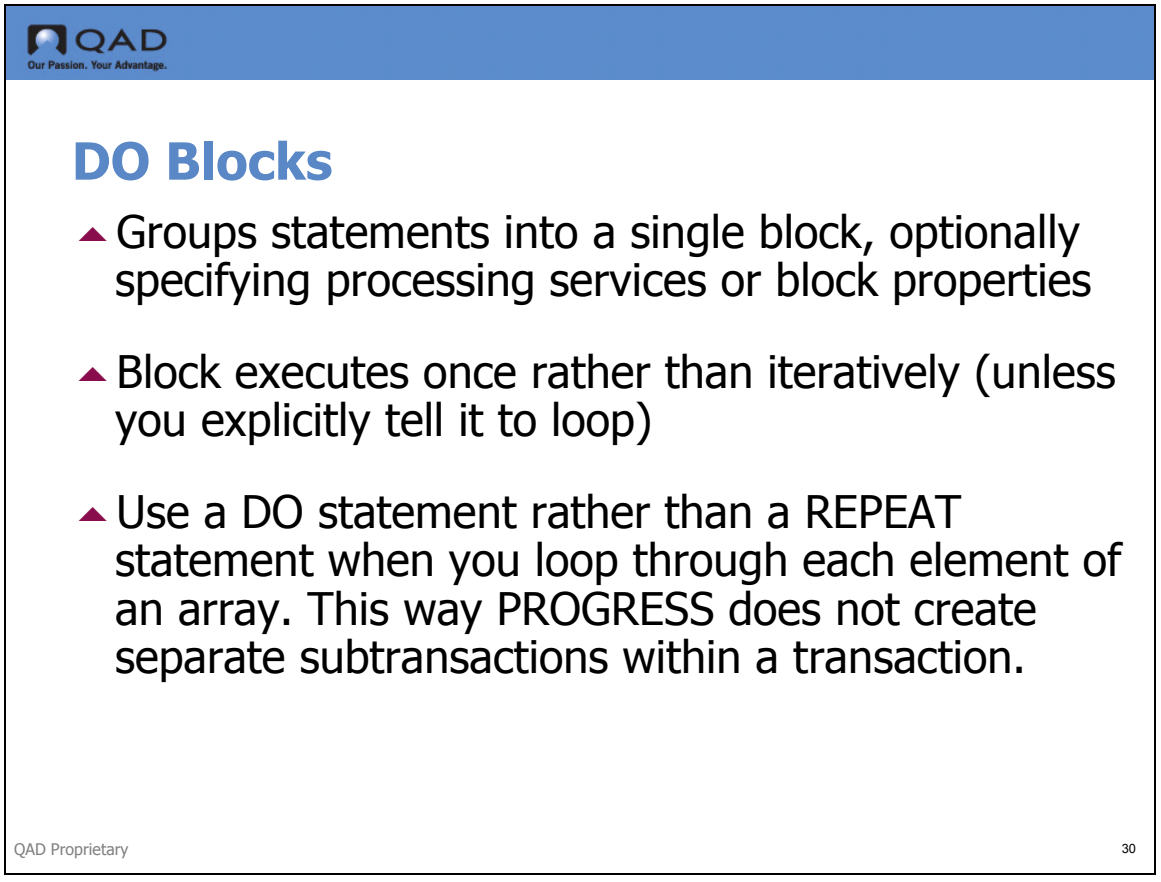
Block Properties

Property	FOR		REPEAT		DO		Procedure or Trigger	
	Implicit	Explicit	Implicit	Explicit	Implicit	Explicit	Implicit	Explicit
Looping	YES	WHILE TO/BY	YES	WHILE TO/BY	NO	WHILE TO/BY	NO	NO
Record reading	YES	Record Phrase	NO	NO	NO	NO	NO	NO
Frame scoping	YES	WITH FRAME	YES	WITH FRAME	NO	WITH FRAME	YES	NO
Record scoping	YES	NO	YES	FOR	NO	FOR	YES	NO
UNDO	YES	TRANS-ACTION ON ERROR	YES	TRANS-ACTION ON ERROR	NO	TRANS-ACTION ON ERROR	YES	NO
ERROR processing	YES	ON ERROR	YES	ON ERROR	NO	ON ERROR	YES	NO

 29

Block Properties

- SCOPE is one of the properties of all blocks. It is the duration that a resource is available to an application. Blocks determine the scope of that resource availability.
- FOR – do for all
- REPEAT – do until I tell you not to
- DO – only what I say and how often I say
- Trigger or Procedure – Just do it! When called.



DO Blocks

Block executes once and creates a transaction. If use WHILE (DO WHILE) then the block will loop as long as the condition is true.

DO statements for arrayed fields will not cause subtransactions. If a REPEAT loop is used, then each of the arrayed fields will initiate a subtransaction within the main transaction.

QAD
Our Passion. Your Advantage.

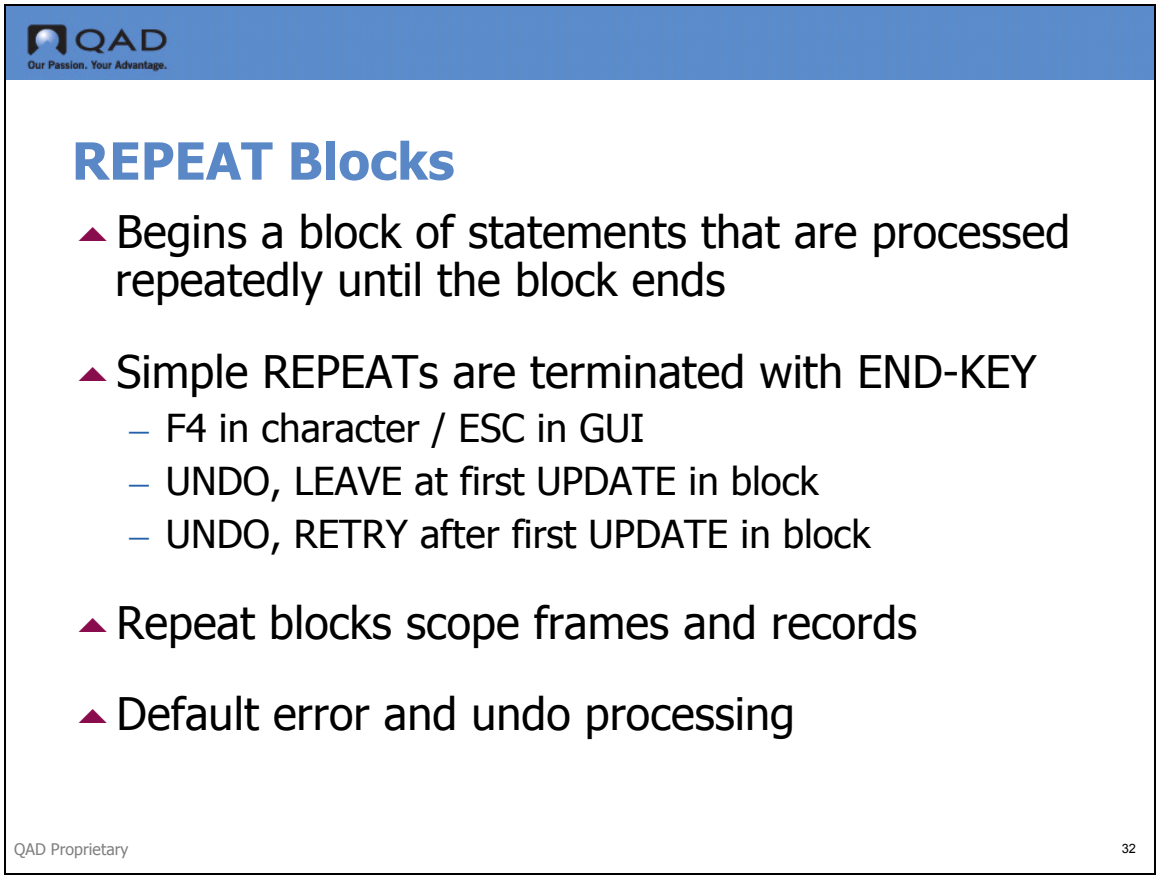
02-009.p – 02-010.p

DO Blocks

- DO:
  - Includes the DO WHILE and DO i = 1 to 100 blocks.
- DO FOR <buffername>:
  - scopes a record
- DO TRANSACTION:
  - scopes a transaction
- DO ON ERROR <action>:
  - scopes a transaction
- DO ON ENDKEY <action>:
  - scopes a transaction
- DO WITH <framename>:
  - scopes a frame

DO Blocks

- DO, DO WHILE and DO I = : Do while a condition is true, start a transaction
- DO FOR <buffername>: Iterates through the buffer until buffer end.
- Scopes a record, meaning that the function performed to that one record is a single transaction.
- DO TRANSACTION: Scopes the entire block to a transaction. All included functions are scoped to this transaction.
- DO ON ERROR: When an error is encountered, perform an action.
- DO ON ENDKEY: If exit key read (F4 or ESC) then perform an action
- DO WITH FRAME: A frame must be explicitly scoped to a DO transaction type.



REPEAT Blocks

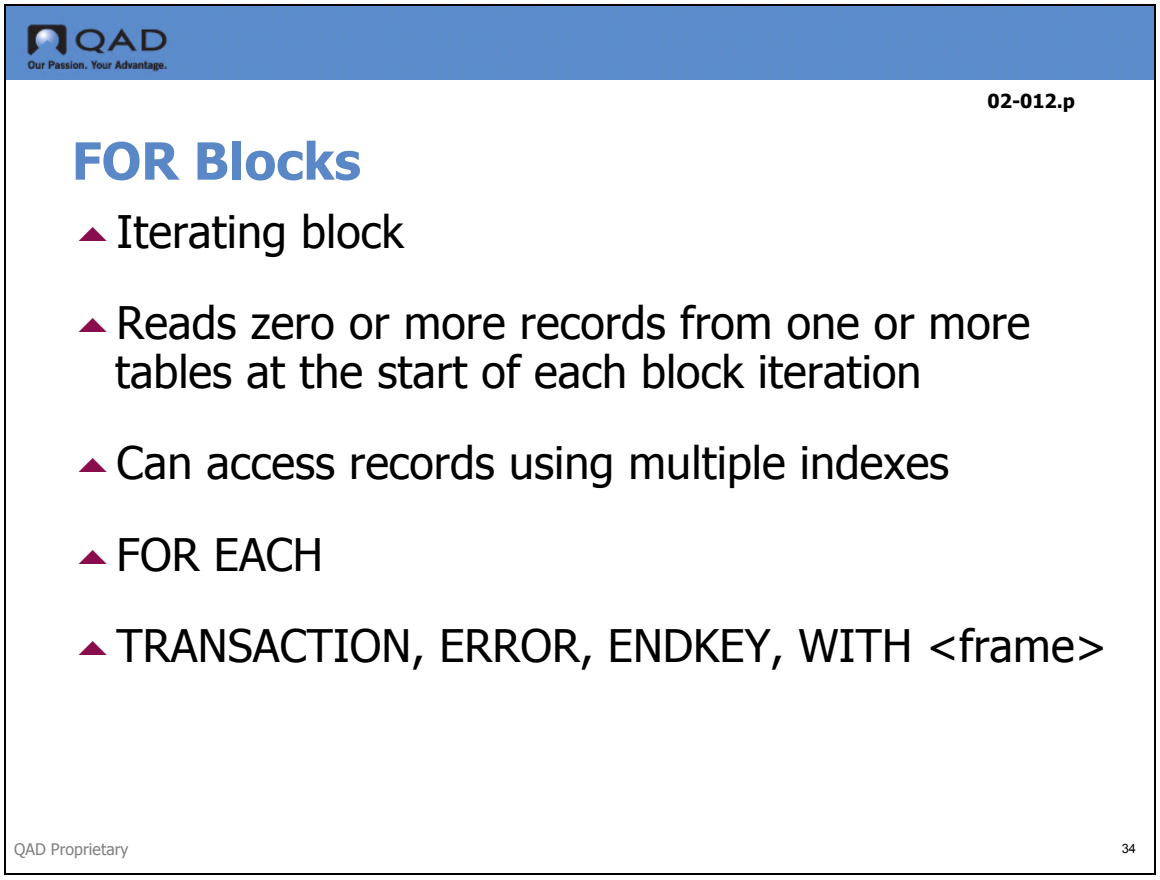
If ERROR or END are not specified, default exit behavior is employed.

QAD
Our Passion. Your Advantage.

02-011.p

REPEAT Blocks

- REPEAT:
- REPEAT FOR <buffername>:
  - scopes a record
- REPEAT TRANSACTION:
  - scopes a transaction
- REPEAT ON ERROR <action>:
  - scopes a transaction
- REPEAT ON ENDKEY <action>:
  - scopes a transaction
- REPEAT WITH <framename>:
  - scopes a frame


QAD
Our Passion. Your Advantage.
02-012.p
FOR Blocks
Iterating block
Reads zero or more records from one or more tables at the start of each block iteration
Can access records using multiple indexes
FOR EACH
TRANSACTION, ERROR, ENDKEY, WITH <frame>
QAD Proprietary
34

FOR Blocks

QAD
Our Passion. Your Advantage.

Nested Blocks v. Joins

- Methods for accessing records from multiple tables in a single statement or block
- Tables are accessed in a parent-child manner
- Nested blocks (outer joins)
  - Displays every parent record and each child record that relates to that parent
- Inner joins
  - Displays only parents that have related child records

Nested Blocks vs. Joins

- Outer Join: Show all data, with relationship as available
- Inner Join: Only show data where relationship is available

QAD
Our Passion. Your Advantage.

02-013.p

Nested Blocks

- Outer join in a 4GL procedure
- Nested FOR EACH statements
  - the outer FOR EACH statement executes once, with one iteration per matching record
  - the inner FOR EACH statement executes n times
    - where *n* is the number of records returned by the outer FOR EACH

QAD Proprietary 36

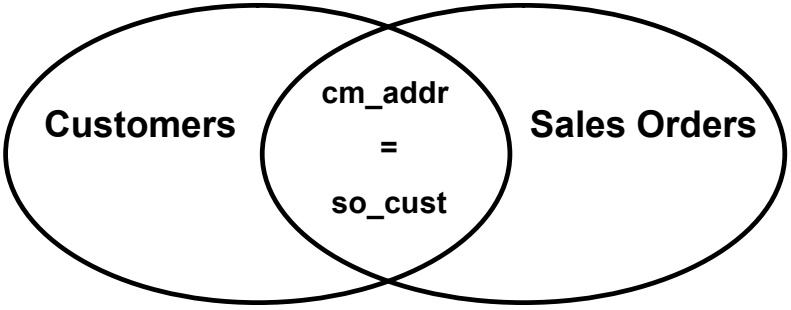
Nested Blocks

QAD
Our Passion. Your Advantage.

02-014.p

Inner Joins

- A *join* is a binary operation that selects and combines the records from multiple tables so that each result contains a single record from each table
- An *inner join* returns the records selected for the table on the left side combined with the related records selected from the table on the right



Inner Joins

QAD
Our Passion. Your Advantage.

Progress Conditions

Condition	Primary Causes	Default Processing	Override Method
ERROR	RETURN ERROR from a trigger or run procedure, or, a procedure error, e.g., failed FIND	Undo and retry nearest REPEAT, FOR EACH or procedure block	ON ERROR phrase
ENDKEY	User presses ENDKEY or, FIND NEXT/PREV fails, or reach end of input file	Undo and leave nearest REPEAT, FOR EACH or procedure block	ON ENDKEY phrase

UNDO Processing

The value in the variable is not UNDONE if the transaction is backed out. Variables defined as NO-UNDO do not write to the .lbi (Local Before Image) file, therefore they require less resources. Use NO-UNDO when the value of a variable is reset after undo of a transaction, or if the significant value is not required later in the process, or if the transaction is re-processed (ie RETRY)

QAD
Our Passion. Your Advantage.

UNDO Processing

- Used in ON ERROR, ON ENDKEY phrases to:
  - RETRY (retries current or named enclosing block)
  - LEAVE (leaves current or named enclosing block)
  - NEXT (iterates current or named block)
  - RETURN (to calling proc from trigger or run proc)
- Default behavior is to restore variable values to value at start of transaction
- NO-UNDO option on DEFINE statements prevents backing out of variable data in memory
  - Use for incrementing values (for example, tracking number of records created)

NO-ERROR Option

IF AVAILABLE then do something. IF NOT AVAILABLE then do something.

QAD
Our Passion. Your Advantage.

NO-ERROR Option

- Tells PROGRESS not to display error messages for any errors it might encounter
- Possible errors are:
  - not finding a record that satisfies the record-phrase
  - finding more than one record for a unique find
  - finding a record that is locked with the NO-WAIT option on the FIND
- Use the AVAILABLE function to test if FIND found a record.

```
PROMPT-FOR pt_part.
FIND pt_mstr USING pt_part NO-ERROR.
```

QAD Proprietary 40

AVAILABLE Function

QAD
Our Passion. Your Advantage.

AVAILABLE Function

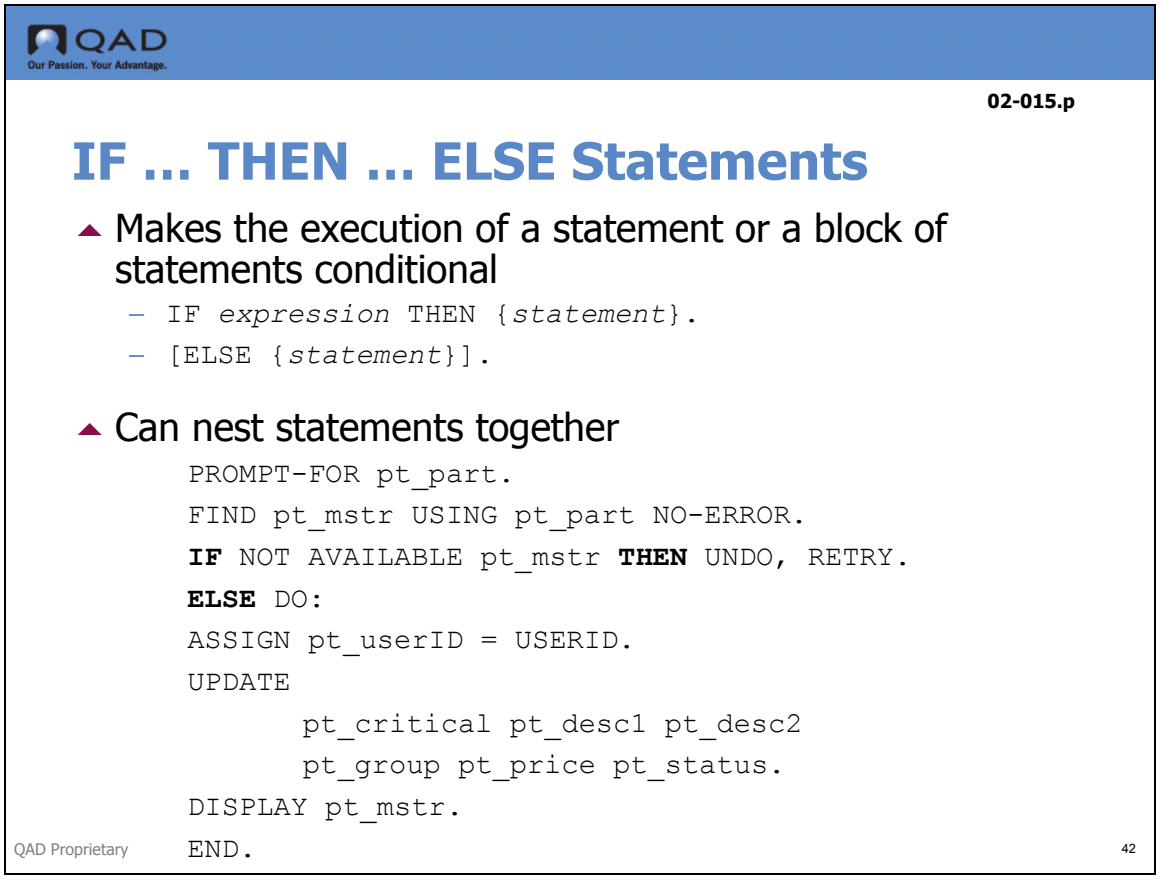
- Returns TRUE if the named record buffer contains a record, FALSE if the record buffer is empty
- FIND or FOR statements read database records into a record buffer with the same name as the file used by the FIND or FOR
- Manage availability with FIND or FOR to avoid error condition

```
IF NOT AVAILABLE cm_mstr THEN UNDO, RETRY.
```

QAD Proprietary 41

IF...THEN...ELSE Statements

If the DO: is not used, then only the next statement is executed on the condition. All other statements that follow are executed unconditionally.



Named Blocks

The name of the block blk1: does not require an END statement to complete the block. The real block starts at the REPEAT: statement and ends at the END. The blk1: statement is just a name for the block.

QAD
Our Passion. Your Advantage.

NAMED BLOCKS

- Allows reference to a specific block other than default block
- Block label precedes block header

```
blk1:
REPEAT:
    PROMPT-FOR pt_part.
    FIND pt_mstr USING pt_part NO-ERROR.
    IF NOT AVAILABLE pt_mstr THEN UNDO blk1, RETRY
blk1.
    UPDATE pt_mstr EXCEPT pt_part.
END.
```


Lab 3

QAD
Our Passion. Your Advantage.

LAB 3

1. Display all item records showing only the part number, description, and price. (L0301.p)
2. Allow users to select an item record for updates. Allow updates to all fields except the userID and part number. Make sure you capture the current user's ID. Handle error and availability conditions. (L0302.p)
3. Add the ability to create a record if it does not exist. (L0303.p)

Lab 3 Review

QAD
Our Passion. Your Advantage.

Lab 3 Review

La

CHAPTER 4

Transactions and Record Scoping

QAD
Our Passion. Your Advantage.

Scoping

- Persistence of a transaction, record, or frame for purposes of access and memory usage
- Impacts:
  - Transactions
    - Controls when changes are committed to the database
  - Records
    - Controls buffer creation/close, helps control locks
  - Frames
    - Controls activation, attributes, field-level display

QAD Proprietary 48

Scoping

QAD
Our Passion. Your Advantage.

Programming Impact

- Scope controls writing of data to the database, record availability, error processing, and user interface behavior
- Closely related to blocks
- Scope can be raised to encompassing blocks or minimized as necessary
- Improper scoping underlies numerous programming errors

Programming Impact

QAD
Our Passion. Your Advantage.

Sample Listing

```
...code\02-015list.p             03/24/02 15:30:52   PROGRESS(R) Page 1
{} Line Blk
-- ---- ---
      1     /* 02-015list.p */
      2
      3     FIND FIRST sp_mstr NO-LOCK.
      4     DISPLAY sp_addr sp_sort.
      5
      6   1 FOR EACH cm_mstr:
      7   1     UPDATE cm_sort cm_balance.
      8     END.
      9
     10     DISPLAY "goodbye" WITH FRAME a.
     File Name      Line Blk. Type Tran            Blk. Label
-------------------- ---- --------- ---- ------------------------------
   -
...code\02-015list.p    0 Procedure   No
    Buffers: reachv9.sp_mstr
    Frames:  a
         Unnamed
...code\02-015list.p    6 For         Yes
    Buffers: reachv9.cm_mstr
    Frames:  Unnamed
```

Buffer and frame scope

Transaction scope

QAD Proprietary

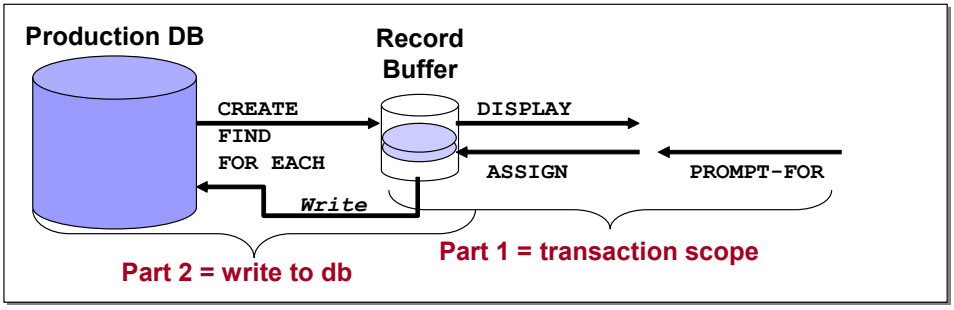
50

Sample Listing

QAD
Our Passion. Your Advantage.

Data Integrity

- Data integrity means that PROGRESS stores completed data and does not store incomplete data in the database
- PROGRESS uses ***transactions*** to automatically handle this processing
- Transactions managed in two parts:



Data Integrity

QAD
Our Passion. Your Advantage.

Transactions

- A transaction is a set of changes to memory variables or database records which the system either completes or discards (leaving no modification to the database) in full
- A transaction controls:
  - when records are written to the database
  - how long a record lock remains after the data is changed
  - how much data will be written or discarded on completion or abandonment of the transaction
  - whether variables changed during the transaction will be restored to their original values if a transaction is undone

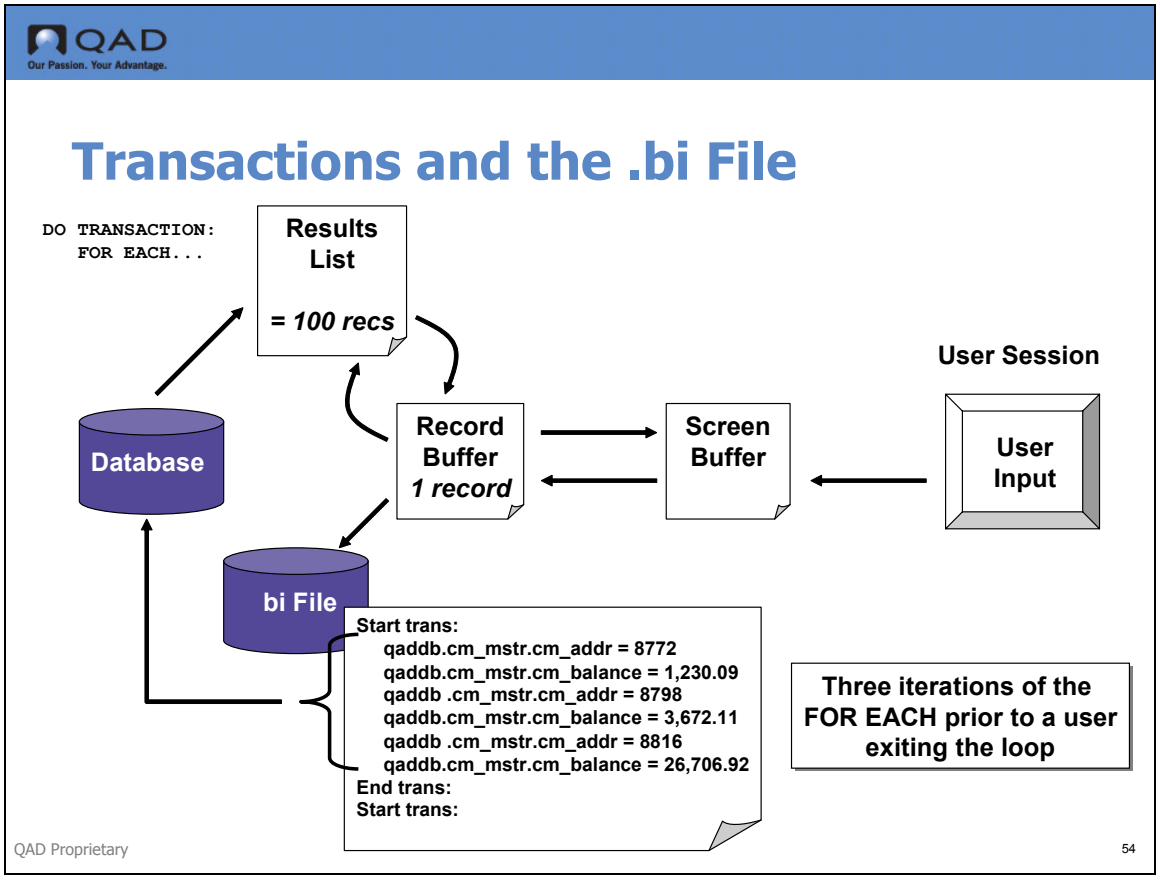
Transactions

QAD
Our Passion. Your Advantage.

Managing Transactions

- Each transaction gets a start note, and if successfully closed, an end note in the Before Image file
- Purpose is to protect the database from partial or corrupt data due to system failures
  - All .BI changes without end notes are backed out on system restart
- Also maintains data integrity, allowing users to undo changes prior to committing them to the database.

Managing Transactions



Transactions and the .bi File

QAD
Our Passion. Your Advantage.

Development Objectives

- Does the program behave identically (commit or rollback data) under all conditions?
  - Normal data processing
  - User exit (ENDKEY) from all entry points
  - Unintended exit, such as power outage or Progress error

Development Objectives

QAD
Our Passion. Your Advantage.

02-017.p – 02-020.p

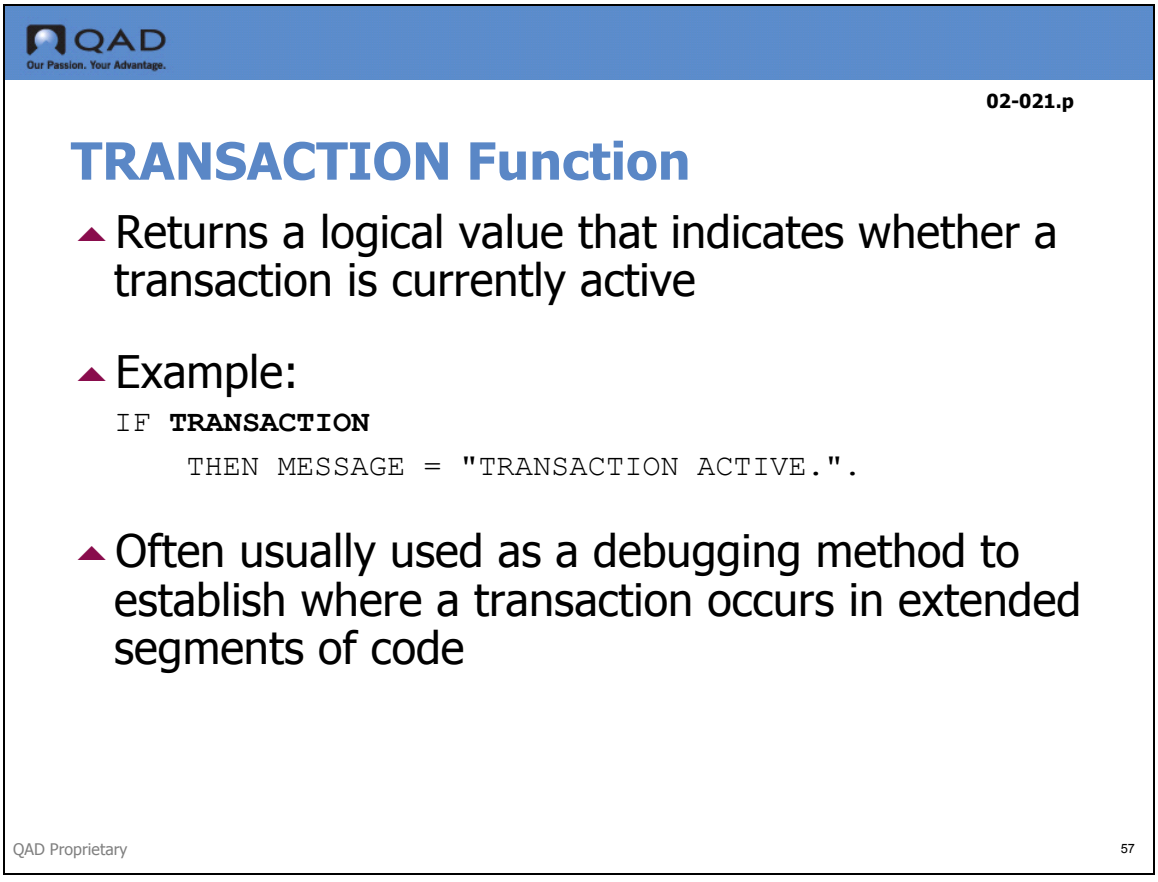
Starting a Transaction

- Only one transaction per session at any time
- Any change to the database
  - Create, Delete, Update, Assign, Insert
- Use of the TRANSACTION keyword in a block header (DO, FOR EACH, or REPEAT)
- Invoking an EXCLUSIVE-LOCK on a FIND or GET
- Don't start transactions unnecessarily:
  - All changes, including variables, are written to disk once a transaction is started
  - All incomplete changes are backed out on start-up or intentional exit

QAD Proprietary 56

Starting a Transaction

- You cannot do multiple simultaneous transactions for a given session.
- Avoid EXCLUSIVE-LOCK on FIND statements if no update is taking place.



TRANSACTION FUNCTION

Used in addition to the LISTING function to determine transaction scope.

QAD
Our Passion. Your Advantage.

Transaction Scope

- Transactions are scoped to:
  - FOR EACH blocks that update the database
  - REPEAT or REPEAT FOR blocks that update the database
  - DO TRANSACTION or other blocks using the TRANSACTION keyword
  - DO ON ERROR or DO ON ENDKEY blocks
- If none of these blocks are present and a change to the database is encountered, scope is raised to the entire procedure

Transaction Scope

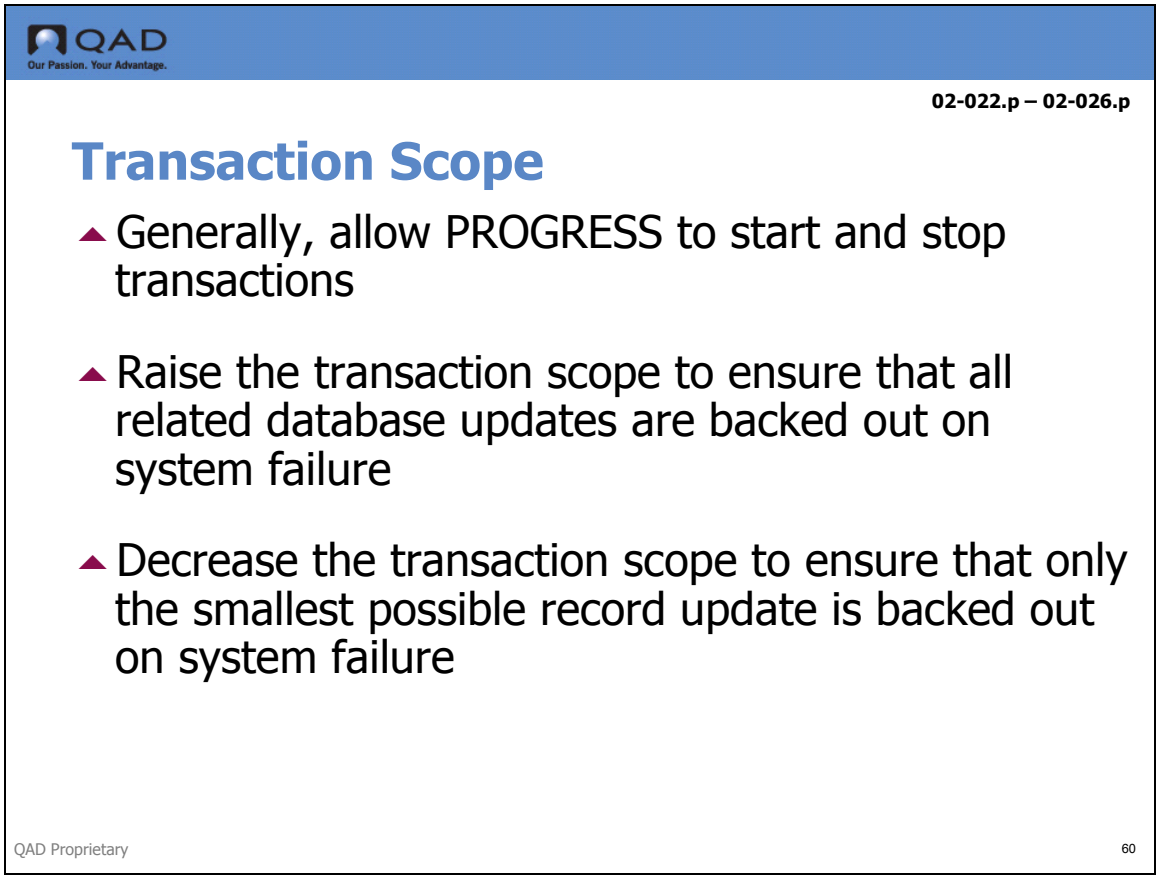
QAD
Our Passion. Your Advantage.

Development Objectives II

- Does the program behave identically (commit or rollback data) under all conditions?
- Does the program scope the transaction as required by the business case?
  1. Commit or rollback parent record as a unit, and commit or rollback all child records as a separate unit?
  2. Commit or rollback parent and children as a single unit?

QAD Proprietary 59

Development Objectives II



Transaction Scope

It depends on application. MRP is very small in scope, as each planned order generated is one transaction. If MRP is terminated, the planned orders planned remain.

If an Invoice is Posting, and the invoice post is terminated, the entire post is undone, indicating very large transaction scope.

The type of application drives the behavior of the transaction scope.

QAD
Our Passion. Your Advantage.

Subtransactions

- A subtransaction is started when a transaction is already active and PROGRESS encounters a *subtransaction* block*:*
  - each iteration of a FOR EACH block nested within a transaction block
  - each iteration of a REPEAT block nested within a transaction block
  - each iteration of a DO ON ERROR, or DO ON ENDKEY block
  - a procedure block that is run from a transaction block in another procedure

Subtransactions

QAD
Our Passion. Your Advantage.

02-027.p – 02-028.p

Subtransactions

- Each subtransaction close gets an end note in the local before image file (LBI)
- Purpose is to allow for incremental control of modifications through undo and error processing and system failures
  - All .LBI changes without end notes are backed out on ENDKEY, ERROR, CTRL-BREAK (failures)
- System failures, and correctly structured transactions, will back out entire transaction and all complete and incomplete subtransactions

QAD Proprietary 62

Allow a smaller scope of control within a larger transaction.

On local terminations (Ctrl-C), the subtransaction is undone. To ensure Data Integrity, the main transaction should then have error processing to continue.

QAD
Our Passion. Your Advantage.

Record Locking

- Record locks apply to individual records
- Three types of locks: NO-LOCK, SHARE-LOCK & EXCLUSIVE-LOCK
- Five lock states (view in PROMON):
  - Locked: record exclusively locked by local process
  - Limbo: record released, but transaction remains open. The record lock is not released until transaction end.
  - Purged: record released and lock freed
  - Queued: record lock requested for a record already held by another process
  - Upgrade: lock upgrade requested (from SHARE to E-LOCK) for a record held by another process

Record Locking

QAD
Our Passion. Your Advantage.

Record Locking

- NO-LOCK
  - Bypasses the normal locking mechanism
    - allows record reading regardless of lock status
    - allows record reading without locking records
- SHARE-LOCK
  - Whenever PROGRESS reads a record (the default lock status), it puts a SHARE-LOCK on that record. Other users can read the record but cannot update it until the procedure releases the SHARE-LOCK.
- EXCLUSIVE-LOCK
  - Whenever PROGRESS updates a record, it puts an EXCLUSIVE-LOCK on that record. Other users cannot read or update that record until the procedure releases the EXCLUSIVE-LOCK (except NO-LOCK reads).

QAD
Our Passion. Your Advantage.

Record Contention

- Occurs when more than one user attempts to update a record at the same time
- Transaction scope is the primary determinant of lock duration
- Reduce record contention by:
  - Run batch updates during off hours (for file-intense applications)
  - Training: don't leave transactions open
  - Coding: don't leave transactions open

Record Contention

QAD
Our Passion. Your Advantage.

02-029.p – 02-030.p

Releasing Locks

Lock Acquired →	During Trans	Outside Trans	Outside but held going into Trans
SHARE	Held until trans. end or record release, which-ever is later.	Held until record release	Held until trans. end or record release, which-ever is later.
EXCLUSIVE	Held until trans. end; converted to SHARE if record scope is larger than transaction.	N/A	N/A

QAD Proprietary 65

Releasing Locks

QAD
Our Passion. Your Advantage.

Record Scope

- A record is read from the database and stored in a record buffer
- Scope is the portion of code during which the record is active
- Record scope controls:
  - When to write a record to the database
  - When to release a record for other users
    - also depends on type of lock and transaction scope
  - When to validate a record
  - When to reinitialize the index cursor
    - initialized upon entry to procedure block where record is scoped

QAD Proprietary 66

Record Scope

The portion of time that a record is stored and active in the record buffer.

This determines how long the record is in the buffer, when it is written to the database, and how long a record lock is in effect.

QAD
Our Passion. Your Advantage.

Programming Impact

- Important for its impact on transaction scope
- Also a source of programming compile and runtime errors
- Progress manages memory resources using record scope
- Communicates with the transaction process to upgrade record locks as necessary

QAD Proprietary 67

Programming Impact

QAD
Our Passion. Your Advantage.

Types of Record Scope

- Four types of record scope:
  1. Strong-scoped reference
     - Buffer is available only in the block where the record is scoped
  2. Medium-scoped reference
     - Buffer available for future finds, but empty, at end of scope
  3. Weak-scoped reference
     - Buffer available, and record in buffer available, at end of scope
  4. Free reference - all other buffer references (FIND, GET etc.)
     - Scopes buffer to the procedure block and raises the scope of weak and medium scoped records where they occur together

Types of Record Scope

QAD
Our Passion. Your Advantage.

Scope Reference

	strong	medium	weak
Can you FIND a record after?	NO	YES	YES
Can you DISPLAY a record after?	NO	NO	YES
Which block is record scoped to?	Strong	Medium*	Weak*

* Scope raised to enclosing block on free references

QAD Proprietary 69

Scope Reference

QAD
Our Passion. Your Advantage.

Block Scope Reference

STRONG:	DO FOR <buffer>: REPEAT FOR <buffer>:
MEDIUM:	REPEAT: REPEAT ON ERROR, ENDKEY: FOR EACH:
WEAK:	REPEAT TRANSACTION: FOR FIRST/LAST:
NONE (FREE):	DO: DO TRANSACTION: DO ON ERROR, ENDKEY:

Block Scope Reference

QAD
Our Passion. Your Advantage.

02-032.p – 02-033.p

Strong Scoped Records

- Strong-scoped reference to a buffer
  - buffer is always scoped to the strong-scoped block
  - two sequential strong-scoped blocks can reference the same buffer
  - cannot reference that buffer in a containing block
  - illegal references to fields return compile error:

    *"** Missing FOR, FIND or CREATE for a table with cm_addr in current block. (232) "*
  - illegal references to a table return compile error:

    *"** Reference to table cm_mstr conflicts with block statement reference. (244) "*

QAD Proprietary 71

Strong Scoped Records

QAD
Our Passion. Your Advantage.

02-034.p – 02-036.p

Medium Scoped Records

- Medium-scoped reference to a buffer
  - buffer is scoped to the block, unless raised by a free reference to the enclosing block
  - cannot redisplay buffer outside the medium-scope block
  - cannot nest two medium-scope blocks for the same buffer
    - "** Illegal nested block statement reference to table cm_mstr. (243)"
  - illegal DISPLAY of record outside the block returns the runtime error:
    - "**No <tablename> record is available (91).”
  - can perform a find on the buffer table

Medium Scoped Records

QAD
Our Passion. Your Advantage.

02-037.p – 02-039.p

Weak-Scoped Records

- Weak-scoped reference to a buffer
  - buffer is scoped to the block, unless raised by a free reference to the enclosing block
  - can display or find buffer outside the weak-scope block
  - cannot nest two weak-scope blocks for the same buffer
  - weak-scope block cannot contain any free references to the same buffer (that is, no FINDs allowed in a FOR EACH)

QAD Proprietary 73

Weak Scoped Records

QAD
Our Passion. Your Advantage.

02-040.p

Free References

- PROGRESS tries to scope buffer to the nearest enclosing block with record scoping properties
- Free references are:
  - FIND
  - GET
  - CREATE
  - INSERT

Free References

CHAPTER 5

Predicates, Indexes, and Field Lists

QAD
Our Passion. Your Advantage.

Predicates, Indexes and Field Lists

	F1	F2	F3	F4	F5
R1				x	
R2		x			
R3		x		x	
R4		x		x	
R5		x			

Predicates:
"WHERE F2 AND F4…"
returns ENTIRE R3 & R4.
Value-dependent,
selects index used.

	F1	F2	F3	F4	F5
R1		1		R	
R2		2			
R3		22		S	
R4		3		Z	
R5		31			

Indexes:
"USE-INDEX F2-F4…"
return all records
sorted by F2 and F4.

	F1	F2	F3	F4	F5
R1				x	
R2		x			
R3		x		x	
R4		x		x	
R5		x			

Field Lists:
"F2, F4."
returns two columns
for all records.

Combine to select records by value, sort records, and select relevant columns.

QAD Proprietary 77

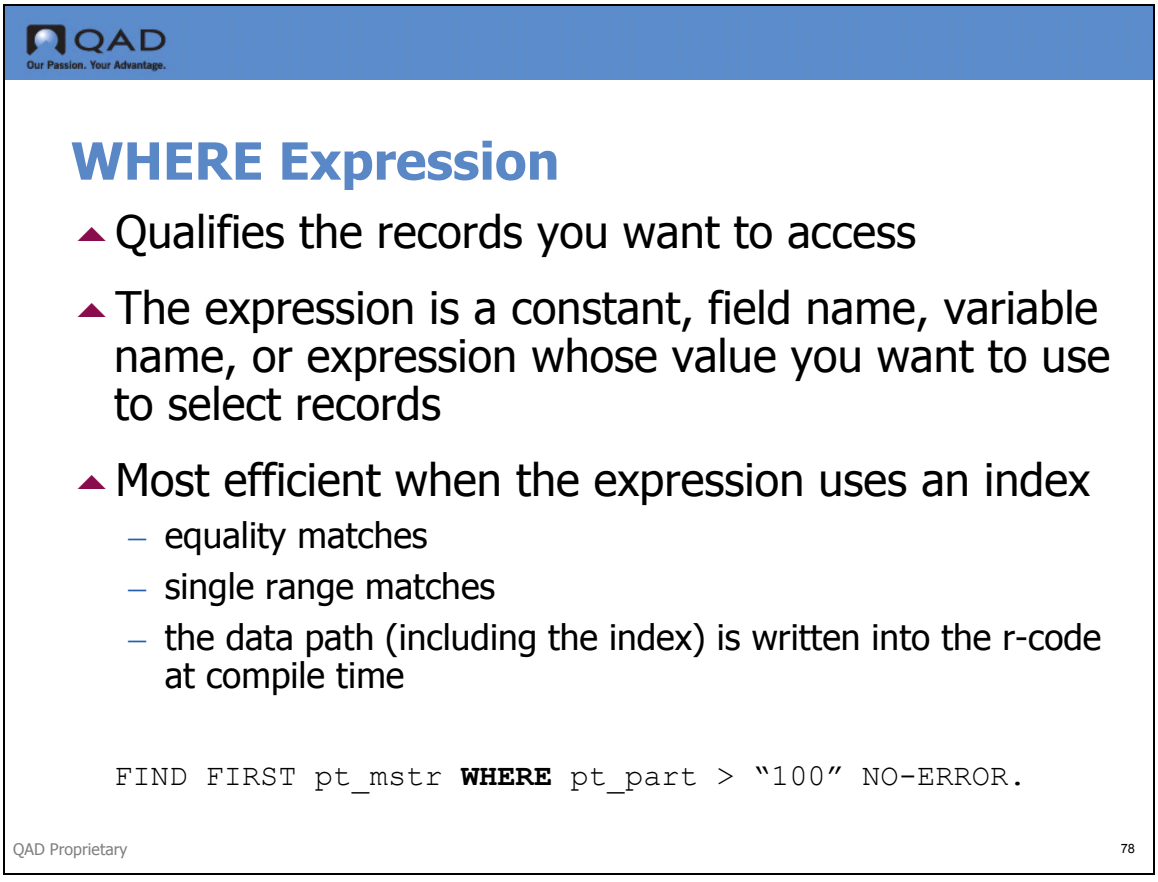
Predicates, Indexes, and Field Lists

- Predicate

  Where clause. Returns entire record selected.
- Index

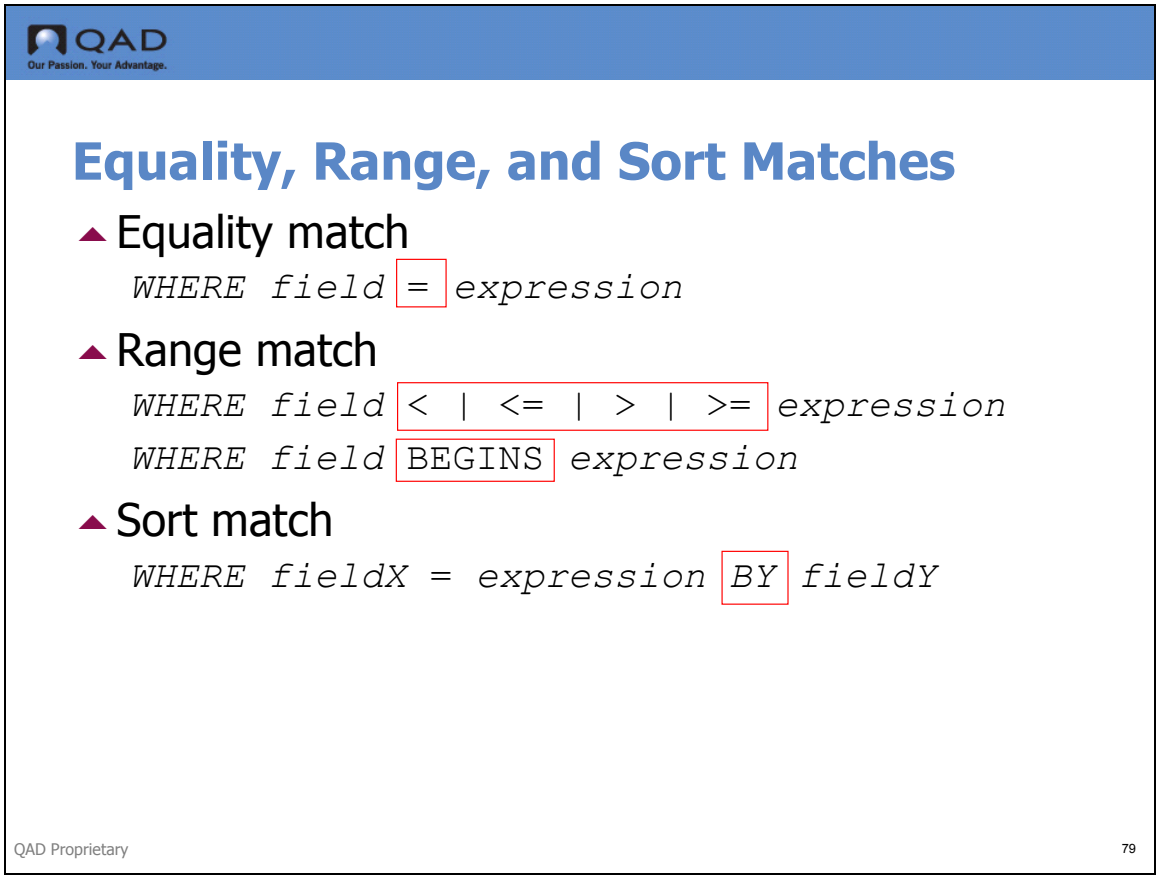
  When specified using the USE-INDEX keyword, indicates sort, not just efficient lookup.
- Field Lists

  Only returns the list of fields specified to the record buffer.



WHERE Expression

The expression (where clause) should bracket an index whenever possible. It should use indexed fields for efficient search.



Equality, Range, and Sort Matches

- Equality

  Where field = value or expression
- Range

  Where field is greater than, less than, greater than or equal to, less than or equal to a value
- Begins

  The value is present at the beginning of the string
- Sort Match

  Select based on X criteria, but sort using field Y via the BY word

QAD
Our Passion. Your Advantage.

Operators—Logical

Operator	Description
NOT	returns a TRUE value if an expression is false, and FALSE if an expression is true
AND	returns a TRUE value if all expressions are true, otherwise FALSE
OR	returns a TRUE value if at least one value is true, otherwise FALSE

Operators–Logical

QAD
Our Passion. Your Advantage.

02-041.p

Operators—Comparison

Name	Operator
less than	< or LT
less than or equal to	<= or LE
greater than	> or GT
greater than or equal to	>= or GE
equal to	= or EQ
not equal to	<> or NE

QAD Proprietary 81

Operators–Comparison

QAD
Our Passion. Your Advantage.

BEGINS Operator

- Tests a character expression to see if that expression begins with a second character expression
- May search on an index if the field is the leading component of an index.
- Example:

```
FOR EACH cm_mstr WHERE cm_sort BEGINS "jo":
   DISPLAY cm_sort.
END.
```

  - NOTE: The query will find "**Jo**hnson" and "**Jo**hn Smith." The query will not find "J Johnson & CO."

QAD Proprietary 82

BEGINS Operator

QAD
Our Passion. Your Advantage.

MATCHES Operator

- Compares a character expression to a pattern and evaluates to TRUE if the expression satisfies the pattern criteria
- Performs a sequential search, will not bracket from an index
- Example:

```
FOR EACH cm_mstr WHERE cm_sort MATCHES "jo*":
   DISPLAY cm_sort.
END.
```

- Use wildcard characters for pattern matching
  - asterisk (*) = any one or more characters
  - period (.) = one character only

MATCHES Operator

QAD
Our Passion. Your Advantage.

02-042.p

CONTAINS Operator

- Checks whether the supplied string is contained anywhere in the field reference when the field is defined as a word index
- Requires a word index

```
 FOR EACH cm_mstr WHERE cm_sort CONTAINS
"John Doe"
    DISPLAY cm_sort.
END.
```

The space is treated as a logical AND. The search will find "John Doe and Company," "Doe & John," and "John Franklin Doe." The search will not find "Johnson and Doe Inc.," "Doe & Company," or "John Acker CEO."

QAD Proprietary 84

CONTAINS Operators

QAD
Our Passion. Your Advantage.

Sorting with BY Option

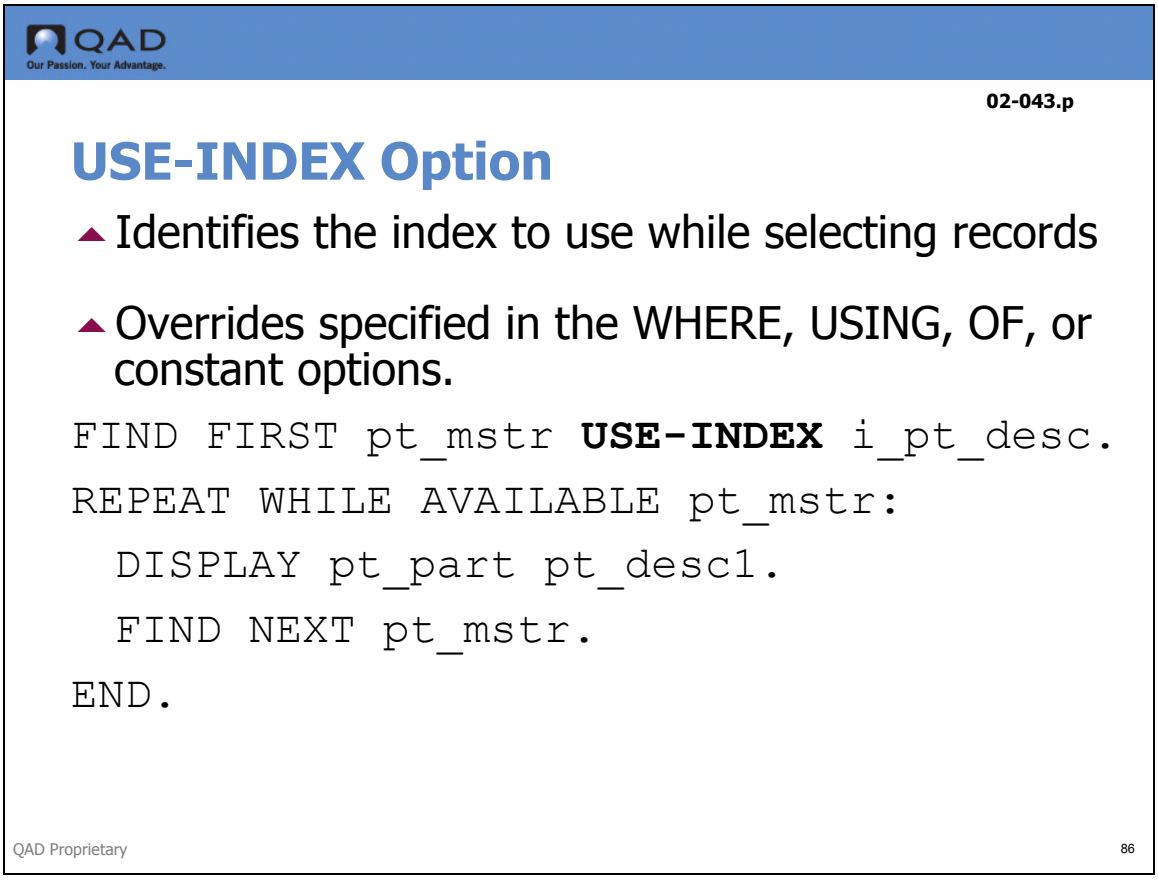
- Syntax:

```
WHERE searchExp BY field1 [DESCENDING]
```

- Sorts the selected records using *field1*
- Without the BY option, records retrieved in the order of the index used to satisfy the *searchExp* criteria
- If the BY field is not indexed, a temporary sort table is generated after selection to sort and then retrieve the records; could be a performance hit
- Example:

```
FOR EACH cm_mstr BY cm_cr_limit BY cm_sort:
```


Sorting with BY Option



USE-INDEX Option

Using the index of `I_pt_desc`, find the first `pt_mstr`.

This indicates that it might not be the first part, but the first description alphabetically. Then, since the FIND NEXT does not specify any index, the primary index is invoked for the remainder of the procedure. So the first `pt_mstr` using the description will be found. Where ever that part number is within the `pt_mstr`, the remainder of the `pt_mstr` will then be displayed.

QAD
Our Passion. Your Advantage.

Indexing

- Indexing greatly impacts performance
- Top reasons of poor performance
  - incorrect index design
  - index design okay but correct index not used by code with use-index
  - the BY option does not match the index sort

QAD Proprietary 87

Indexing

QAD
Our Passion. Your Advantage.

Simple WHERE Clause

```
WHERE searchExp [BY field1]
```

- If there is an index on the searchExp field, or if the field is first in a compound index, uses that index
- Else: Primary index
- If searchExp references a word index, uses that
- BY clause uses an index for *field1* if available
- Else: temporary sort table built

Simple WHERE Clause

QAD
Our Passion. Your Advantage.

Compound WHERE Clause 1

```
WHERE searchExp AND searchExp [BY
  field1]
```

- Progress evaluates both sides of compound clauses
- On FOR EACH, if both sides are indexed, or are leading components of compound indexes, uses both
- Else: whichever index qualifies and primary index
- FINDs use only first qualifying index, else primary
- Progress does not guarantee sort order for multiple index retrievals
  - Use the BY option, or the USE-INDEX option to guarantee sort order

QAD Proprietary 89

Compound WHERE Clause 1

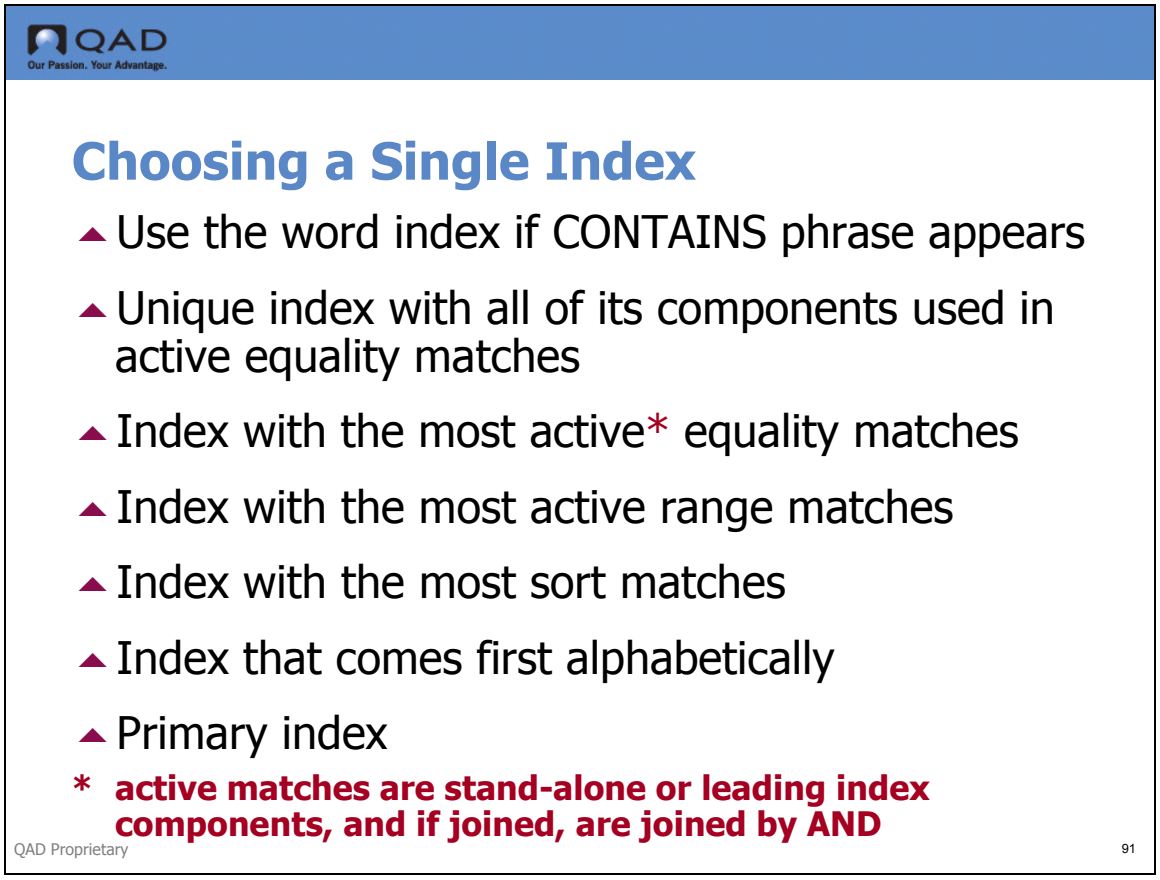
QAD
Our Passion. Your Advantage.

Compound WHERE Clause 2

```
WHERE searchExp OR searchExp [BY
  field1]
```

- Progress evaluates both sides of compound clauses
- If both sides are indexed, or are leading components of compound indexes, uses both
- Else: whichever index qualifies and primary index

Compound WHERE Clause 2



Choosing a Single Index

This is the order in which Progress will select an index for any lookup if the index is not mandated via the USE-INDEX option.

QAD
Our Passion. Your Advantage.

XREF Output

```
1 COMPILE d:\v7\v7ctrain\code\x.p
1 CPINTERNAL iso8859-1
1 CPSTREAM ibm850
3 STRING "pt_mstr" 8 NONE UNTRANSLATABLE
3 SEARCH train1.pt_mstr i_pt_part WHOLE-INDEX
4 ACCESS train1. pt_mstr pt_part
4 STRING ">>9" 3 NONE TRANSLATABLE  FORMAT
5 STRING "Part Number" 15 LEFT TRANSLATABLE
5 STRING "pt_part" 8 LEFT TRANSLATABLE
5 STRING "----------------" 17 NONE
  UNTRANSLATABLE
5 STRING "i_pt_part" 10 NONE UNTRANSLATABLE
```


XREF Output

QAD
Our Passion. Your Advantage.

02-044.p

Word Index

```
FOR EACH so_mstr WHERE so_rmks CONTAINS
  "rush"
  AND so_nbr = "3000":
     DISPLAY so_nbr so_due_date so_rmks.
END.
```

- Uses index so_rmks

QAD Proprietary 93

Word Index

QAD
Our Passion. Your Advantage.

02-045.p

Unique Index – Equality Matches

```
FOR EACH pt_mstr WHERE pt_critical = NO
    AND pt_group = "G1"
   AND pt_status = "":
   DISPLAY pt_group pt_status pt_price.
END.
```

Uses index i_pt_critical

- Equality matches on all components
- Used on *type, status, group* fields

QAD Proprietary 94

Unique Index–Equality Matches

QAD
Our Passion. Your Advantage.

02-046.p

Most Active Equality Matches

```
FOR EACH pt_mstr WHERE pt_group = "G1" AND
    pt_part >= "100" AND pt_price > 1.00:
    DISPLAY pt_group pt_status pt_price.
END.
```

Uses index i_pt_group

- pt_group has an active equality match, pt_part has an active range match

QAD Proprietary 95

Most Active Equality Matches

QAD
Our Passion. Your Advantage.

02-047.p

Most Active Range Matches

```
FOR EACH pt_mstr WHERE pt_desc1 >= "a"
     AND pt_group MATCHES "G*":
    DISPLAY pt_part pt_group pt_price.
END.
```

- Uses index i_pt_desc
  - the MATCHES operator never brackets an index
  - the field pt_desc1 is the only active range match

Most Active Range Matches

QAD
Our Passion. Your Advantage.

02-048.p

Most Active Sort Matches

```
FOR EACH pt_mstr WHERE pt_qoh > 100 BY
  pt_part:
  DISPLAY pt_desc1 pt_part pt_group
  pt_qoh.
END.
```

- Uses index i_pt_part
  - the field pt_part is the sort index
  - pt_qoh is not indexed
  - **Note**: all sort matches are active

QAD Proprietary 97

Most Active Sort Matches

QAD
Our Passion. Your Advantage.

02-049.p

Alphabetic Index Selection

```
FOR EACH pt_mstr WHERE pt_part > "10"
   AND pt_desc1 BEGINS "a":
   DISPLAY pt_part pt_desc1.
END.
```

- Uses index i_pt_desc
  - a tie: the two indexes have the same number of active range matches
  - use the alphabet to pick the index. i_pt_desc comes before i_pt_part alphabetically

Alphabetic Index Selection

QAD
Our Passion. Your Advantage.

02-050.p – 02-051.p

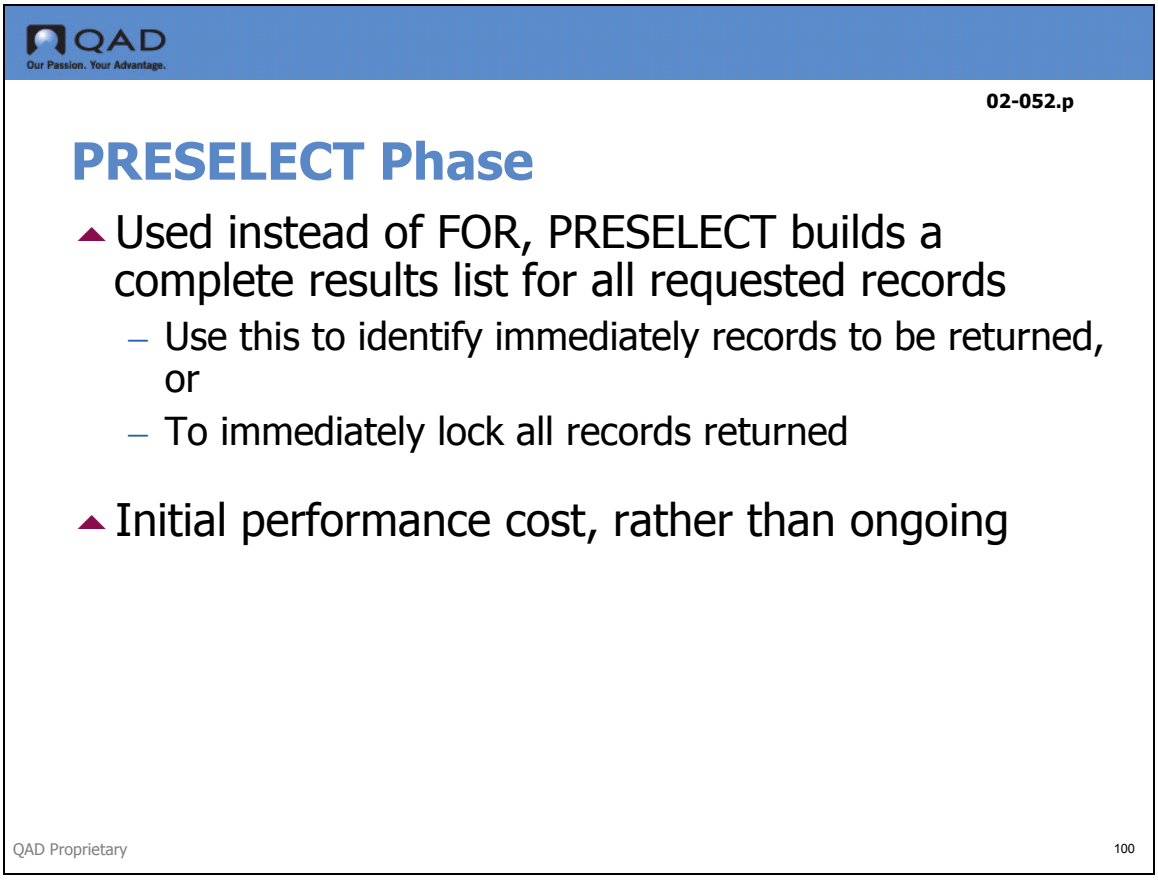
Primary Index

```
FOR EACH pt_mstr WHERE pt_price >
  100.00
        AND pt_qoh > 100:
       DISPLAY pt_part pt_price pt_qoh.
END.
```

- Uses primary index, i_pt_part
  - neither search field indexed
  - selects using primary index
  - may use WHOLE-INDEX

QAD Proprietary 99

Primary Index



PRESELECT Phase

FIND is then used on the records that are part of the PRESELECT.

QAD
Our Passion. Your Advantage.

Bracketing

- Once one or more indexes are selected, Progress attempts to return the fewest records that satisfy
- Rules for bracketing – examine each index component:
  - Bracket on active equality matches
  - Bracket an active range match, but no more after that
  - Unbracketed search:

```
SEARCH reachv9.pt_mstr i_pt_part WHOLE-INDEX
```

QAD Proprietary 101

Bracketing

QAD
Our Passion. Your Advantage.

Some Indexing Guidelines

- As number of records that satisfy a query to total records increases, desirability of index access decreases
- Implement fewer indexes on tables with heavy update activity and limited access, and vice versa
- Most index processing occurs on server side, returning the minimum required records to the client
- To avoid:
  - Joining range matches with AND
  - Non-indexed ORs
  - Multiple wild card strings in word indexes
  - Word indexes OR non-indexed criterion

QAD Proprietary 102

Indexing Guidelines

QAD
Our Passion. Your Advantage.

Field Lists

- A *field list* is a subset of the fields that define a record and includes those fields that the client actually requires from the database server
  - optimizes preselected/presorted fetches from the database server
  - may be used with many types of record fetches
    - FOR Statements
    - queries
    - PRESELECT Statements
    - SQL SELECT Statements
- Syntax

```
record-bufname [FIELDS[([field …])] | EXCEPT[([field …])]]
```

QAD Proprietary 103

Field Lists

QAD
Our Passion. Your Advantage.

Field Lists

- File: Customer
  - Fields: cm_addr cm_balance cm_class cm_cr_hold cm_cr__limit cm_cr_rating cm_cr_terms cm_sort cm_xslspsn

```
FOR EACH cm_mstr
NO-LOCK:
```

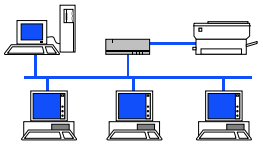
=

```
FOR EACH cm_mstr
   FIELDS(cm_addr cm_balance
          cm_class
          cm_cr_hold
          cm_cr_limit
          cm_cr_rating
          cm_cr_terms
          cm_sort
         cm_xslspsn)
 NO-LOCK:
```


Can be deactivated using the `-fldisable` parameter. Progress will then retrieve the whole record, overriding the field list in the procedure.

QAD
Our Passion. Your Advantage.

Field List Benefits



- Fetching over a network
  - reduction in network traffic
- Fetching from local DataServers
  - benefits from reading only a portion of the record
- Fetching from remote DataServers that send multiple rows at a time
  - DataServers that package multiple rows per network message benefit from field list fetches
- Compiling using DataServers for foreign databases
  - Fetch commands store foreign database commands as ASCII text in the r-code. Without field lists, the list of field information can be quite large, thus increasing the size of the r-code

Field List Benefits

MUST call out every field that is required for update or display.

QAD
Our Passion. Your Advantage.

Lab 4

1. Allow users to update customer records. Make sure appropriate updates occur to the associated ad_mstr records, using the Class File Relationships slide. (L0401.p)

2. Allow users to update or create new customer records. Make the appropriate assignments to ad_mstr. (L0402.p)

3. Display the sales order detail by customer and order. Display the following fields: sod_line, sod_qty_item, sod_price, and sod_price multiplied by sod_qty_item. (L0403.p)

QAD
Our Passion. Your Advantage.

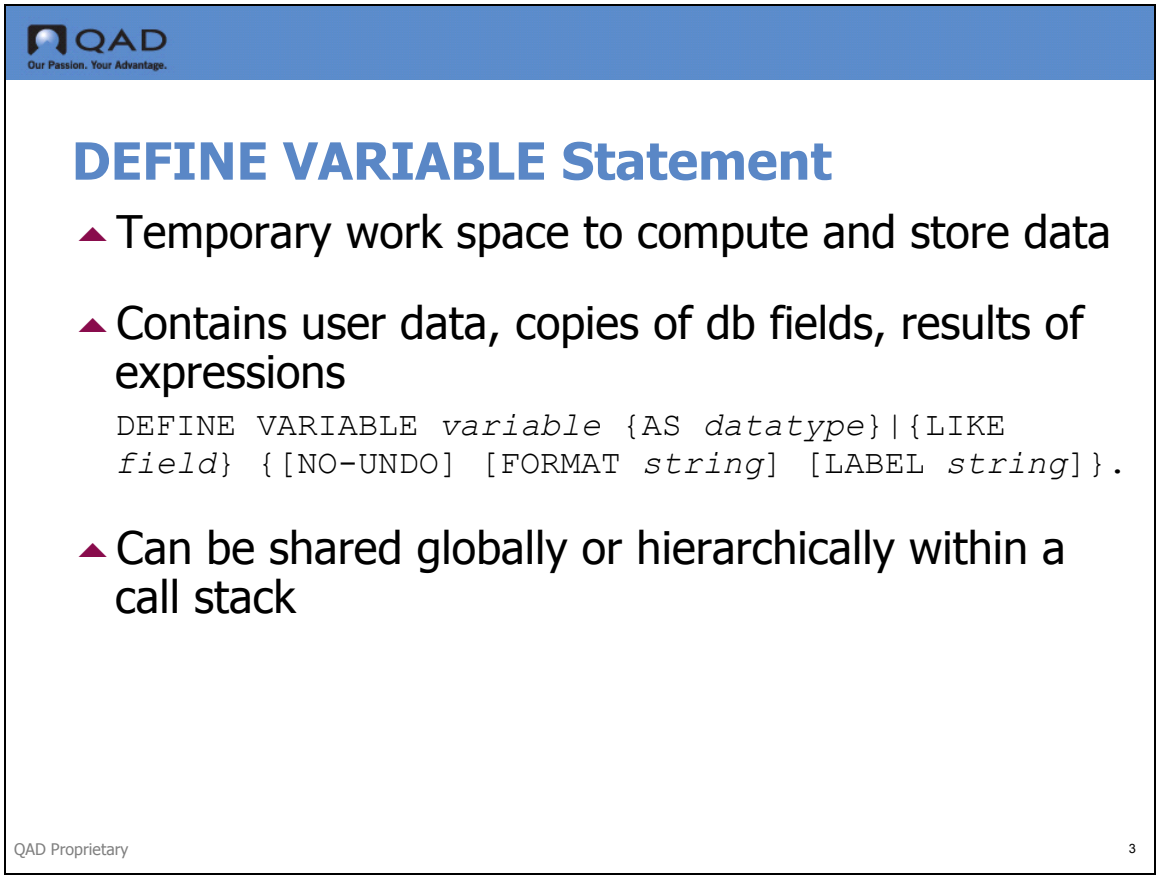
Lab 4 Review

QAD Proprietary 107

Lab 4 Review

CHAPTER 6

Variables, Messages, and Conversions

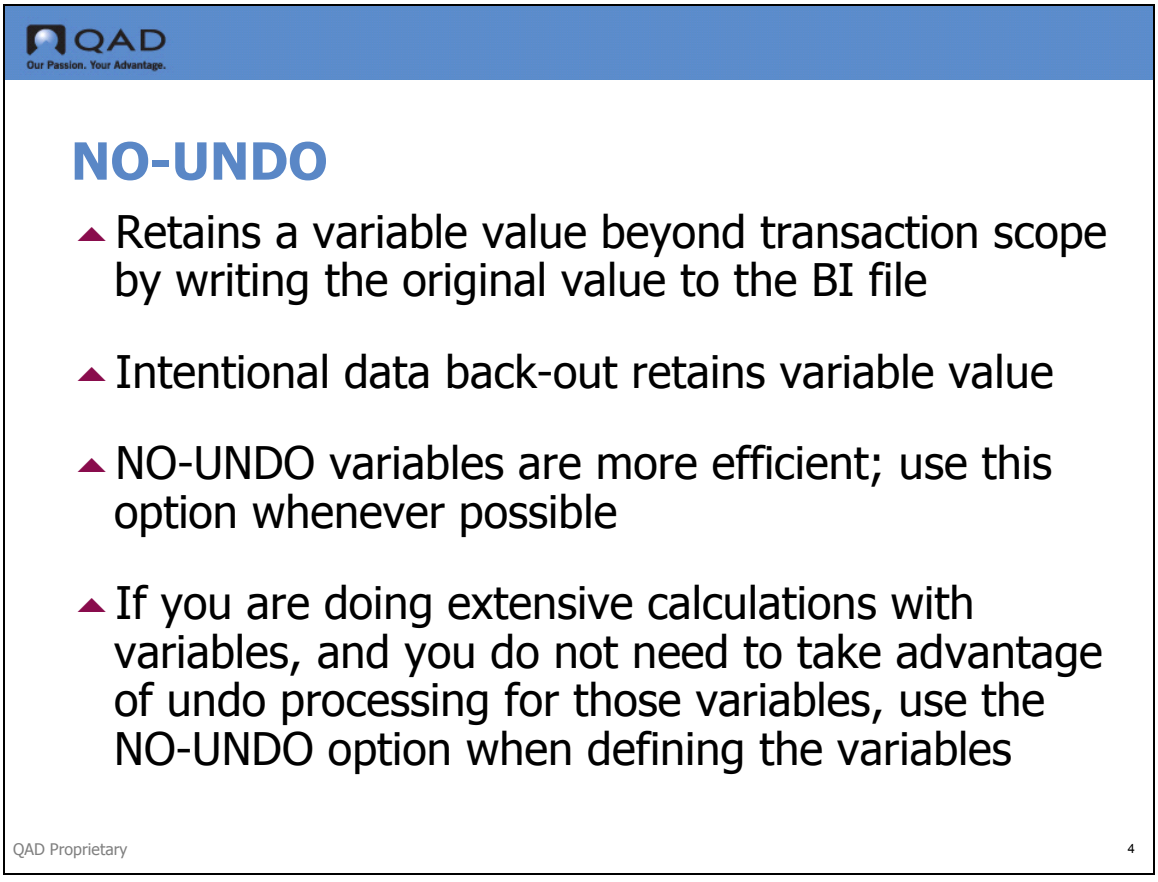


DEFINE VARIABLE Statement

SHARED or GLOBAL.

If a variable is defined LIKE a database field, then it takes on the same attributes as database field.

If format is not specified, the default format for the field type is the defined format.



NO-UNDO

Used when a value does not need to be undone if a particular block is terminated.

QAD
Our Passion. Your Advantage.

02-057.p – 02-058.p

Screen Input: INPUT

- INPUT Option
  - references the value of a field or variable in the screen buffer
  - PROMPT-FOR stores user entries in window buffer; INPUT function can look-up, display or assign the values there
- Like USING option, but not limited to database fields

```
DEFINE VARIABLE x LIKE pt_part LABEL "Part Number".
PROMPT-FOR x.
FIND pt_mstr WHERE pt_part = INPUT x NO-ERROR.
```

QAD Proprietary 5

Screen Input: INPUT

Uses the value of the input buffer for the defined variable

QAD
Our Passion. Your Advantage.

STATUS Statement

- Specifies the text that appears in the status line of a window
- Syntax

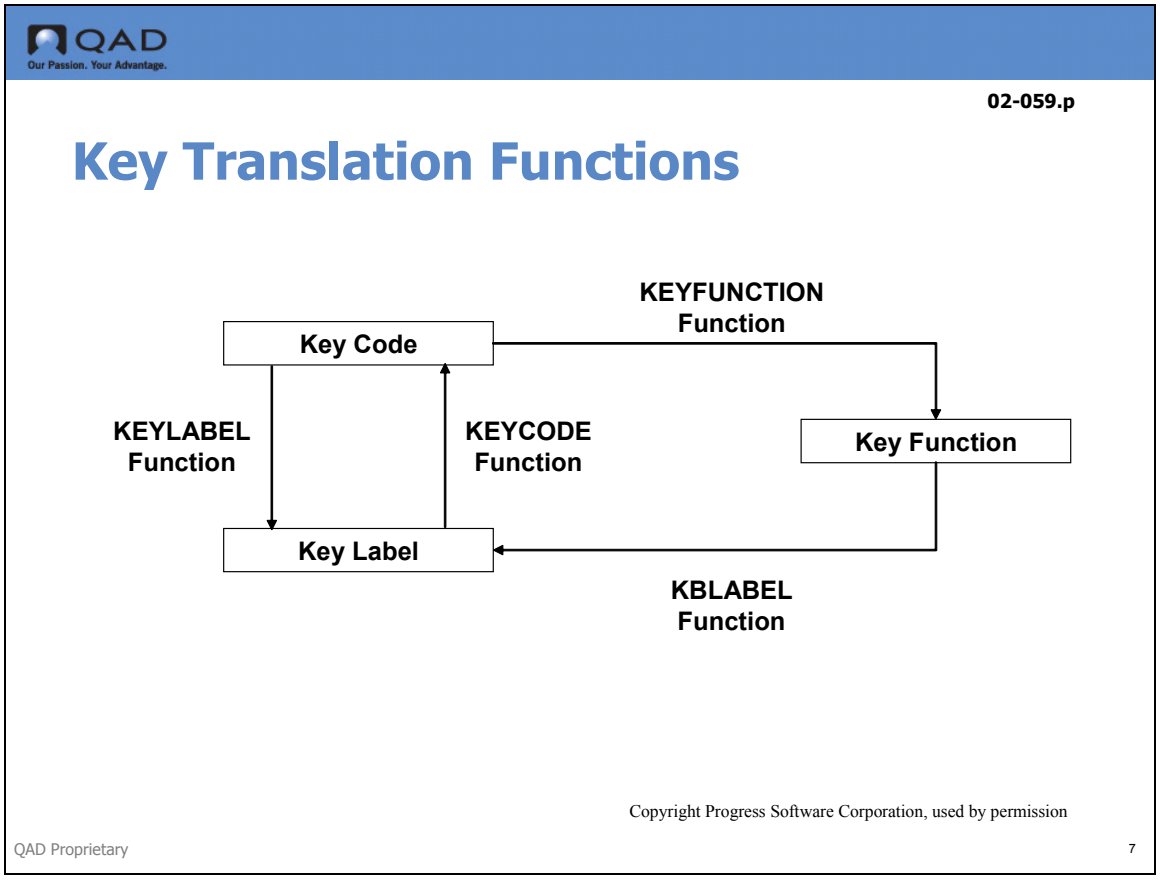
  `STATUS{{DEFAULT[`*`expression`*`]}|{INPUT[OFF|expression]}}`

  – Limit of 63 characters
- Reset STATUS at the end of the code requiring the change

QAD Proprietary 6

STATUS Statement

STATUS appears frequently in QAD Enterprise Applications code. It is used to display the function key options at the bottom of the screen. This does not control how these keys operate, but lists what keys are active for what purpose.



Key Translation Functions

PROGRESS can port across platforms because of the ability to translate key definitions.

For example: GO is F1 in a character environment, CTRL-X in some platforms and F2 in GUI/ Windows. How these keys translate can be built into the code for easy error or exit processing.

- READKEY

  Reads one keystroke at a time and sets the value of LASTKEY to the KeyCode of that keystroke.

- LASTKEY

  Returns the integer value (keycode) of the most recent user event (keystroke)

QAD
Our Passion. Your Advantage.

Message Statement

- Displays messages in the message area at the bottom of the window or in an alert box
- Can include simple SET or UPDATE
- Procedural checkpoint
- Basic syntax:

```
MESSAGE {expression|{SKIP [(n)]}}... [VIEW-AS
  ALERT-BOX [alert-type] [BUTTONS button-set]
  [TITLE title-string]] [{SET|UPDATE} field]
```

- Example:

```
DEFINE VARIABLE x AS LOGICAL.
MESSAGE "Add Part?" UPDATE x.
```


Message Statement

QAD
Our Passion. Your Advantage.

02-060.p – 02-061.p

ALERT-BOX Phrase

- Graphical message box with buttons
- INFO, QUESTION, WARNING, ERROR types
- YES-NO, YES-NO-CANCEL, OK, OK-CANCEL, RETRY-CANCEL button sets
- Alert-box messages can only update logical variables

```
MESSAGE
    "Update item?"
    VIEW-AS ALERT-BOX QUESTION BUTTON YES-NO-CANCEL
    UPDATE x AS LOGICAL.
```

QAD Proprietary 9

ALERT-BOX Phrase

QAD
Our Passion. Your Advantage.

STRING, SUBSTRING Functions

- STRING converts a value of any data type into a character value
- Syntax:

```
STRING(source[,format])
```

- SUBSTRING extracts a portion of a character string from a field or variable
- Syntax:

```
SUBSTRING (source,position[,length[,type]])
```


STRING, SUBSTRING Functions

QAD
Our Passion. Your Advantage.

TRIM Function

- Removes leading and trailing white space, or other specified characters, from a character string.
- Syntax:

```
TRIM(string[,trim-chars])
```

QAD Proprietary 11

TRIM Function

QAD
Our Passion. Your Advantage.

02-065.p

CASE Statement

- Provides a multi-way decision based on the value of a single expression
- Put the most likely matches first
  - On the first branch where the *value* matches the *expression*, the associated *statement* is executed

```
CASE expression:
    {WHEN value [OR WHEN value]… THEN {statement}}
    [OTHERWISE {statement}]
END [CASE].
```

- Added in Version 8

QAD Proprietary 13

CASE Statement

QAD
Our Passion. Your Advantage.

IF ... THEN ... ELSE Function

- Evaluates to one of two expressions, depending on the value of a third conditional expression
- Syntax:

```
IF conditional-expression
   THEN expression1 ELSE expression2
```

- Example:

```
ASSIGN
   msg = IF x > 5 THEN "High" ELSE "Low".
```

QAD Proprietary 14

IF...THEN...ELSE Function

QAD
Our Passion. Your Advantage.

Editing Phrase

- Allows the application to manage the input one key stroke at a time during a PROMPT-FOR, SET, or UPDATE statement
- This block executes once for each key stroke received during execution of the input statement
- Basic syntax

```
UPDATE field1 field2… EDITING:
   READKEY.
   APPLY LASTKEY.
END.
```

QAD Proprietary 15

Editing Phrase

EDITING is used when there needs to be a protection against specific characters within an input field. For example, in User Maintenance, if a special character is restricted, there could be logic to prevent a special character by disallowing entry of that character keystroke.

QAD
Our Passion. Your Advantage.

02-066.p – 02-067

FRAME-FIELD, FRAME-VALUE

- FRAME-FIELD returns the name of the input field the cursor is in during data entry; otherwise returns the name of the input field the cursor was last in
- FRAME-VALUE returns the (character string) value of the input field that the cursor is in to the current input field; otherwise returns the (character string) value of the input field the cursor was last in
  - Can be assigned

QAD Proprietary 16

FRAME-FIELD, FRAME-VALUE

Used along with EDITING. Identifies what value is in the screen buffer where the cursor resides. For example, if you begin to type a part number and then press F2 help. The browse recognizes what was already input and begins the browse as of that substring

CHAPTER 7

Validation and Help

QAD
Our Passion. Your Advantage.

Validation Overview

- Validation is method of ensuring data integrity
- Most effective on the schema level
- Improved schema validation via triggers unavailable in earlier Progress versions - covered later in course
- Field validation only in procedural model

QAD Proprietary 19

Validation Overview

QAD
Our Passion. Your Advantage.

Field-Level Validation

- Schema-defined
  - works in all programs that reference the field
- Session-defined
  - overrides the schema-defined validation
  - specified in the field's *format-phrase*
  - can also use to "turn off" schema defined validation
  - VALIDATE Statement and Method()
- Bound in the r-code at compile time
- For complex validations use schema or session triggers

QAD Proprietary 20

Field-Level Validation

- SCHEMA-Defined

  Actually resides within the data dictionary on the field. The .v program is bound into the r-code at compile time for any procedure that references that field

QAD
Our Passion. Your Advantage.

Session-Defined Validation

- Specifies a condition which must be true for user input to be accepted
- Specifies error message to be displayed if not
- Validation occurs when the field is modified
  - if the field is not modified, validation occurs on the GO event
- Example

```
UPDATE
   x
   y VALIDATE (y > 0, "Must be greater than zero")
   z.
```

QAD Proprietary 21

Session-Defined Validation

VALIDATE statement forces the condition to be met. Additionally, it can 'grand-father' in old records that should not be subject to new validations. For example, if a new trigger forces new parts to have price greater than 0, it is possible with the validate statement to only validate for new items, letting old items to not be held to new rules.

QAD
Our Passion. Your Advantage.

02-068.p

VALIDATE Statement

- Verifies that a record complies with mandatory field and unique index definitions
- Because validation is done automatically, you rarely have to use the VALIDATE statement
- PROGRESS automatically validates a record when
  - a record in the record buffer is replaced by another
  - a record's scope iterates or ends
  - the innermost iterating subtransaction block that creates a record iterates, or
  - a transaction ends

VALIDATE Statement

QAD
Our Passion. Your Advantage.

VALIDATE() Method

- Executes any validation tests established in a database or by VALIDATE option of the format phrase
- Executes validation tests for every supported field-level object in the frame or dialog
- Logical; returns TRUE if successful
- On failure, validation message displays and focus moves to first failed screen object

VALIDATE() Method

QAD
Our Passion. Your Advantage.

CAN-FIND Function

- Returns TRUE if a record is found that meets the specified FIND criteria; otherwise returns FALSE
- Use to see if a record exists with less system overhead than a FIND – does not retrieve record
- Basic syntax

```
CAN-FIND([FIRST|LAST] record [WHERE expression]
[USE-INDEX index] [SHARE-LOCK|NO-LOCK] [NO-WAIT])
```

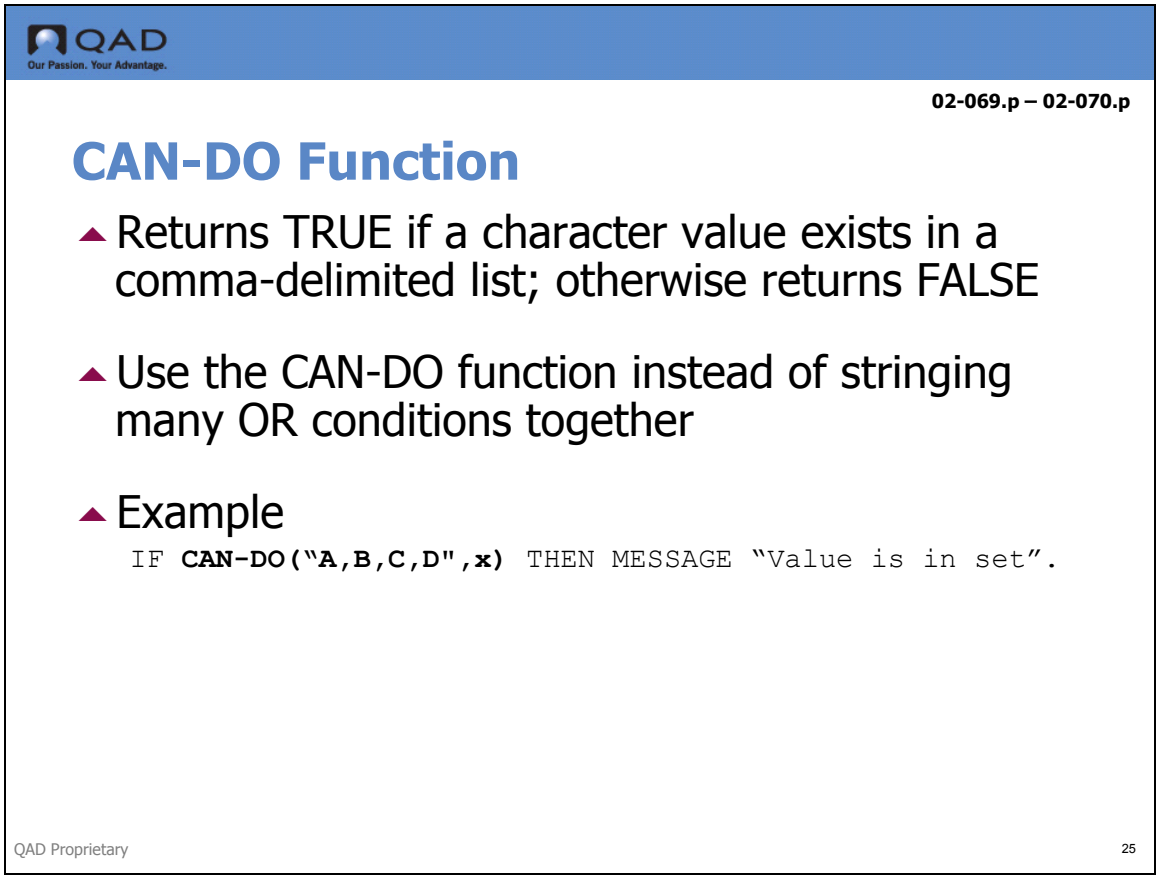
- Example

```
IF CAN-FIND(cm_mstr WHERE cm_addr = "1") THEN
   MESSAGE "Record exists.".
```

QAD Proprietary 24

CAN-FIND Function

Does not actually pull record into the record buffer. Merely a way to test for existence of the record.



CAN-DO Function

If the current value of 'x' is present in the list of arguments (A, B, C and D), then TRUE is returned.

QAD
Our Passion. Your Advantage.

Field-Level Help

- If you define field-level help in the schema, the help message appears in all programs that reference the field
- To override the schema field-level help, you may define field-level help in the *format-phrase* for the particular field
- PROGRESS binds the field-level help into the r-code at compile time

QAD Proprietary 26

Field-Level Help

Field-Level Help is hard-coded if not applied at the schema level. This is not the same as traditional FIELD HELP or PROCEDURE HELP. Field-Level Help displays at the bottom of a screen while the field the help applies to is in focus.

Use the HELP statement to hardcode.

QAD
Our Passion. Your Advantage.

Field-Level Help

- Limited to 63 characters
  - truncated at runtime
- If input device is not terminal, HELP strings ignored
- Can also specify NO-HELP to override Data Dictionary and FORMAT phrase help with no help.

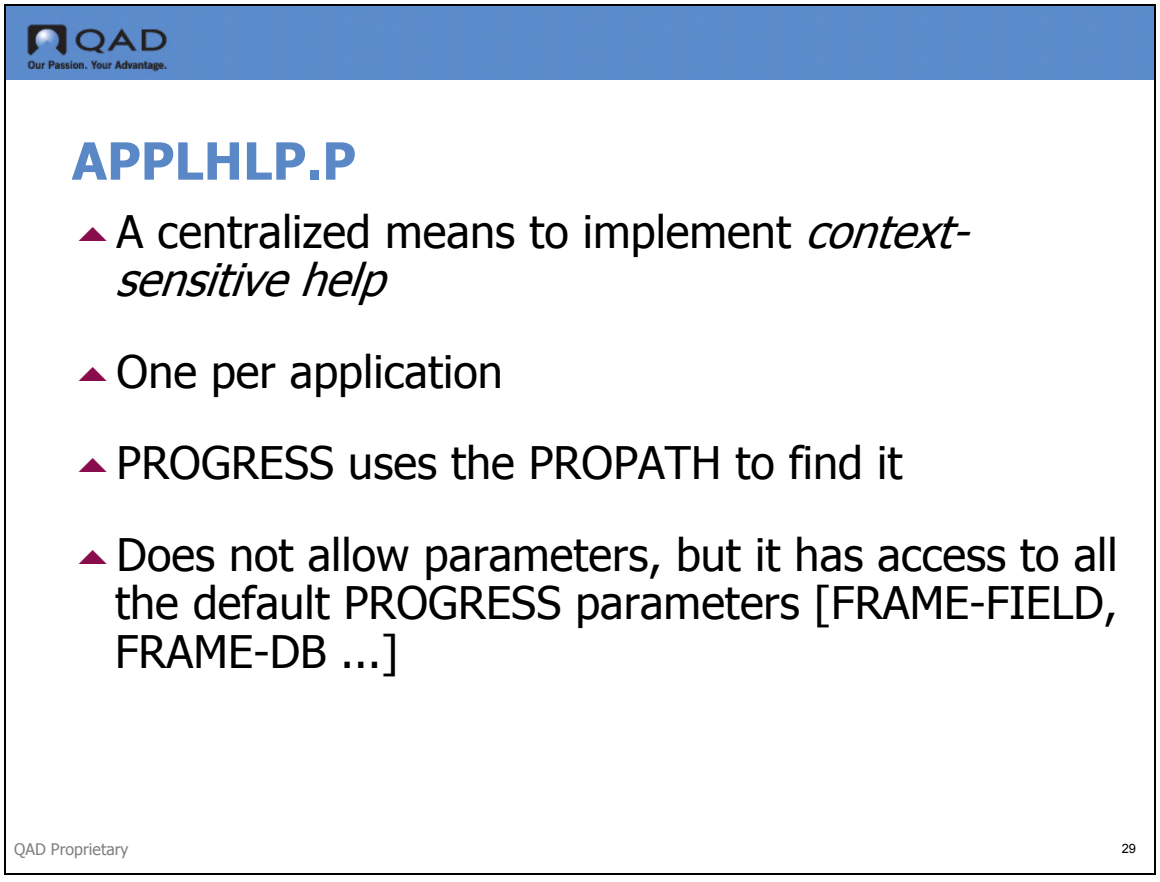
QAD
Our Passion. Your Advantage.

02-071.p – 02-072

Field-Level Help

- Field-level help must be a constant
  - expressions provided as field-level help do not compile
- Example:

```
/* This program does not compile */
FOR EACH cm
   UPDATE
      cm_ad                          de for " +
  cm_sort.
END.
```



APPLHELP.P

Along with `gpaplhlp.p` (in QAD Enterprise Applications), `APPLHELP.P` controls the use of help via the F2 or F6 (user menu) functions.

QAD
Our Passion. Your Advantage.

applhelp.p

Manipulating the PROPATH

- You can change your **PROPATH** on the fly
- PROGRESS searches the PROPATH from left to right to find a program to run
- Changing the PROPATH on the fly speeds up operating system searches
- PROPATH is a character string
- All string functions may be used to manipulate the PROPATH
- Example:

```
UPDATE x.
IF x = "GL" THEN PROPATH = "/usr/gl".
ELSE IF x = "AR" THEN PROPATH = "/usr/ar".
```


Manipulating the PROPATH

QAD
Our Passion. Your Advantage.

Lab 5

1. Update the part master using a variable to find records. Run the update from a user's response to a message statement. (L0501.p)

2. Add a message to allow user to create a new part record. (L0502.p)

3. Add help to pt_desc2 and validation for pt_group. (L0503.p) Valid values for pt_group are G1 and G2.

QAD
Our Passion. Your Advantage.

Lab 5 Review

QAD Proprietary 32

Lab 5 Review

CHAPTER 8

Frames

QAD
Our Passion. Your Advantage.

Frames in Progress

- A rectangular display area within a window that PROGRESS uses to display field-level widgets and other frames
- Allocated to hold field-level widgets referenced by screen display and input statements
- Allocation is determined by the compiler's top-to-bottom pass of a procedure
  - frames are allocated to blocks that display widgets
    - REPEAT blocks
    - FOR EACH blocks
    - DO WITH FRAME blocks
    - Procedure blocks

QAD Proprietary 35

Frames in PROGRESS

QAD
Our Passion. Your Advantage.

Frame Characteristics

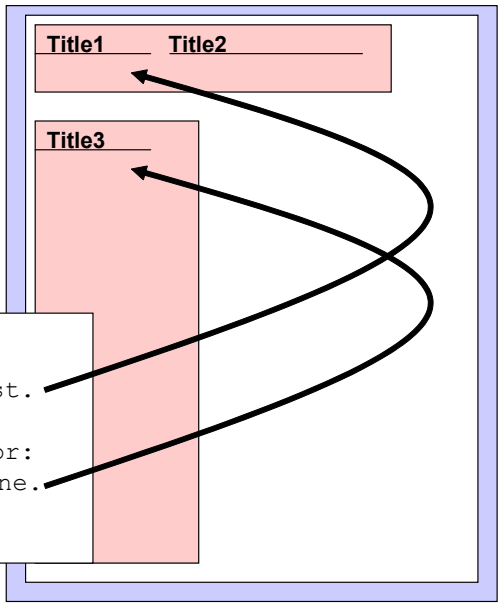
- Allocation:
  - The display of field-level objects (widgets)
- Scope:
  - The display of multiple records or value sets
- Appearance:
  - Use of the frame-phrase to manage size, layout, color, and placement
- Processing:
  - Controlling "down" attributes, reusing frames, scrolling in frames

Frames Characteristics

QAD
Our Passion. Your Advantage.

Allocation: Top-Down Compile

- One pass:
  - screen real estate
  - data processing
- Top-to-bottom
- Left-to-right




```
FOR EACH so_mstr:
        DISPLAY so_nbr so_cust.
        FOR EACH sod_det
        WHERE sod_nbr = so_nbr:
                DISPLAY so_line.
        END.
END.
```

QAD Proprietary 37

Allocation: Top-Down Compile

QAD
Our Passion. Your Advantage.

Default Frame Appearance

- Basic syntax

```
WITH [SIDE-LABELS] [n COLUMNS] [FRAME frame].
```

- Examples

```
INSERT pt_mstr.
INSERT pt_mstr WITH SIDE-LABELS.
INSERT pt_mstr WITH 2 COLUMNS.
```


Default Frame Appearance

QAD
Our Passion. Your Advantage.

Appearance: Frame Phrase

- `INSERT pt_mstr`
  - column labels
  - all fields are displayed
  - fields displayed by Data Dictionary "field order number"
  - if no field label is defined, field name is used
  - PROGRESS subscripts the arrays

QAD Proprietary 39

Appearance: Frame Phrase

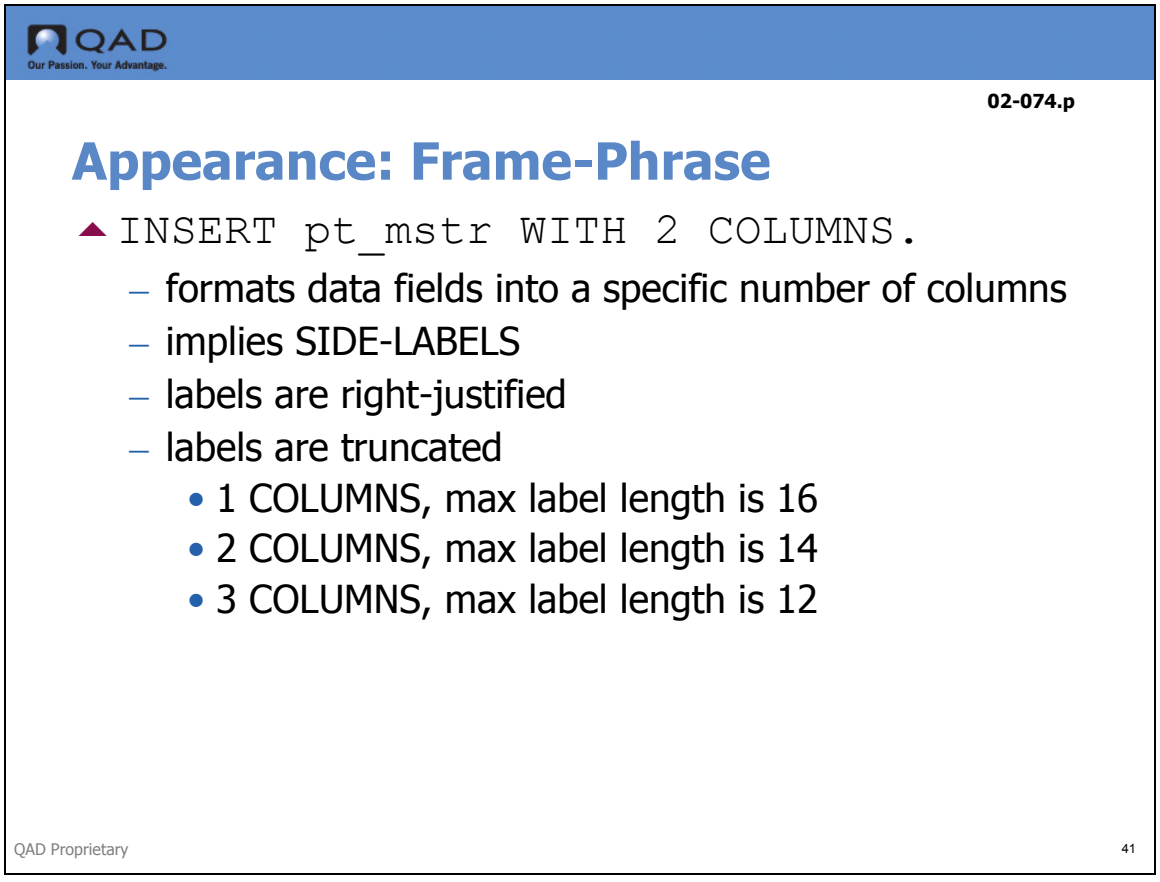
QAD
Our Passion. Your Advantage.

02-073.p

Appearance: Frame-Phrase

- `INSERT pt_mstr WITH SIDE-LABELS`
  - labels display on the left side of the field
  - all else is the same

Appearance: Frame-Phrase



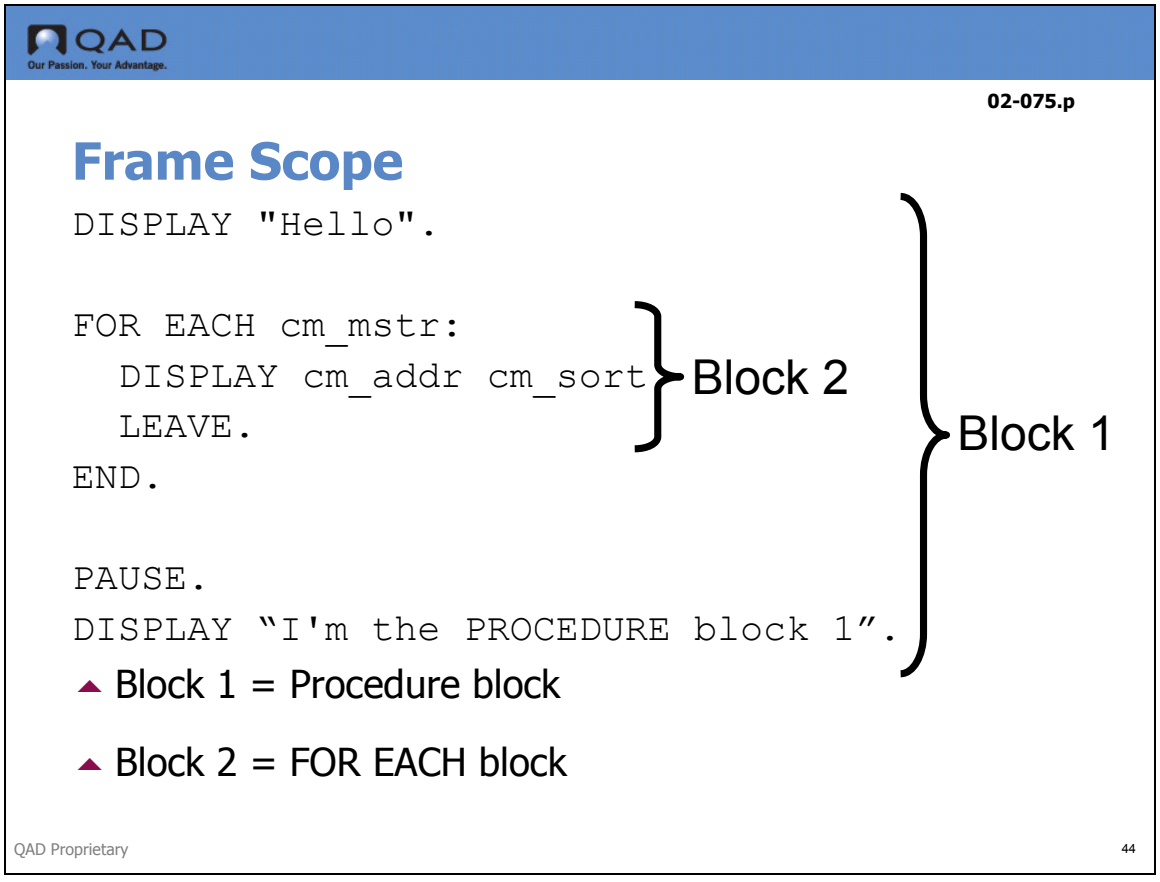
Appearance: Frame-Phrase

QAD
Our Passion. Your Advantage.

Frame Scope

- All REPEAT and FOR EACH blocks scope frames by default
- Innermost iterating block gets a "down frame"
- Frames are scoped to one block only
- Frame scope is not dynamic: memory is not freed when scope is complete

Frame Scope


QAD
Our Passion. Your Advantage.
02-075.p
Frame Scope
DISPLAY "Hello".
FOR EACH cm_mstr:
DISPLAY cm_addr cm_sort
LEAVE.
END.
PAUSE.
DISPLAY "I'm the PROCEDURE block 1".
Block 2
Block 1
Block 1 = Procedure block
Block 2 = FOR EACH block
QAD Proprietary
44

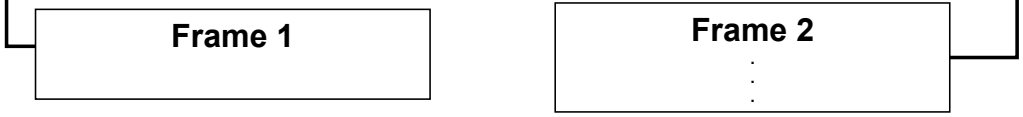
Frame Scope

QAD
Our Passion. Your Advantage.

02-076.p

Frame Scope

```
FOR EACH cm_mstr:
       DISPLAY cm_addr cm_sort.
       FOR EACH so_mstr WHERE so_cust = cm_addr:
              DISPLAY so_nbr so_due_date.
       END.
END.
```



Frame 1 is the default frame scoped to the outer FOR EACH block.

If named, you can also access this frame from the inner FOR EACH block.

Frame 2 is the default frame scoped to the inner FOR EACH block.

You cannot access this frame from the outer FOR EACH block without raising its scope.

QAD Proprietary 45

Frame Scope

QAD
Our Passion. Your Advantage.

Determining Down Frames

Is the block an iterating block?	YES
Is the default frame scoped to the block?	YES
Is it the innermost block (no inside block?)	YES
Then the frame type is:	DOWN
All other default combinations:	ONE-DOWN

QAD Proprietary 46

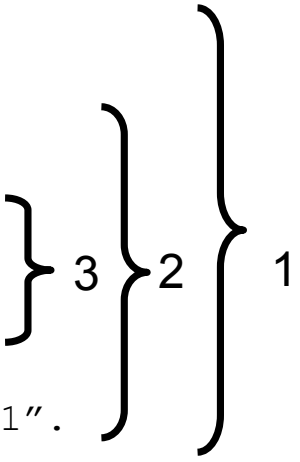
Determining Down Frames

QAD
Our Passion. Your Advantage.

02-077.p

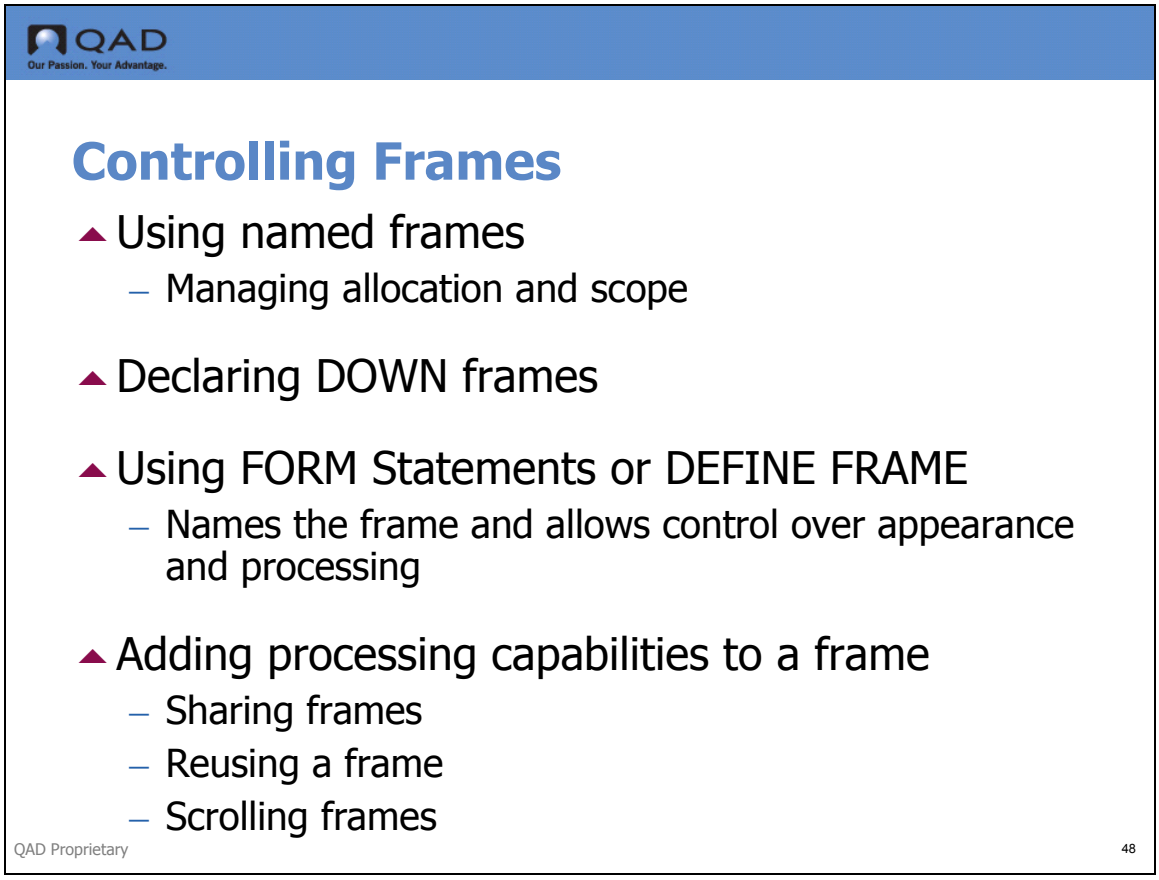
Frame Scope

```
DISPLAY "Hello".
FOR EACH cm_mstr:
   DISPLAY cm_addr cm_sort.

   FOR EACH so_mstr
    WHERE so_cust = cm_addr:
      DISPLAY so_nbr.
   END.
END.
DISPLAY "I'm the PROCEDURE block 1".
```



- Block 1 = Procedure block
- Block 2 = FOR EACH cm_mstr block
- Block 3 = FOR EACH so_mstr block -- DOWN

Frame Scope



Controlling Frames

Sharing a frame is similar to sharing a variable. The frame definition is shared and accessed by multiple procedures, allowing data from multiple procedures to be displayed within the same frame.

QAD
Our Passion. Your Advantage.

Named Frames

- Use named frames to:
  - Reference frames in enclosed blocks (raise the scope of a frame)
  - Control frame appearance and processing
  - Override default frame characteristics

Named Frames

QAD
Our Passion. Your Advantage.

02-078.p – 02-081.p

Named Frames

- Four options:
  - Name the default frame (FOR EACH … with FRAME a) to reuse the frame in an enclosed block
    - Takes on default characteristics
  - Name a new frame for use by the DISPLAY component (DISPLAY … with FRAME a)
    - Loses default characteristics (DOWN)
  - Use the FORM statement
    - Scopes named frames to the procedure
    - Allows for extensive definition of appearance and processing
  - Use the DEFINE FRAME statement
    - Allows you to scope named frames as needed
    - Also allows extensive definition

Named Frames

QAD
Our Passion. Your Advantage.

02-082.p – 02-083.p

Named Frames

- The default frame is used unless you explicitly name a frame and use that frame
- If you name a frame it becomes the default frame and no unnamed frame is allocated to the block
- To raise the scope of a frame (include records from enclosing blocks), scope the frame to an enclosing block
- Scoping a frame to an encompassing block AFTER it has been scoped to an inner block returns a compile-time error:
  - "Invalid reference to a frame outside its scope. (231)"

Named Frames

QAD
Our Passion. Your Advantage.

02-084.p – 02-087.p

Creating Named Down Frames

- Down is allocation of lines AND data
- If a frame is scoped to the block but not the default frame of the block, make it a down frame
- If a frame is not scoped to the block, make it a down frame and use the DOWN WITH FRAME <*framename*> statement

QAD Proprietary 52

Creating Named Down Frames

QAD
Our Passion. Your Advantage.

Frame Flashing

- Occurs when an iterating block displays to a frame that is scoped to an enclosing block
- Each iteration of the block overwrites the data from the previous iteration
- Solved with the DOWN WITH FRAME statement

Frame Flashing

QAD
Our Passion. Your Advantage.

FORM Statement

- Defines the layout and certain processing attributes of a frame
- Scopes the frame to the current block
- Determines the display sequence
- An UPDATE statement using the FORM determines the Tab sequence (unless…)
- Basic syntax

```
FORM [form-item…] [frame-phrase]
```

QAD Proprietary 54

FORM Statement

QAD
Our Passion. Your Advantage.

DEFINE FRAME Statement

- Defines and creates a frame or dialog box that can be used within a procedure or shared among several procedures
- Does not scope the frame to a block until it is used
- Syntax
- DEFINE [[NEW] SHARED] FRAME frame [form-item ...] [DROP-TARGET] [{ HEADER | BACKGROUND } head-item ...] { [frame-phrase] }
- DEFINE [[NEW] SHARED] FRAME frame record [EXCEPT field ...] [DROP-TARGET] { [frame-phrase] }

DEFINE FRAME Statement

QAD
Our Passion. Your Advantage.

FORMAT Phrase

- Specifies one or more attributes for a widget
- Apply to widgets in FORM or DEFINE FRAME
- Syntax

```
[at-phrase ][ AS datatype | LIKE field ][ ATTR-SPACE |
NO-ATTR-SPACE ][ AUTO-RETURN ][ BGCOLOR expression ][
BLANK ][ COLON n | TO n ][ COLUMN-LABEL label  ][
DEBLANK ][ DCOLOR expression ][ DISABLE-AUTO-ZAP ][
FGCOLOR expression ][ FONT expression ][ FORMAT
expression ][ HELP string ][ LABEL label [ , label ] ...
| NO-LABELS ][ NO-TAB-STOP ][ PFCOLOR expression ][
VALIDATE ( condition , msg-expression ) ][ view-as-
phrase]
```

- Example

```
DEFINE FRAME f1
  var_1 AT 20 AS CHARACTER FORMAT "X(18)" HELP "Choose
  carefully." LABEL "Pick One" VIEW-AS RADIO-SET RADIO-
  BUTTONS "&Life", 1, "&Death", 2.
```

QAD Proprietary 56

FORMAT Phrase

QAD
Our Passion. Your Advantage.

02-088.p – 02-090.p

Frame Phrase

- Specifies overall layout and processing properties of a frame for frame definition (DEFINE FRAME and FORM), block header (DO, FOR EACH, and REPEAT), and data handling (DISPLAY, SET, etc.) statements
- Syntax

```
WITH [ACCUM [max-length] [at-phrase] [ATTR-SPACE|NO-ATTR-
SPACE][CANCEL-BUTTON button-name] [CENTERED] [color-
specification][COLUMN expression] [n COLUMNS] [DEFAULT-
BUTTON button-name][[expression] DOWN] [EXPORT] [FONT
expression] [FRAME frame] [KEEP-TAB-ORDER] [NO-BOX] [NO-
HIDE] [NO-LABELS] [USE-DICT-EXPS] [NO-VALIDATE] [NO-HELP]
[NO-UNDERLINE] [OVERLAY][PAGE-BOTTOM|PAGE-TOP] [RETAIN n]
[ROW expression] [SCREEN-IO|STREAM-IO] [SCROLL n]
[SCROLLABLE][SIDE-LABELS] [size-phrase][STREAM stream]
[THREE-D][title-phrase] [TOP-ONLY][USE-TEXT] [V6FRAME
[USE-REVVIDEO|USE-UNDERLINE]] [VIEW-AS DIALOG-BOX] [WIDTH
n] [IN WINDOW window]
```


Frame Phrase

QAD
Our Passion. Your Advantage.

Lab 6

1. Update or create part records in a frame. Continue to use availability, validation, and help. (L0601.p)

2. Enable users to review sales orders for a part if they wish. Use a message box. (L0602.p)

QAD Proprietary 58

QAD
Our Passion. Your Advantage.

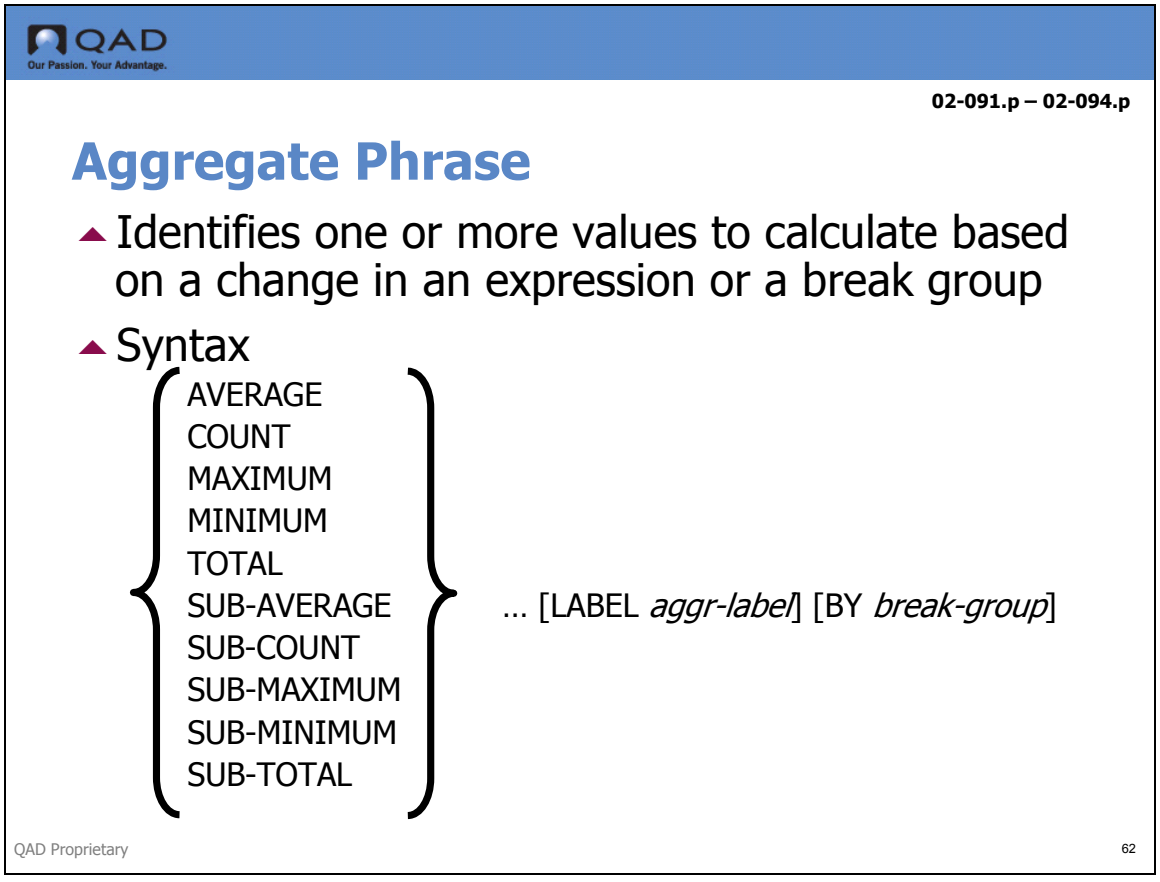
Lab 6 Review

QAD Proprietary 59

Lab 6 Review

CHAPTER 9

Reporting and Input/Output



Aggregate Phrase

Used in conjunction with looping through a FOR EACH

QAD
Our Passion. Your Advantage.

BREAK Group

- Used when you want to conditionalize processing based on when the value of a certain field changes over a series of block iterations
- Use in conjunction with the ACCUMULATE statement and ACCUM function.
- Syntax

```
FOR EACH file WHERE expression BREAK BY field1:
   IF FIRST-OF(field) THEN…
   IF LAST-OF(field) THEN…
END.
```


Break Group

QAD
Our Passion. Your Advantage.

ACCUMULATE Statement

- Calculates one or more aggregate values of an expression during the iterations of a block
- Use the ACCUM function to access the result of this accumulation
- Syntax

  ACCUMULATE {*expression*(*aggregate-phrase*)}
- Example

```
FOR EACH so_mstr, each sod_det where sod_nbr =
  so_nbr
BREAK BY so_nbr:
  ACCUMULATE sod_qty_item * sod_price(TOTAL BY
  so_nbr).
END.
```

QAD Proprietary 64

ACCUMULATE Statement

ACCUMULATE performs the function of accumulating (summing) a value.

ACCUM will access the value the ACCUMULATE statement arrived at.

QAD
Our Passion. Your Advantage.

02-095.p – 02-096.p

ACCUM Function

- Returns the value of an aggregate expression that is calculated by an ACCUMULATE or aggregate phrase of a DISPLAY statement
- Syntax

  ACCUM {aggregate-*expression*}
- Example

```
FOR EACH so_mstr, each sod_det where sod_nbr = so_nbr
BREAK BY so_cust:
  ACCUMULATE sod_qty_item * sod_price(TOTAL BY so_cust).
  IF LAST-OF(so_cust) then
    DISPLAY so_cust
     (ACCUM TOTAL BY so_cust sod_qty_item * sod_price).
END.
```

QAD Proprietary 65

ACCUM Function

QAD
Our Passion. Your Advantage.

Input/Output

- Input
  - The keyboard is the default input
  - You may redirect input from another device
    - FILE / OTHER
- Output
  - The screen is the default output
  - Screen is not a paged device
  - You may redirect output to another device
    - PRINTER / FILE
  - GUI objects only display well on the screen, use STREAM-IO in your frame statement to format the frame for streaming to a text file or printer
- May have multiple INPUT & OUTPUT streams

Input/Output

QAD
Our Passion. Your Advantage.

OUTPUT TO Statement

- Specifies a new output destination
- Use with DISPLAY, EXPORT, PUT statements
- Syntax:

```
OUTPUT [STREAM stream] TO {PRINTER [printer-name]
|opsys-file | opsys-device |  TERMINAL
| VALUE (expression)   |  "CLIPBOARD"  }

[ APPEND ] [ ECHO | NO-ECHO ]  [ KEEP-MESSAGES ]
[ NO-MAP | MAP protermcap-entry ]  [ PAGED ]
[ PAGE-SIZE { constant | VALUE ( expression ) } ]
[ UNBUFFERED ]
[ NO-CONVERT | { CONVERT  [ TARGET target-codepage ]
[ SOURCE source-codepage ] }
```


OUTPUT TO Statement

QAD
Our Passion. Your Advantage.

Redirecting Output

- OUTPUT TO PRINTER
  - redirects the output to the default printer
- OUTPUT THROUGH lp -d *printer-name*
  - typical UNIX printer command
  - only available on shared memory systems
- OUTPUT TO *opsys-file*
  - outputs to a file
- OUTPUT TO TERMINAL PAGE-SIZE 20
  - good for quick testing
  - turns the screen into a paged device
- OUTPUT CLOSE
  - terminates output
  - the program end also terminates output

Redirecting Output

QAD
Our Passion. Your Advantage.

DISPLAY Statement

- Uses default formats and displays it to the current output destination
- Example:

```
OUTPUT TO custfile.
FOR EACH cm_mstr NO-LOCK:
   DISPLAY cm_mstr.
END.
```


DISPLAY Statement

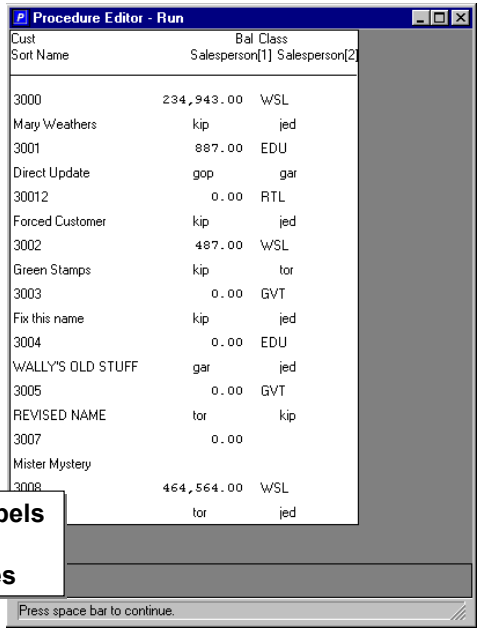
QAD
Our Passion. Your Advantage.

02-097.p

DISPLAY Statement

```
/* 02-097.p */

OUTPUT TO TERMINAL.

FOR EACH cm_mstr:
    DISPLAY cm_mstr
    EXCEPT cm_cr_hold
           cm_cr_limit
           cm_cr_rating
           cm_cr_terms.
END.
```

- **Defaults to column labels**
- **Unquoted strings**
- **Untrimmed field values**



QAD Proprietary 70

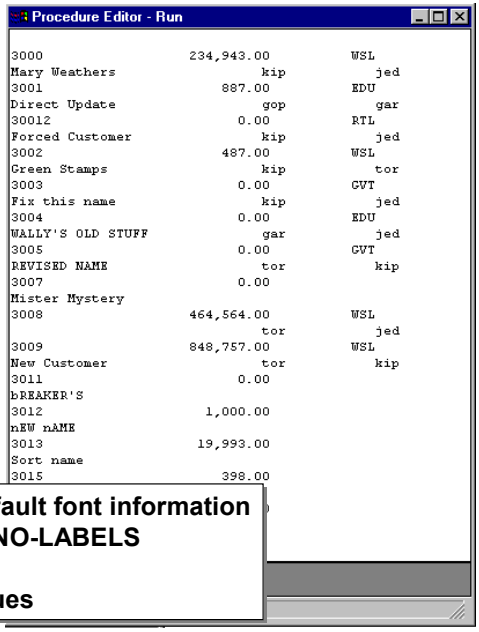
DISPLAY Statement

QAD
Our Passion. Your Advantage.

02-098.p

DISPLAY Statement

```
/* 02-098.p */

OUTPUT TO TERMINAL PAGED.

FOR EACH cm_mstr:
   DISPLAY cm_mstr
   EXCEPT cm_cr_hold
          cm_cr_limit
          cm_cr_rating
          cm_cr_terms
   WITH NO-LABELS.
END.
```



- **PAGED removes default font information**
- **Labels removed by NO-LABELS**
- **Unquoted strings**
- **Untrimmed field values**

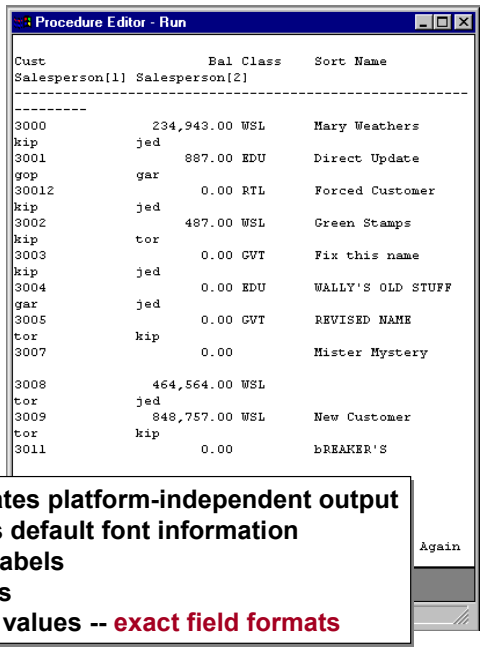
Display Statement

QAD
Our Passion. Your Advantage.

02-099.p

DISPLAY Statement

```
/* 02-099.p */

OUTPUT TO TERMINAL PAGED.

FOR EACH cm_mstr:
   DISPLAY cm_mstr
   EXCEPT cm_cr_hold
          cm_cr_limit
          cm_cr_rating
          cm_cr_terms
   WITH STREAM-IO.
END.
```



- **STREAM-IO creates platform-independent output**
- **PAGED removes default font information**
- **Default column labels**
- **Unquoted strings**
- **Untrimmed field values -- exact field formats**

QAD Proprietary 72

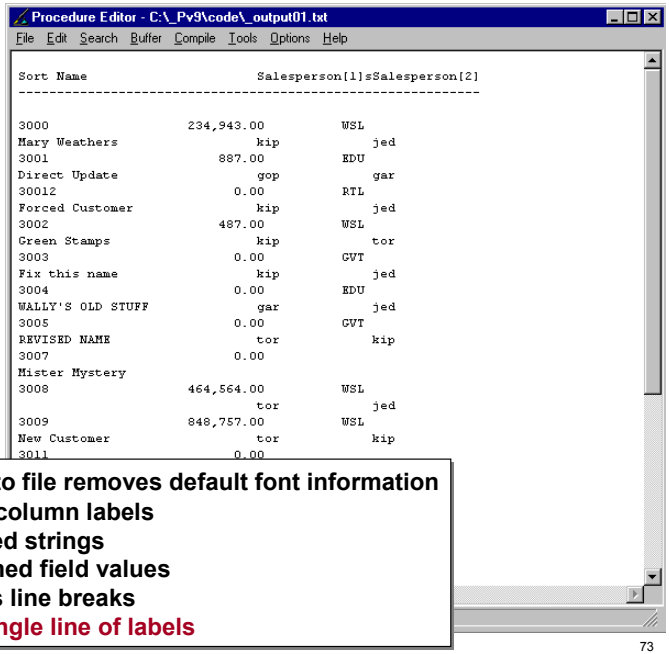
DISPLAY Statement

QAD
Our Passion. Your Advantage.

02-100.p

DISPLAY Statement

```
/* 02-100.p */

OUTPUT TO
  C:\_02-100.txt.

FOR EACH cm_mstr:
  DISPLAY cm_mstr
  EXCEPT cm_cr_hold
      cm_cr_limit
      cm_cr_rating
      cm_cr_terms.
END.
```



- Output to file removes default font information
- Default column labels
- Unquoted strings
- Untrimmed field values
- Includes line breaks
- Note: single line of labels

QAD Proprietary

73

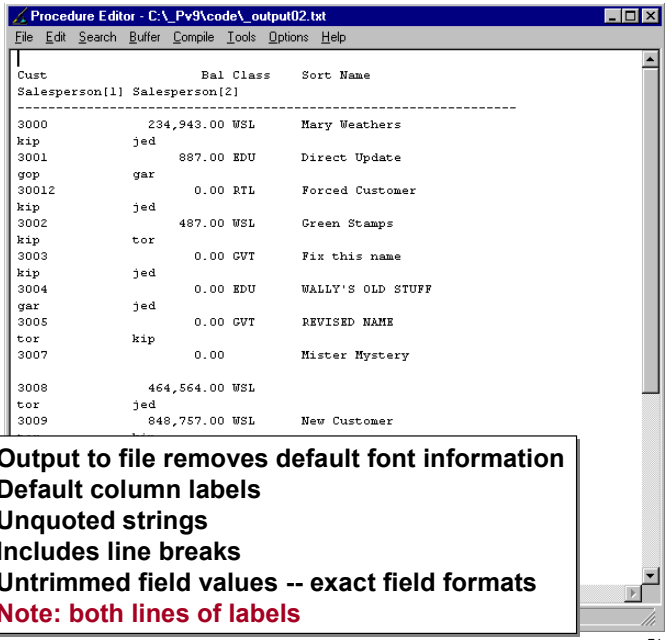
DISPLAY Statement

QAD
Our Passion. Your Advantage.

02-101.p – 02-102.p

DISPLAY Statement

```
/* 02-101.p */

OUTPUT TO
  C:\_02-101.txt.

FOR EACH cm_mstr:
  DISPLAY cm_mstr
  EXCEPT cm_cr_hold
    cm_cr_limit
    cm_cr_rating
    cm_cr_terms
  WITH STREAM-IO.
END.
```



- Output to file removes default font information
- Default column labels
- Unquoted strings
- Includes line breaks
- Untrimmed field values -- exact field formats
- Note: both lines of labels

QAD Proprietary 74

DISPLAY Statement

QAD
Our Passion. Your Advantage.

EXPORT Statement

- Converts data to a standard character format and displays it to the current output destination
- Syntax

```
EXPORT [STREAM stream] [DELIMITER character]
  {expression…|record [EXCEPT field…]}
```

- Example

```
OUTPUT TO custfile.
FOR EACH cm_mstr NO-LOCK:
   EXPORT cm_mstr.
END.
```

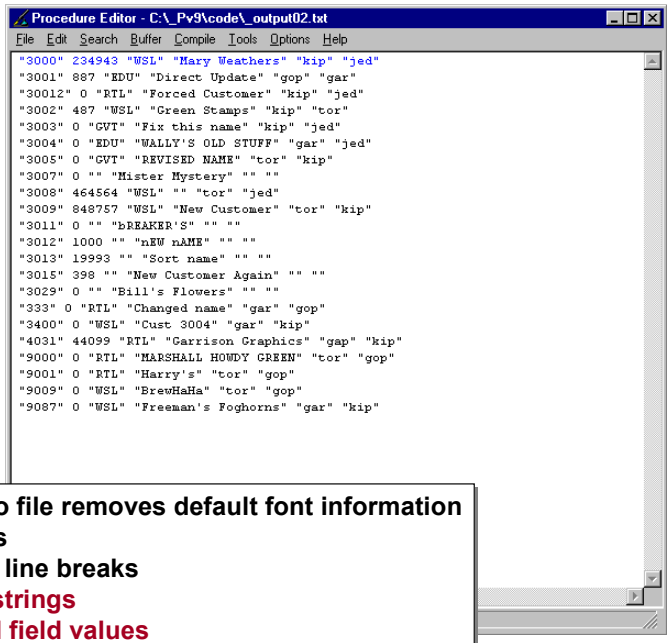

EXPORT Statement

QAD
Our Passion. Your Advantage.

02-103.p

EXPORT Statement

```
/* 02-103.p */

OUTPUT TO
  C:\_02-103.txt.

FOR EACH cm_mstr:
  EXPORT cm_mstr
  EXCEPT cm_cr_hold
    cm_cr_limit
    cm_cr_rating
    cm_cr_terms.
END.
```



- Output to file removes default font information
- No labels
- Includes line breaks
- Quoted strings
- Trimmed field values

QAD Proprietary 76

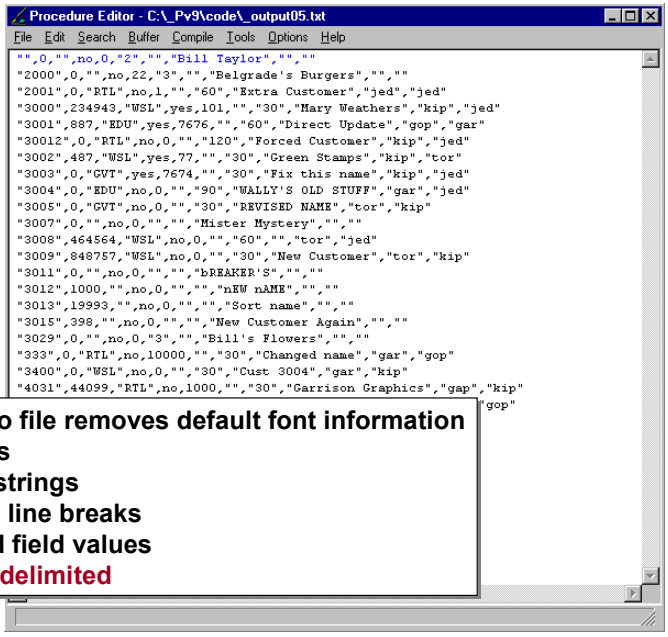
EXPORT Statement

QAD
Our Passion. Your Advantage.

02-104.p – 02-105.p

EXPORT Statement

```
/* 02-104.p */

OUTPUT TO
  C:\_02-104.txt.

FOR EACH cm_mstr:
  EXPORT
     DELIMITER ",”
     cm_mstr.
END.
```



- Output to file removes default font information
- No labels
- Quoted strings
- Includes line breaks
- Trimmed field values
- Comma-delimited

Export Statement

QAD
Our Passion. Your Advantage.

PUT Statement

- Sends the value of one or more expressions to an output destination other than the terminal
- Syntax:

```
PUT [STREAM stream] [UNFORMATTED]
[{expression [FORMAT string] [{AT|TO} expression]}|
{SKIP[(expression)]}|{SPACE[(expression)}]...
```

- Or:

```
PUT [STREAM stream] CONTROL expression ...
```

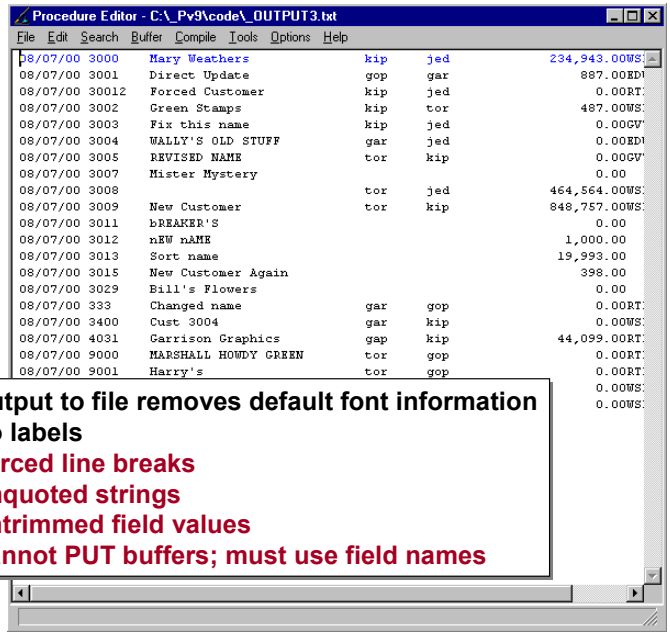

PUT Statement

QAD
Our Passion. Your Advantage.

02-106.p

PUT Statement

```
/* 02-106.p */

OUTPUT TO
  C:\_02-106.txt.

FOR EACH cm_mstr:
  PUT TODAY SPACE
   cm_addr cm_sort
   cm_xslspsn[1]
   cm_xslspsn[2]
   cm_balance
   cm_class SKIP.
END.
```



- Output to file removes default font information
- No labels
- Forced line breaks
- Unquoted strings
- Untrimmed field values
- Cannot PUT buffers; must use field names

QAD Proprietary 79

PUT Statement

QAD
Our Passion. Your Advantage.

02-107.p

PUT Statement

```
/* 02-107.p */

OUTPUT TO
  C:\_02-107.txt.

FOR EACH cm_mstr:
  PUT UNFORMATTED
    TODAY    SPACE
    cm_addr  SPACE
    cm_sort  SPACE
    …
    cm_class SKIP.
END.
```

Procedure Editor - C:_Pv9\code_output06.txt

File Edit Search Buffer Compile Tools Options Help

```
08/08/00 3000 Mary Weathers kip jed 234943 WSL
08/08/00 3001 Direct Update gop gar 887 EDU
08/08/00 30012 Forced Customer kip jed 0 RTL
08/08/00 3002 Green Stamps kip tor 487 WSL
08/08/00 3003 Fix this name kip jed 0 GVT
08/08/00 3004 WALLY'S OLD STUFF gar jed 0 EDU
08/08/00 3005 REVISED NAME tor kip 0 GVT
08/08/00 3007 Mister Mystery   0
08/08/00 3008  tor jed 464564 WSL
08/08/00 3009 New Customer tor kip 848757 WSL
08/08/00 3011 bREAKER'S   0
08/08/00 3012 nEW nAME   1000
08/08/00 3013 Sort name   19993
08/08/00 3015 New Customer Again   398
08/08/00 3029 Bill's Flowers   0
08/08/00 333 Changed name gar gop 0 RTL
08/08/00 3400 Cust 3004 gar kip 0 WSL
08/08/00 4031 Garrison Graphics gap kip 44099 RTL
08/08/00 9000 MARSHALL HOWDY GREEN tor gop 0 RTL
08/08/00 9001 Harry's tor gop 0 RTL
08/08/00 9009 BrewHaHa tor gop 0 WSL
08/08/00 9087 Freeman's Foghorns gar kip 0 WSL
```

- Output to file removes default font information
- No labels
- Unquoted strings
- All formatting forced -- SPACE, SKIP
- Trimmed field values

QAD Proprietary 80

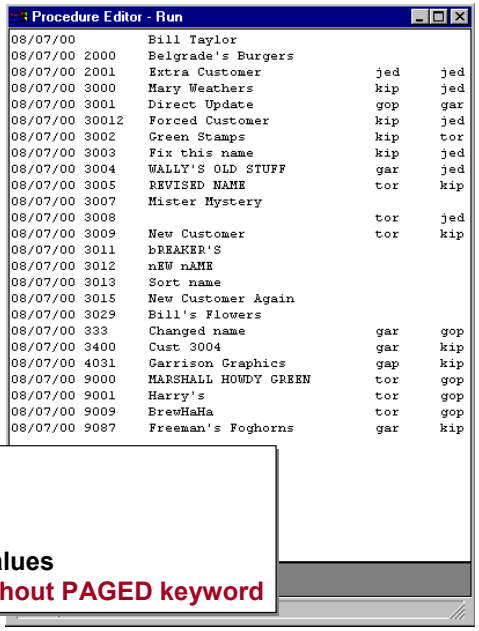
PUT Statement

QAD
Our Passion. Your Advantage.

02-108.p

PUT Statement

```
/* 02-108.p */

OUTPUT TO TERMINAL PAGED.

FOR EACH cm_mstr:
  PUT TODAY SPACE
   cm_addr cm_sort
   cm_xslspsn[1]
   cm_xslspsn[2]
   cm_balance
   cm_class SKIP.
END.
```



- System font
- No labels
- Unquoted strings
- Untrimmed field values
- Will not appear without PAGED keyword

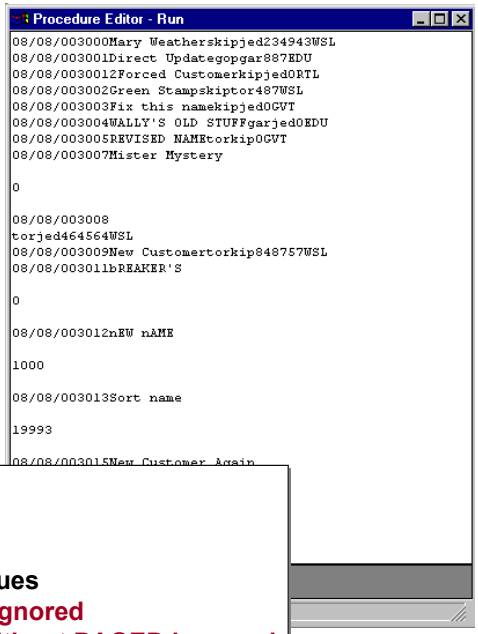
PUT Statement

QAD
Our Passion. Your Advantage.

02-109.p

PUT Statement

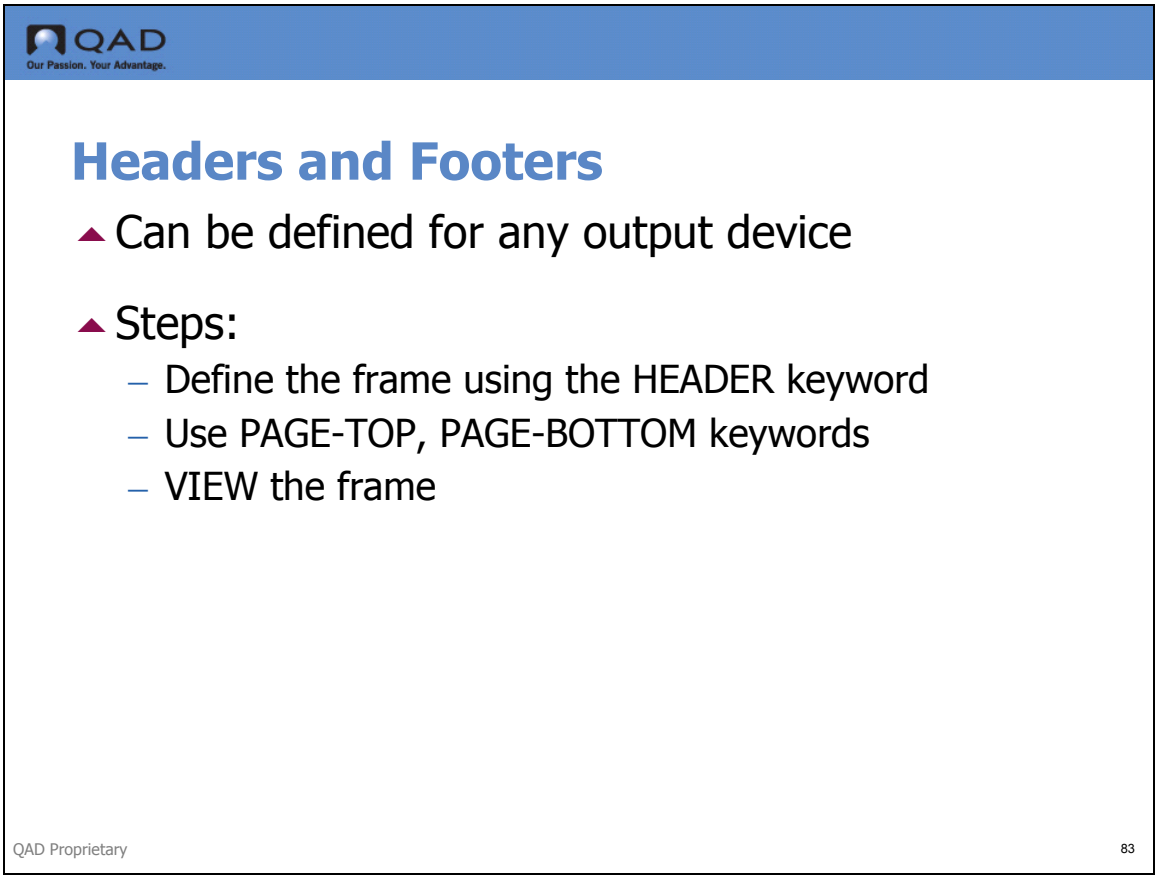
```
/* 02-109.p */

OUTPUT TO TERMINAL PAGED.

FOR EACH cm_mstr:
  PUT UNFORMATTED
   TODAY SPACE
   cm_addr cm_sort
   cm_xslspsn[1]
   cm_xslspsn[2]
   cm_balance
   cm_class SKIP.
END.
```



- System font
- No labels
- Unquoted strings
- Trimmed field values
- SPACE keyword ignored
- Will not appear without PAGED keyword

QAD Proprietary 82

PUT Statement



Headers and Footers

RULES for Headers and Footers:

- Define a frame.
- Use HEADER keyword.
- Use PAGE-TOP keyword.
- VIEW FRAME statement.

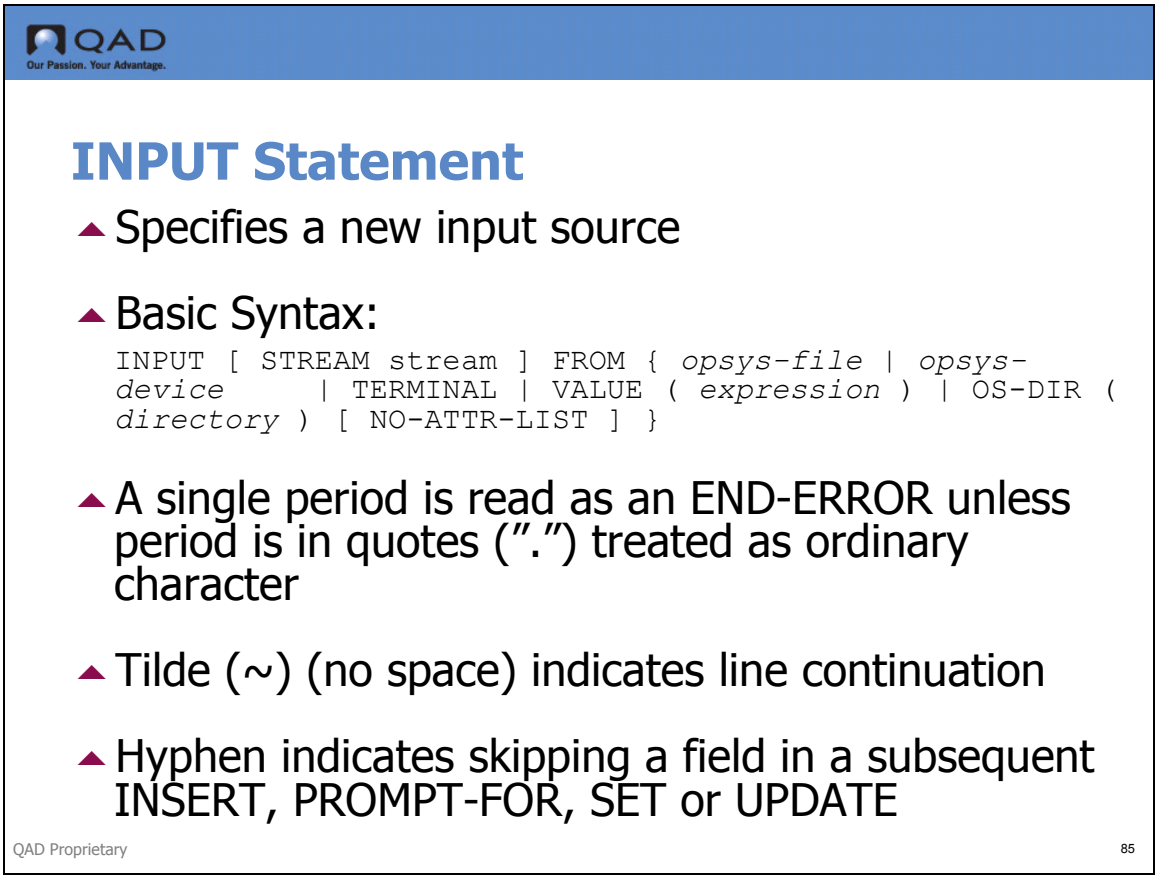
QAD
Our Passion. Your Advantage.

02-110.p – 02-111.p

Output to Multiple Streams

- Define each stream
- Attach each stream to an output device
- Use the streams when they're referenced:
  - `VIEW STREAM s1`
  - `DISPLAY STREAM s2`
  - `PUT STREAM s1`
  - `PAGE STREAM s1`
- For frames in streams, use STREAM-IO option on the Frame Phrase

Output to Multiple Streams



INPUT Statement

Defines a new input source other than the keyboard.

QAD
Our Passion. Your Advantage.

02-112.p – 02-113.p

IMPORT Statement

- Reads a line from an input file that might have been created by EXPORT
- Syntax

```
IMPORT [STREAM stream] {{[DELIMITER character]
  {field|^|record}}|{UNFORMATTED field}} [NO-ERROR]
```

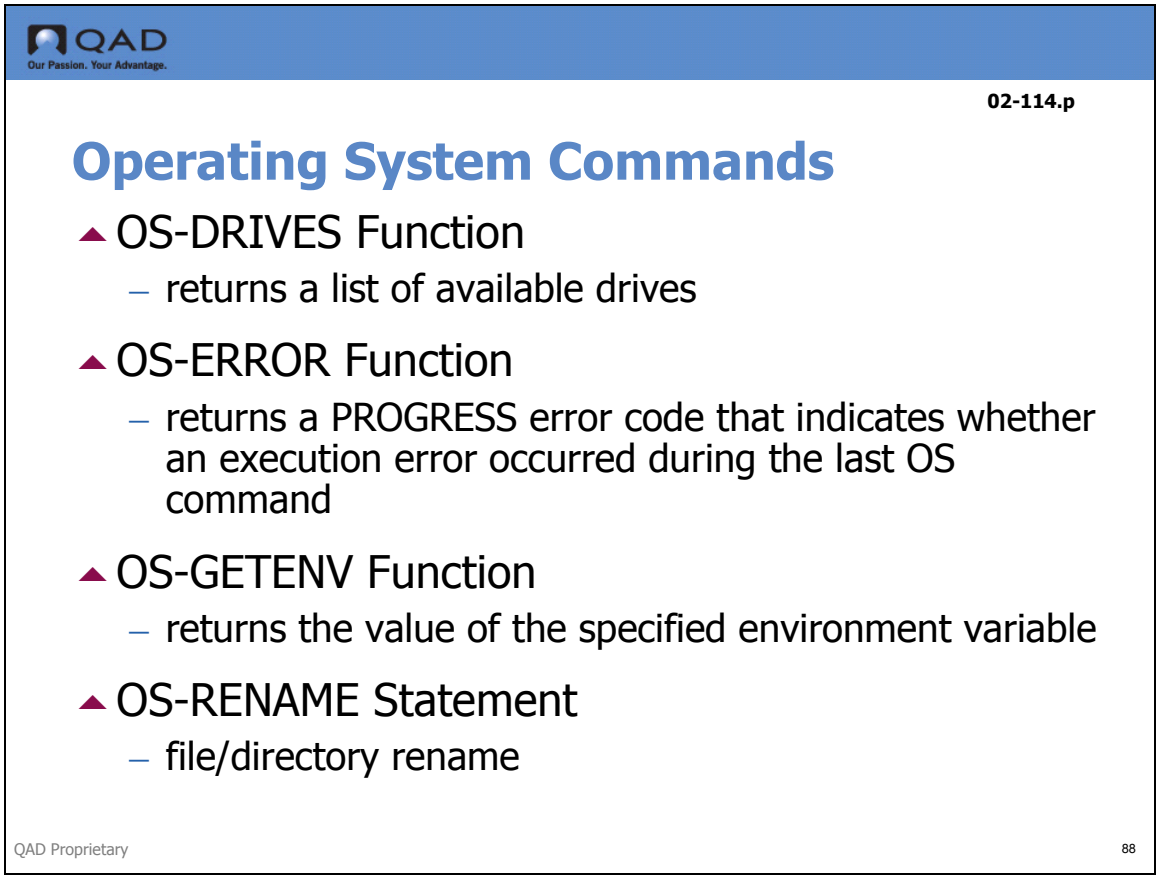
  - STREAM *stream*
    - specifies the name of a stream. If you do not name a stream, PROGRESS uses the default unnamed stream
  - DELIMITER *character*
    - the character used as a delimiter between field values in the file. The character parameter must be a quoted single character. The default is a space character

QAD
Our Passion. Your Advantage.

Operating System Commands

- OPSYS Function
  - identifies the OS being used
- OS-APPEND Statement
  - file append command
- OS-COMMAND Statement
  - escapes to current OS and executes an OS command
- OS-COPY Statement
  - OS file copy command
- OS-CREATE-DIR Statement
  - creates new directory
- OS-DELETE Statement
  - file delete

Operating System Commands



Operating System Commands

QAD
Our Passion. Your Advantage.

Lab 7

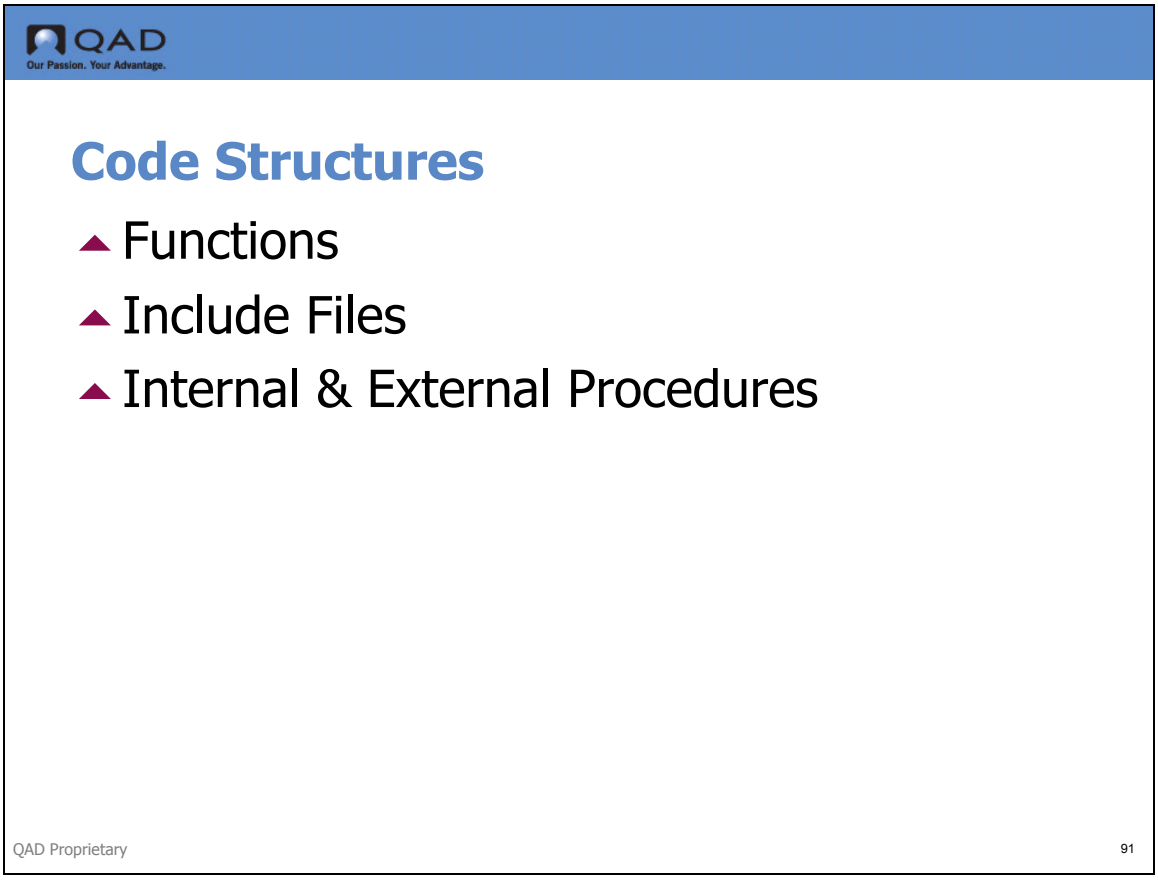
1. Create a sales report BY customer BY order. Send it to the terminal. (L0701.p)
2. Modify 1 to use defined frames. (L0702.p)
3. Modify 2 to use headers and footers (L0703.p)
4. BONUS: Modify 3 by adding a stream showing salesperson information per order to an operating system file. (L0704.p)

QAD
Our Passion. Your Advantage.

Lab 7 Review

QAD Proprietary 90

Lab 7 Review



Code Structures

QAD
Our Passion. Your Advantage.

4GL Functions

- Prepackaged solution, sometimes accepts runtime arguments, returning a value to be used in an expression
  - Arithmetic: RANDOM, EXP
  - Character: STRING, TRIM, LENGTH
  - Date: TODAY, MONTH
  - Validation: AVAILABLE, CAN-FIND, LAST-OF
  - State: CURRENT-CHANGED, PROPATH, CONNECTED
- Global
  - No need to declare the function

4GL Functions

QAD
Our Passion. Your Advantage.

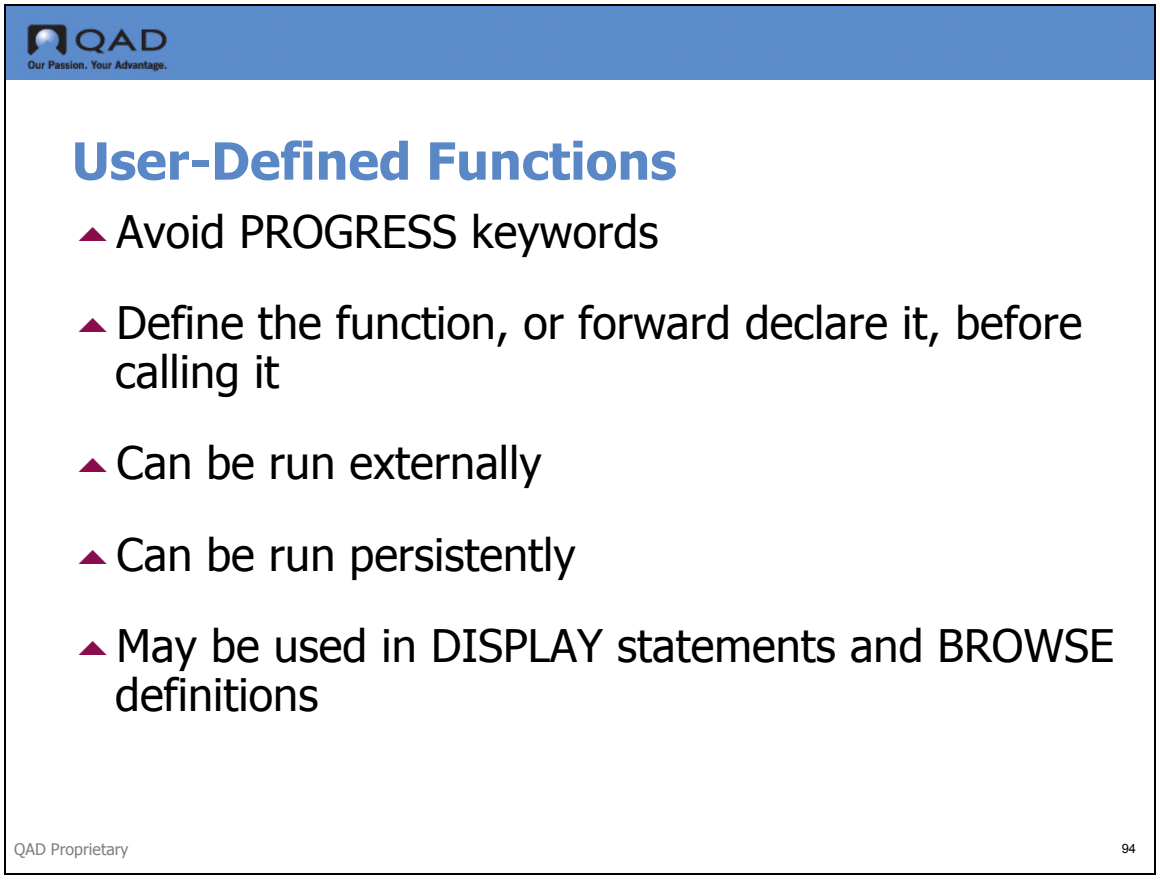
User-Defined Functions

- Let applications define a logical rule or transformation once, then apply the rule or transformation an unlimited number of times
- Syntax

```
FUNCTION function-name [RETURNS] data-type
  [(param[,param]…)] {FORWARD | [MAP [TO] actual-
  name] IN proc-handle}
```

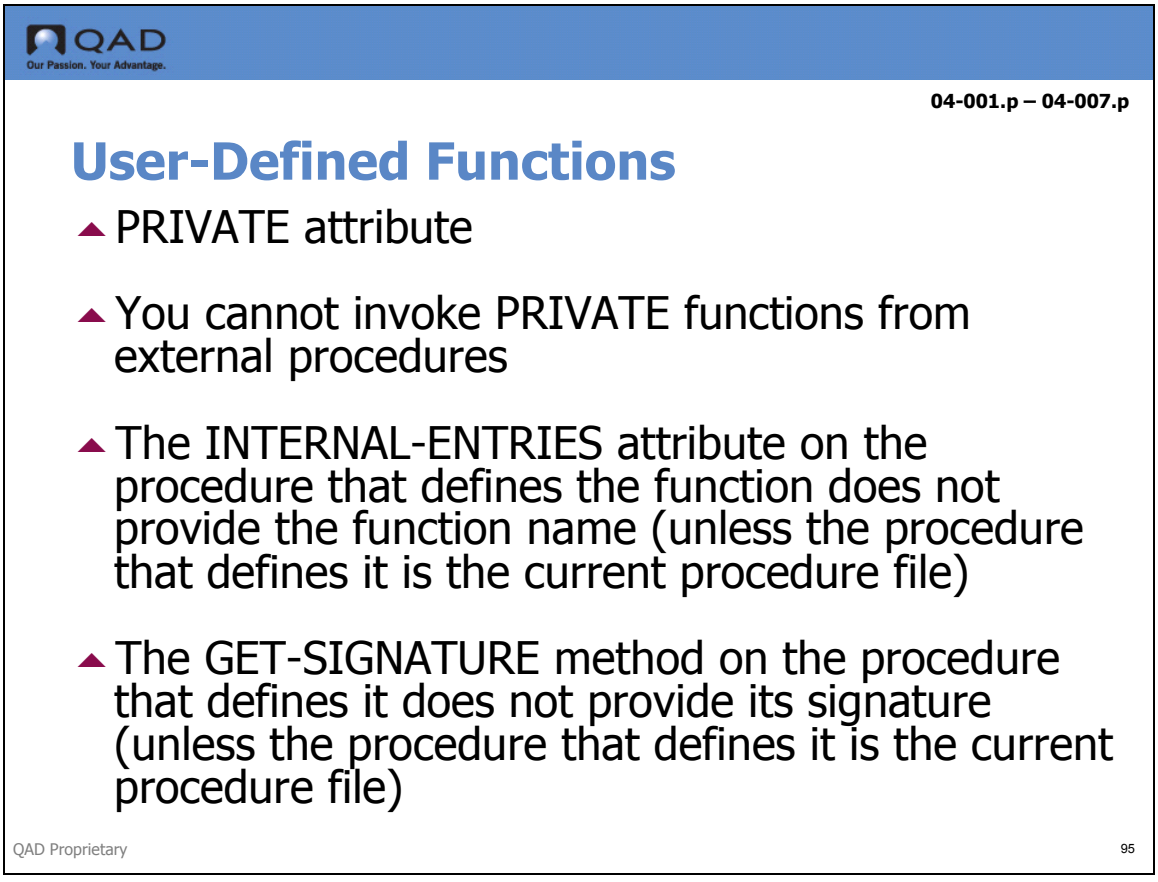
  - returns a single value
  - useful for complex calculations
  - no user interface statements
  - local variables allowed

User-Defined Functions



User-Defined Functions

FORWARD declare allows the function to be defined (referenced) before it is called, but allows the actual structure of the function to be done later (forward) in the program for clarity.



User Defined Functions

This attribute lets you initiate user-defined functions only for the initiating program.

QAD
Our Passion. Your Advantage.

04-008.p – 04-010.p

Include Files

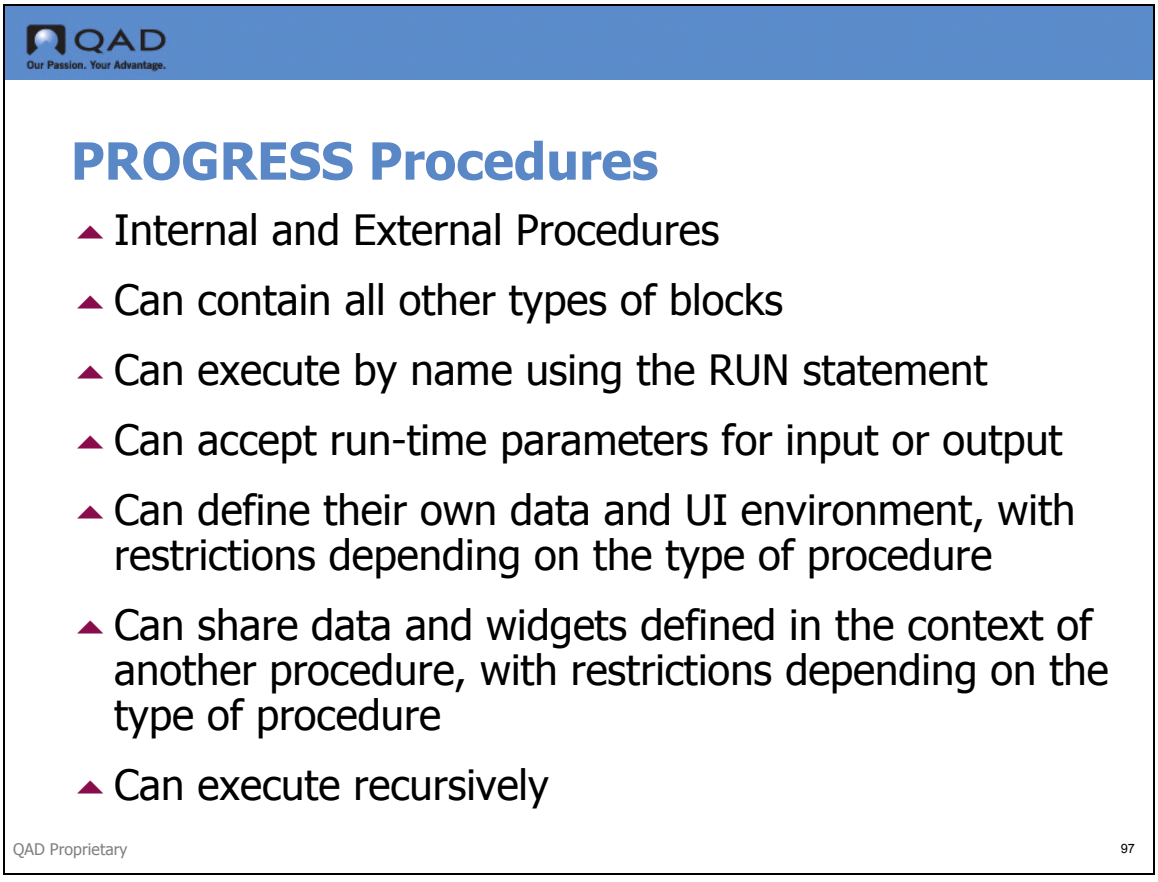
- Causes PROGRESS to retrieve the statements in a file and compile them as part of the main procedure
- Syntax

```
{include-file [argument | &arg-name="arg-value"]…}
```

- One method for reuse of code
- PROGRESS searches the PROPATH to find include files at compile time

QAD Proprietary 96

Include Files



Progress Procedures

Procedure 1 is the main block, calling procedure 2.

Procedure 2 context is invisible to procedure 1.

Procedure 2 can access shared objects or buffers from procedure 1 or parameters passed from 1.

QAD
Our Passion. Your Advantage.

RUN Statement

- Calls a procedure which can be local or remote, external or internal
- Basic syntax
  - `RUN [proc-name|VALUE(extern-expression)].`
- PROGRESS tries to run an internal procedure first
  - if *proc-name* is defined as an internal procedure within the current procedure, PROGRESS first tries to execute the internal procedure
- If an internal procedure is not defined, PROGRESS tries to run an external procedure
  - searches the directories listed in the PROPATH
- V9, searches for super-procedures…

QAD Proprietary 98

RUN Statement

QAD
Our Passion. Your Advantage.

Internal Procedures

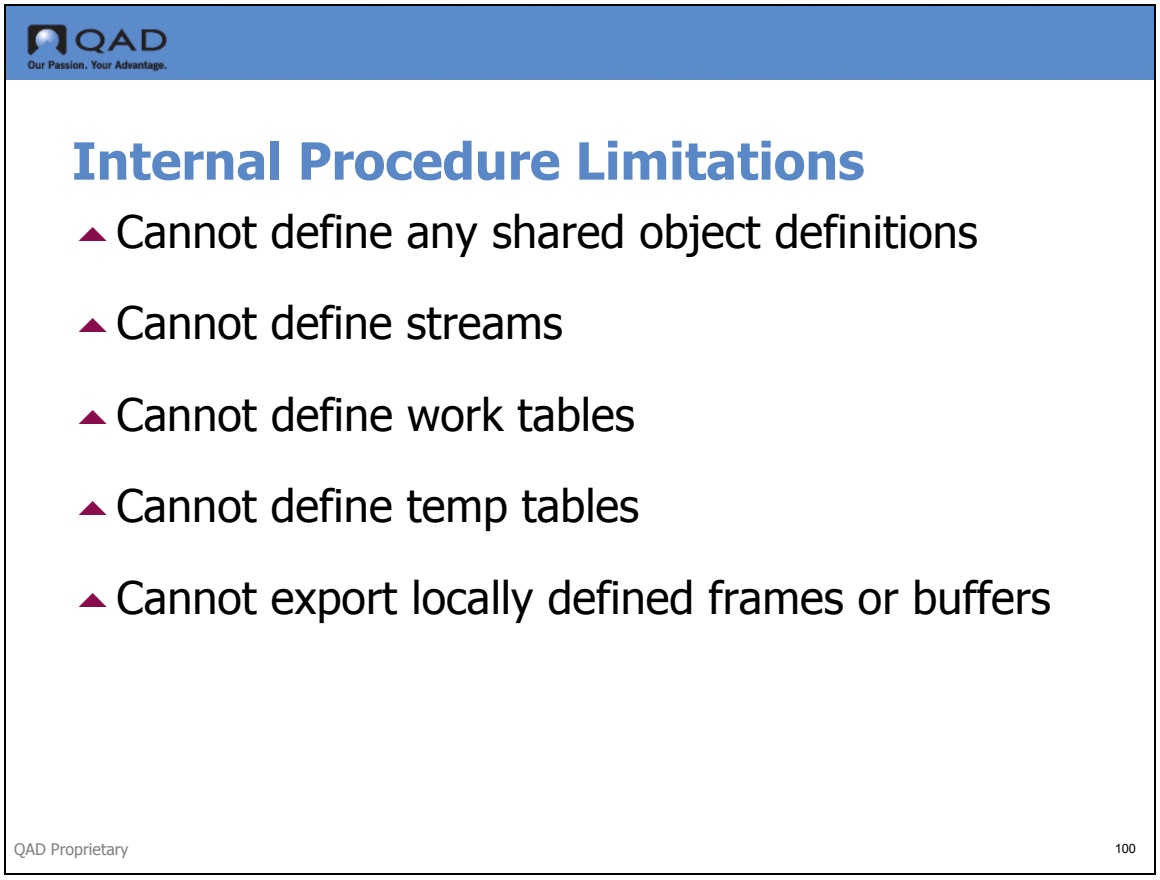
- Block of code defined within an external procedure
- Called with the RUN statement:
- Can be referenced by procedures other than the containing procedure
- Part of the context of the containing procedure
- Can use with parameters or a RETURN-VALUE

```
/* Proc1.p */

…code

RUN proc2.

…


PROCEDURE proc2:
        DO WHILE TRUE:
        WHAT I SAY.
END.
```

QAD Proprietary 99

Internal Procedures

Scope versus context: context is the application environment in which the procedure executes. Scope is the duration a specific context is available.



Internal Procedure Limitations

QAD
Our Passion. Your Advantage.

04-011.p – 04-016.p

RETURN Statement

- Leaves the procedure block and returns to the calling procedure, or to the Procedure Editor if there is no calling procedure
- Syntax

```
RETURN [ERROR|NO-APPLY] [return-value]
```

- Use the RETURN-VALUE function to check the value being returned

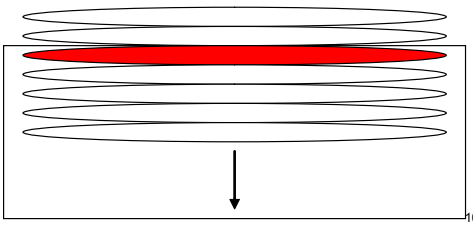
RETURN Statement

QAD
Our Passion. Your Advantage.

External Procedures

- PROGRESS's fundamental procedure
- Created, stored, and compiled separately as an operating system file
- Can be called from any other procedure using the RUN Statement
- Can be run persistently (added to current and future procedure scope and context)

Call stack: If the colored procedure is run persistently, it's functions, variables, internal procedures, etc. are available to programs added to the stack later in the same session.



QAD Proprietary

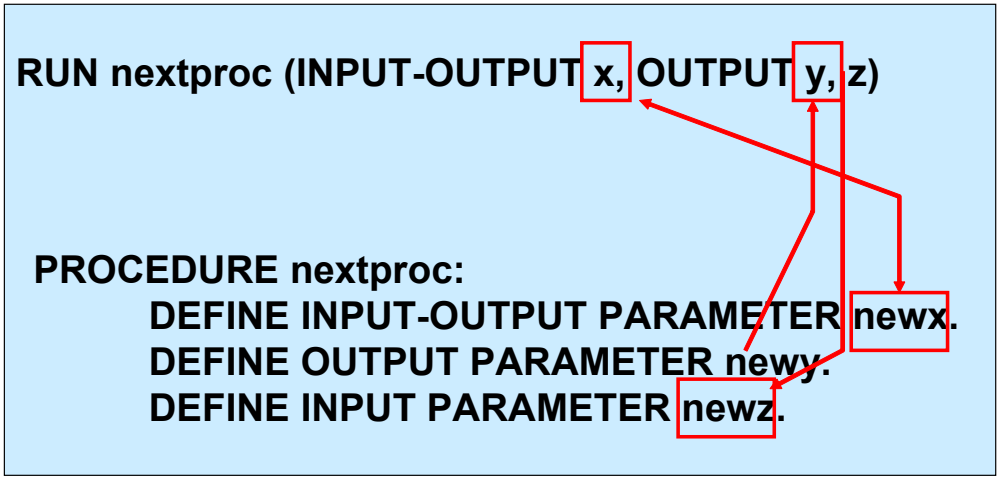
102

External Procedures

Scope versus context: context is the application environment in which the procedure executes. Scope is the duration a specific context is available.

QAD
Our Passion. Your Advantage.

Passing Parameters



Output is requesting output from the called procedure.
Input is providing input to the called procedure.

Passing Parameters

QAD
Our Passion. Your Advantage.

04-017.p – 04-019.p

DEFINE PARAMETER Statement

- Defines a run-time parameter in a sub-procedure or Windows dynamic link library (DLL) routine
- Basic syntax

```
DEFINE {INPUT | OUTPUT | INPUT-OUTPUT} PARAMETER
  parameter {AS datatype}
```

- Parameter is a variable in a called procedure
  - accepts an input value from calling procedure, outputs a value to a calling procedure, or both
- Must specify in the RUN statement in the same order they are defined with the DEFINE statements
  - Datatypes must agree
  - Cannot pass an array as a parameter

DEFINE PARAMETER Statement

QAD
Our Passion. Your Advantage.

04-020.p – 04-023.p

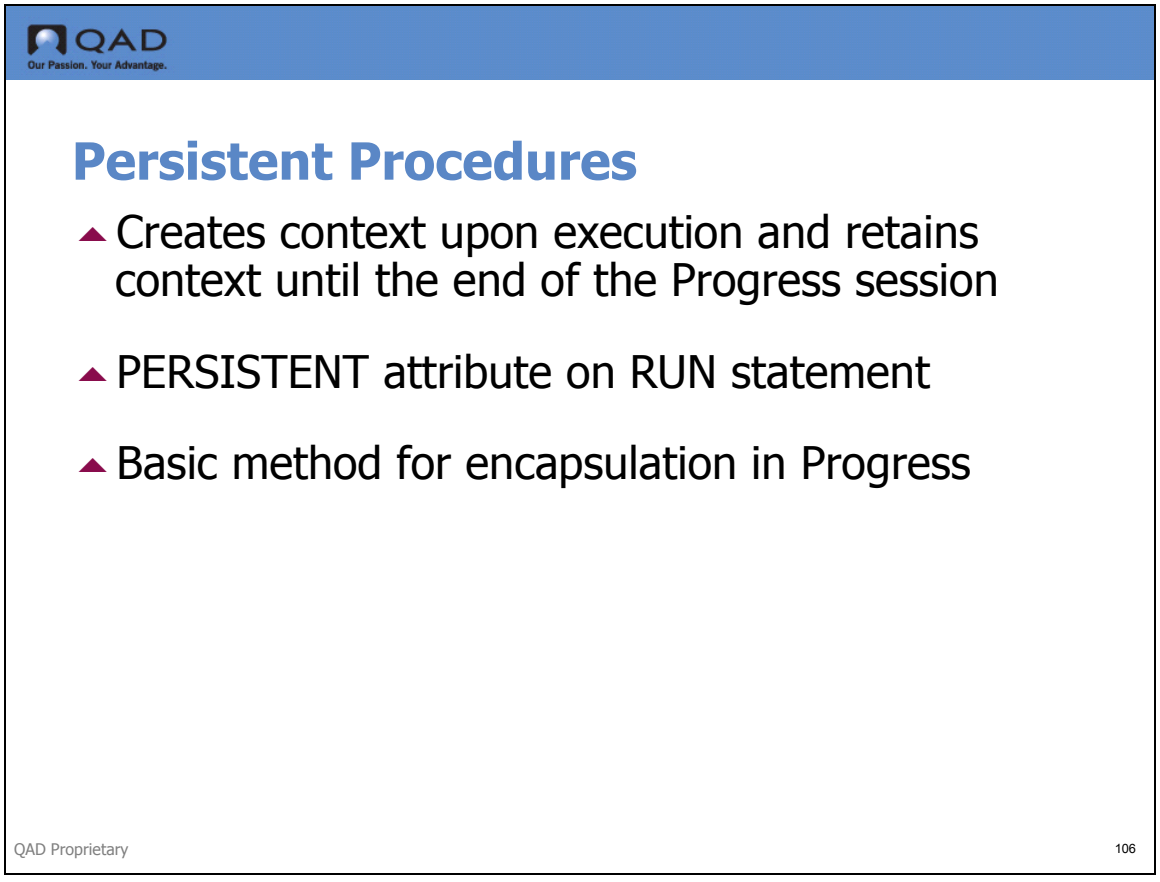
DEFINE VARIABLE Statement

- Shared Variables
  - Global variables are global to your session only
  - Use NEW and GLOBAL in the defining programs
  - PROGRESS searches the call stack looking for the most recent DEFINE NEW SHARED VARIABLE statement that created that shared variable
  - Data types and array extents must match
- Syntax

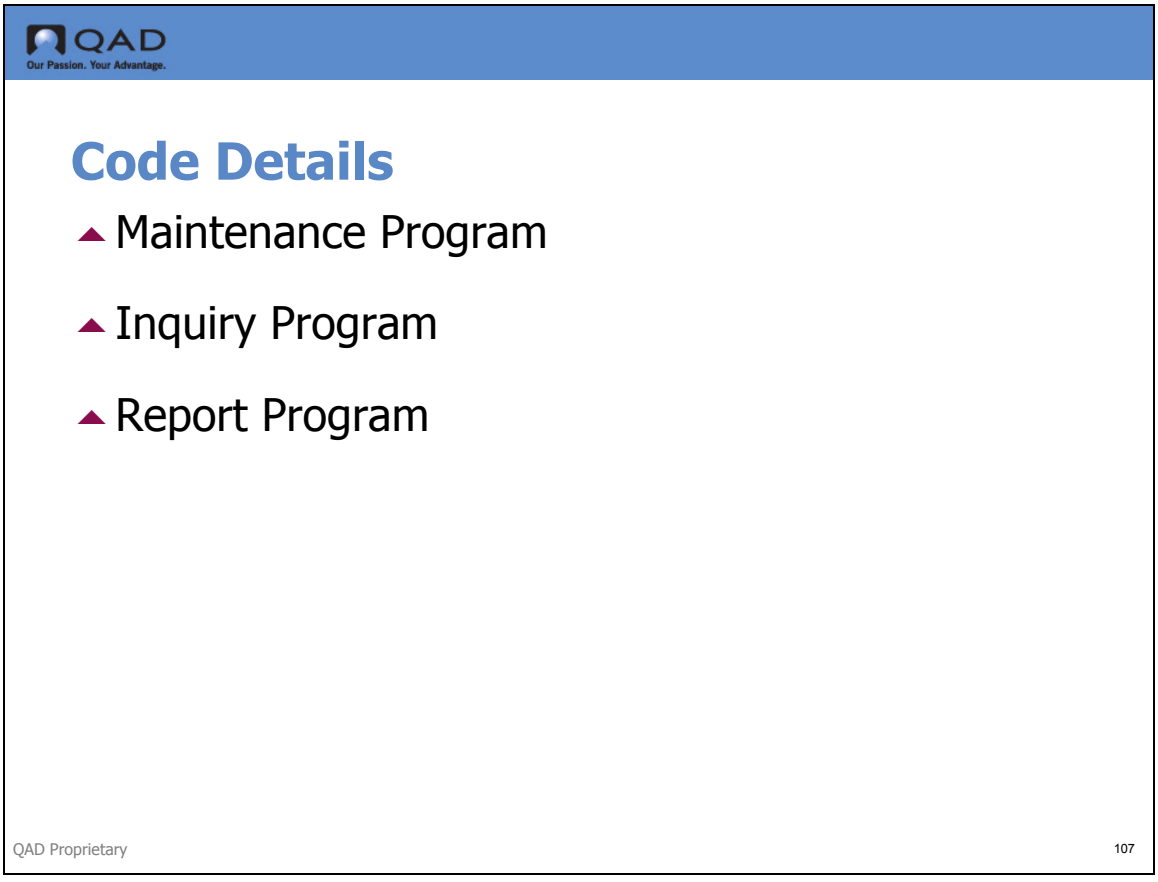
```
DEFINE [[ NEW [GLOBAL]] SHARED] VARIABLE variable { AS
datatype | LIKE field } [NO-UNDO] [BGCOLOR expression]
[COLUMN-LABEL label] [DCOLOR expression]  [DECIMALS n]
[DROP-TARGET] [EXTENT n] [FONT expression] [FGCOLOR
expression]
[FORMAT string] [INITIAL { constant|{[ constant
[,constant] ...]}}] [LABEL string[,string] ...] [MOUSE-
POINTER expression]  [[NOT]CASE-SENSITIVE] [PFCOLOR
expression]{[view-as-phrase]}  {[trigger-phrase]}
```

QAD Proprietary 105

DEFINE VARIABLE Statement



Persistent Procedures



Code Details

QAD
Our Passion. Your Advantage.

Maintenance Program Template

```
/* samplemt.p - Sample Maintenance program  */
/*V8:ConvertMode= Maintenance */
/*REVISION: x.x LAST MODIFIED MM/DD/YY BY: XXX *ECOX**/
{mfdtitle.I "g"}

Mainloop: REPEAT:
   PROMPT-FOR fields…   WITH FRAME EDITING:
        {mfnp.I} /*NEXT/PREV logic */
    END.
     /* ADD/MODIFY/DELETE SECTION */
END. /* MAINLOOP */

STATUS INPUT.
```

QAD Proprietary 108

Maintenance Program Template

QAD
Our Passion. Your Advantage.

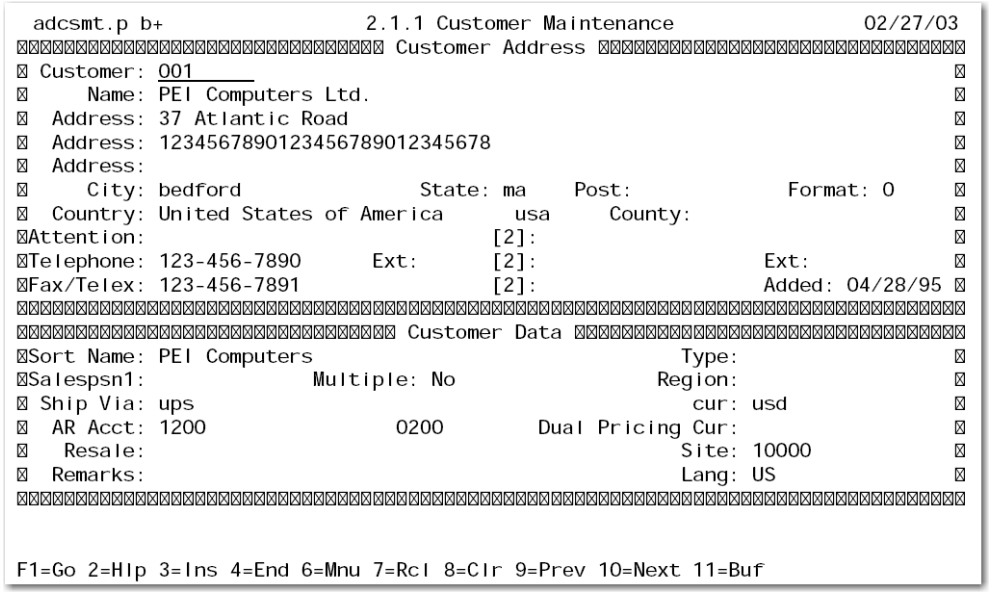
Maintenance Program Template

```
/*ADD/MODIFY/DELETE */
 FIND table WHERE key1 = … NO-LOCK no-ERROR.
 If NOT AVAILABLE table1 THEN DO:
     {mfmsg.I 1 1} /* ADDING NEW RECORD *?
     Create New Record
 END.
 Status = stline[2].
 STATUS INPUT ststatus.
 DISPLAY fields.
 Setloop: ON ENDKEY UNDO blockx, RETRY blockx:
    SET fields …..  GO-ON (F5 CTRL-D).
    IF LASTKEY = KEYCODE(F5) OR …. /*DELETE */
       del_yn = YES.
       {mfmsg01.I 11 1 del_yn}
       if del_yn = YES THEN DO:
          DELETE table1.
       END.
       VALIDATIONS …
  END. /* SETLOOP */
```

QAD Proprietary 109

Maintenance Program Template

QAD
Our Passion. Your Advantage.



QAD
Our Passion. Your Advantage.

```
/* adcsmt.p */


 run p-upd-frameb(input-output reccm,
                  input-output recad  output l_exit).
```

QAD
Our Passion. Your Advantage.

```
/* adcsmt.p */
do transaction:  /* Oracle lock problem */
  loopa:
  do with frame a on endkey undo, leave loopa:

   display cm_addr
    ad_name ad_line1 ad_line2 ad_line
    ad_city ad_state ad_zip ad_format ad_country
    ad_ctry when ({txnew.i} or ec_ok) ad_date
    ad_attn ad_phone ad_ext ad_attn2 ad_phone2 ad_ext2 ad_fax
    ad_fax2 ad_county.
    set1:
       do on error undo, retry:
           set ad_name ad_line1 ad_line2  ad_line3
              ad_city ad_state ad_zip ad_format
              ad_country when ((not {txnew.i}) and (not ec_ok))
              ad_ctry when ({txnew.i} or ec_ok) ad_county
              ad_attn ad_phone ad_ext ad_fax
              ad_attn2 ad_phone2 ad_ext2 ad_fax2 ad_date
              go-on (F5 CTRL-D).
```

QAD
Our Passion. Your Advantage.

```
/* adcsmtp.p */
set
  cm_sort cm_slspsn[1] mult_slsps
  cm_shipvia
  cm_ar_acct
  cm_ar_sub
  cm_ar_cc
  cm_resale cm_rmks cm_type cm_region cm_curr
  cm_scurr when (et_print_dc and
  (new_cmmstr or is-union or is-member or new_curr <> old_curr))
  cm_site
  cm_lang

with frame b.
setb2:
      do on error undo, retry:
        set
          cm_taxable when (not {txnew.i})
            cm_taxc    when (not {txnew.i})
            cm_pr_list2
            cm_pr_list
            cm_fix_pr
            cm_class
            cm_partial
/*N20Y*/      cm__qadl01
         with frame b2.
```

QAD
Our Passion. Your Advantage.

```
 adcsmt.p b+                    2.1.1 Customer Maintenance              02/27/03
⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ Customer Address ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠
⊠ Customer: 001    ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠ Address Tax Data ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠                  ⊠
⊠     Name: PEI Com⊠            Taxable: Yes                ⊠                  ⊠
⊠  Address: 37 Atla⊠           Tax Zone: SA                 ⊠                  ⊠
⊠  Address: 1234567⊠          Tax Class: n                  ⊠                  ⊠
⊠  Address:        ⊠          Tax Usage:                    ⊠                  ⊠
⊠     City: bedford⊠             Tax In: No                 ⊠    Format: 0     ⊠
⊠  Country: United ⊠   Tax ID - Federal: dd                 ⊠                  ⊠
⊠Attention:        ⊠   VAT Registration:                    ⊠                  ⊠
⊠Telephone: 123-456⊠    Tax ID - Misc 1:                    ⊠  Ext:            ⊠
⊠Fax/Telex: 123-456⊠    Tax ID - Misc 2:                    ⊠  Added: 04/28/95 ⊠
⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠    Tax ID - Misc 3:                    ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠
⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠            In City: No                 ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠
⊠       Taxable: Ye⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠                  ⊠
⊠   Price Table:              Invoice by Authorization: No                     ⊠
⊠  Discount Tbl:                                                               ⊠
⊠   Fixed Price: No                                                            ⊠
⊠         Class:                                                               ⊠
⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠


F1=Go 2=Help 3=Ins 4=End 6=Menu 7=Rcl 8=Clr 11=Paste
```

QAD Proprietary 114

QAD
Our Passion. Your Advantage.

```
/* adcsmtp.p */
setb2:
      do on error undo, retry:
        set
          cm_taxable when (not {txnew.i})
            cm_taxc    when (not {txnew.i})
            cm_pr_list2
            cm_pr_list
            cm_fix_pr
            cm_class
            cm_partial
/*N20Y*/      cm__qadl01
         with frame b2.
```

QAD
Our Passion. Your Advantage.

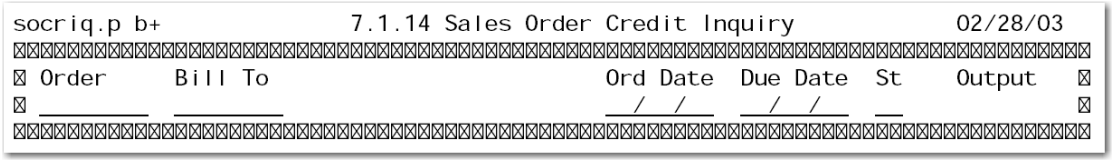
Inquiry Program Template

```
/* samplemt.p - Sample Inquiry program  */
/*V8:ConvertMode */
/*REVISION: x.x LAST MODIFIED MM/DD/YY BY: XXX *ECOX**/
{mfdtitle.I "g"}

REPEAT:
     [get selection criteria] with frame a.
     {mfselprt.I "terminal 80}
     FOR EACH [selection criterial]
          DISPLAY …
          {mfrpchk.I}
   END.
   {mfreset.I}
   {mfmsg.I 8 1}
END.
```

QAD Proprietary 116

Inquiry Program Template

QAD
Our Passion. Your Advantage.



QAD Proprietary 117

QAD
Our Passion. Your Advantage.

```
/* socriq.p */
form
            nbr
            cust
            ad_name  no-label format "x(20)”
            ord_date
            due_date
            act_stat
         with frame a  no-underline attr-space width 80.

         /* SET EXTERNAL LABELS */
         setFrameLabels(frame a:handle).

/*L024*  find first gl_ctrl no-lock.  *L024*/
      repeat:
           if ord_date = low_date then ord_date = ?.
           if due_date = low_date then due_date = ?.
         update nbr cust ord_date due_date act_stat
         with frame a editing:
         {mfselprt.i "terminal" 80}
           if nbr <> "" then
               for each so_mstr no-lock where (so_nbr = nbr)
               with frame c down width 80:
               /* SET EXTERNAL LABELS */
               setFrameLabels(frame c:handle).
               {mfrpchk.i}
```

QAD
Our Passion. Your Advantage.

```
else if cust <> "" then
    for each so_mstr no-lock where (so_bill = cust)    and
       (so_ord_date = ord_date or ord_date = low_date) and
       (so_due_date = due_date or due_date = low_date) and
         (so_stat = act_stat or act_stat = "") with frame d down:


  ...
  display so_bill so_nbr so_conf_date so_due_date so_ship
 so_cr_terms
          open_amt so_stat with frame e width 80 down.
```

QAD
Our Passion. Your Advantage.

Report Program Template

```
/* samplemt.p - Sample Report program  */
/*V8:ConvertMode FULLGUIReport*/
/*REVISION: x.x LAST MODIFIED MM/DD/YY BY: XXX *ECOX**/
{mfdtitle.I "g"}
DEFINE parameter/variable/form definition
REPEAT:
  IF Selectionx = hi_datatype_value THEN selectionx = default_value.
  UPDATE
     {mfquoter.I selectionx}
     {mfhead.I}
     FOR EACH [selection criterial]
          DISPLAY …
          {mfrpchk.I}
   END.
   {mfreset.I}
 END.
```

QAD Proprietary 120

Report Program Template

```
 socrrp.p b+              7.1.15 Sales Order Credit Report             02/28/03

          Sales Or'r: ________                  To: ________
             Bill To: ________                  To: ________
          Order Date:   /  /                    To:   /  /
            Due Date:   /  /                    To:   /  /
       Action Status: __                        To: __

      Include Unconfirmed: No_                          Output:
                                                      Batch ID:

F1=Go 2=Help 3=Ins 4=End 6=Menu 7=Rcl 8=Clr 11=Paste
```

QAD
Our Passion. Your Advantage.

```
Form
             nbr             colon 20
             nbr1            label {t001.i} colon 54 skip
             cust            colon 20
             cust1           label {t001.i} colon 54 skip
             ord_date        colon 20
             ord_date1       label {t001.i} colon 54 skip
             due_date        colon 20
             due_date1       label {t001.i} colon 54 skip
             act_stat        colon 20
             act_stat1       label {t001.i} colon 54 skip(1)
             include_unconf colon 25
with frame a side-labels attr-space width 80.
repeat:
             if nbr1  = hi_char then nbr1 = "".
             if cust1 = hi_char then cust1 = "".
             if ord_date  = low_date then ord_date = ?.
             if ord_date1 = hi_date or ord_date1 = low_date then ord_date1 = ?.
             if due_date  = low_date then due_date = ?.
             if due_date1 = hi_date or due_date1 = low_date then due_date1 = ?.
             if act_stat1 = hi_char then act_stat1 = "".
view frame a.
             update nbr
                    nbr1
                    cust
                    cust1 ...
             with frame a side-labels attr-space width 80.
             bcdparm = "".
             {mfquoter.i nbr       }
             {mfquoter.i nbr1      } ...
```

QAD
Our Passion. Your Advantage.

```
if nbr1 = "" then nbr1 = hi_char.
if cust1 = "" then cust1 = hi_char.
if ord_date = ? then ord_date = low_date.
if ord_date1 = ? then ord_date1 = hi_date.
if due_date = ? then due_date = low_date.
if due_date1 = ? then due_date1 = hi_date.
if act_stat1 = "" then act_stat1 = hi_char.
/* Select printer */
      {mfselbpr.i "printer" 132}
      {mfphead.i}
      last_so_bill = "".
      for each so_mstr use-index so_bill no-lock where
           so_bill     >= cust     and so_bill     <= cust1 and
           so_nbr      >= nbr      and so_nbr      <= nbr1  and
           so_ord_date >= ord_date and so_ord_date <= ord_date1
/*N0TF*    with frame b width 132 */
/*N0TF*/   with frame b width 132 down
       break by so_bill by so_ord_date:
     accumulate open_amt (total by so_bill).
     if last_so_bill <> so_bill then do with frame c:
          /* SET EXTERNAL LABELS */
              setFrameLabels(frame c:handle).
              Display
                cm_addr
                cust_name
                cm_high_date
                cm_cr_hold ...
              with frame c width 132.
```