Experience a new world of interaction

NCR Counterpoint

Course 307 Customizing Counterpoint

©NCR Corporation 2012

For Your Information...

Here is some logistical information to help you find your way around the training facilities:

RESTROOMS	Restrooms are located outside the Training Room on the left. The Women's restroom is immediately to the left and the Men's restroom is further down the hallway.
BADGES	All trainees are required to wear a VISITOR badge during their time in the facility. You will need to scan the badge to obtain access to the Training Room. Please return the badge to the instructor at the conclusion of your training.
PHONES	A telephone is available in the Training Room. Press 9 to get an outside line. Please turn off your cell phone while in class.
WIRELESS ACCESS	The GUEST wireless network provides trainees with access to the Internet.
SMOKING	This is a smoke-free building. A smoking area is provided in a patio area outside the back door of the building. Please use the ashtrays provided.
REFRIGERATOR	Help yourself to the drinks in the refrigerator in the Training Room. A filtered water dispenser is also available in the room. You can also use the refrigerator to store food items that you may have.
LUNCH	The instructor will stop class for approximately one hour to allow participants to enjoy their lunch. There are several restaurants within walking distance of the Memphis training facility.
OFFICES	We are proud of our offices but request that you have an escort to visit areas outside of the Training Room.
APPOINTMENTS	All of our employees schedule their time each day, so you will generally need an appointment to see them. If you need to speak with your Sales Rep or a Support Rep, please notify the Training Coordinator or Training Instructor so that an appointment can be scheduled.

Copyright 1995 – 2016 by NCR Corporation

PROPRIETARY RIGHTS NOTICE: All rights reserved. No part of this material may be reproduced or transmitted in any form or by any means, electronic, mechanical, or otherwise, without permission in writing from NCR Corporation.

SYNCHRONICS and the Synchronics designs are registered trademarks of NCR Corporation, Inc., Memphis, TN.

Other products mentioned are trademarks of their respective manufacturers.

REV: 011916

CUSTOMIZING NCR COUNTERPOINT

Table of Contents

SECTION 1: SQL FUNDAMENTALS

Agenda

What Is SQL?

SQL = Structured Query Language = "Sequel"

Structured Query Language (SQL) gives users access to data in Relational Database Management Systems (RDBMS) by allowing them to describe the data they wish to see. The syntax of statements generally follows the "ANSI Standard" but many enhancements are provided by each manufacturer (Microsoft, Oracle, Informix, etc).

➔ **What is SQL?**

- Industry standard database language
- Transaction based – reduces corruption and loss of data
- Scalable from single-user to very large installations
- Uses server-side processing – optimizes queries
- Reporting and analysis – many SQL tools (e.g., Crystal) are currently available
- Many On-Line Analytical Processing (OLAP) tools available

➔ **Microsoft Enhancements to SQL**

- Stored procedures (sp_attach_DB, sp_detach_DB, etc.)
- Other syntax (for example, “select top 10 from IM_ITEM”)
- English Query
- Enterprise Manager
- Data Transformation Services (DTS)

➔ **Enter and execute SQL statements using either**

- Query Editor (MSSQL Management Studio) or
- SQL Script Utility (NCR Counterpoint)

The SELECT Statement

SELECT statement Use to query the database and retrieve selected data

Format: (**Bold**=required)

SELECT [ALL | DISTINCT] **column1-name**[,column2-name]
FROM table1-name[,table2-name]
[WHERE "conditions"]
[GROUP BY "column-list"]
[HAVING "conditions"]
[ORDER BY "column-list" [ASC | DESC]]

ALL or DISTINCT - return all records or only unique

Examples:

```
SELECT city
FROM AR_CUST
```
Default (assumes "ALL"), can return duplicates

```
SELECT DISTINCT city
FROM AR_CUST
```
Return only unique records

WHERE - return only records that match WHERE condition(s)

Examples:

```
SELECT  nam, adrs_1, city, state, zip_cod
FROM AR_CUST
WHERE categ_cod='MEMBERS'

SELECT nam,adrs_1, city,state,zip_cod
FROM AR_CUST
WHERE nam LIKE 'Ca%'

Select ITEM_NO, DESCR, PRC_1, CATEG_COD
from IM_ITEM
where (CATEG_COD = 'GOLF' or CATEG_COD ='APPAREL')
and PRC_1 < 100.00
```

Operators
= Equal
> Greater than
< Less than
>= Greater than or equal to
<= Less than or equal to
<> Not equal to
LIKE

Note the use of single quotes around text values.

GROUP BY - gather all records that match WHERE condition and optionally perform aggregate function on column

Examples:

```
SELECT  AVG(bal), categ_cod
FROM AR_CUST
WHERE cust_typ='A'
GROUP BY categ_cod

SELECT COUNT(*), categ_cod
FROM AR_CUST
GROUP BY categ_cod
```

Aggregate functions	
Min	return smallest value
Max	return largest value
Sum	return sum of numeric values
Avg	return average value
Count	return total number of values
Count(*)	return number of rows in table

HAVING - use to filter group data; follows GROUP BY clause

Examples:

```
SELECT  AVG(bal), categ_cod
FROM AR_CUST
WHERE cust_typ='A'
GROUP BY categ_cod
HAVING AVG(bal)>400
```

Use HAVING instead of WHERE when using an aggregate function in the condition.

WHERE limits records before they're grouped; HAVING limits groups after they're grouped.

ORDER BY - display results of query in a sorted order

Examples:

```
SELECT  nam, adrs_1, city, state, zip_cod
FROM AR_CUST
WHERE categ_cod='MEMBERS'
ORDER BY zip_cod
```

```
select ZIP_COD
from AR_CUST
order by ZIP_COD
```

Results:

38016
38016
38016
38016
38016
38016
38118
38120
38120
38120
38121

The SELECT Statement

Rules for selecting records from more than one table

- When listing the columns you want to see from each table, include the table name as part of the name.

  Example: SELECT DISTINCT ar_cust.categ_cod, ar_categ_cod.descr

- In the FROM clause, list all tables from which the columns come.

  Example: FROM AR_CUST, AR_CATEG_COD

- In the WHERE clause, "connect" the tables by requiring a matching value in a least one key column from each table.

  Example: WHERE (ar_cust.cust_no <> 'CASH')
  AND (ar_cust.categ_cod=ar_categ_cod.categ_cod)

Simplifying Complex SELECT Statements

Complex conditions can often be simplified with the use of IN or BETWEEN

```
Select ITEM_NO, DESCR, PRC_1, CATEG_COD
from IM_ITEM
where PRC_1 < 100.00
and (CATEG_COD = 'GOLF' or CATEG_COD ='APPAREL')
```

is simplified by the use of **IN**:

```
Select ITEM_NO, DESCR, PRC_1, CATEG_COD
from IM_ITEM
where PRC_1 < 100.00
and CATEG_COD in ('GOLF', 'APPAREL')
```

Use **NOT IN** to output all rows not in the list

```
Select ITEM_NO, DESCR, PRC_1, CATEG_COD
from IM_ITEM
where PRC_1 < 100.00 and PRC_1 > 50.00
```

is simplified by the use of **BETWEEN**:

```
Select ITEM_NO, DESCR, PRC_1, CATEG_COD
from IM_ITEM
where PRC_1 BETWEEN 50.00 and 100.00
```

Use **NOT BETWEEN** to output rows outside the range

Try it Yourself!

This exercise points out the difference between two "amount" columns in the AR Open Items table, and illustrates how to provide a list of receivables payments over $100, including the name and telephone number of the customer and the payment date. Use Query Editor in SQL Management Studio (or the SQL Script Utility) and the DemoGolf database for your work.

1. In SQL Management Studio, in the Object Explorer, expand the tables for your Counterpoint database and then expand the columns for the database table AR_OPN_ITEM.

   Notice that there are two "amount" columns: AMT and ENTD_AMT.

2. Click 

to open a Query Editor window and type the following SELECT statement:

   SELECT * FROM ar_opn_item WHERE doc_typ='p'

   Execute the statement by clicking 

on the toolbar or by pressing F5 (if using SQL Script Utility, click  on the toolbar).

   Look at the difference between the values in the two columns AMT and ENTD_AMT.

3. Now refine the SELECT statement:

   SELECT * FROM ar_opn_item WHERE doc_typ='p' and entd_amt>100

   Execute the statement.

   This statement shows all payments that exceed $100.

4. To provide the rest of the information, you must query not only AR_OPN_ITEM for payments over $100, but you must also query the AR_CUST table to pull the names and phone numbers.

   Replace "*" in the SELECT command with the columns that you need for the list:

Table	Columns
AR_CUST	nam, phone_1
AR_OPN_ITEM	entd_amt, doc_dat

Since two tables are now used in the query, you will need to indicate the table and column name (e.g., AR_CUST.NAM).

When finished, the first line of the command will look like this:

```
SELECT ar_cust.nam, ar_cust.phone_1, ar_opn_item.entd_amt, ar_opn_item.doc_dat
```

5. Include the second table (ar_cust) in the FROM clause.

```
FROM ar_cust, ar_opn_item
```

6. In the WHERE clause, add the table name "ar_opn_item" to the beginning of the column names.

```
WHERE (ar_opn_item.doc_typ='p')
AND (ar_opn_item.entd_amt>100)
```

7. Add an additional condition to the WHERE clause that connects the two tables together on the CUST_NO column.

```
AND (ar_cust.cust_no=ar_opn_item.cust_no)
```

8. The finished SQL query should look like this:

```
SELECT ar_cust.nam, ar_cust.phone_1, ar_opn_item.entd_amt, ar_opn_item.doc_dat
FROM ar_cust, ar_opn_item
WHERE (ar_opn_item.doc_typ='p')
AND (ar_opn_item.entd_amt>100)
AND (ar_cust.cust_no=ar_opn_item.cust_no)
```

Execute the statement.

The results should return five records, four for Bill Baker and one for Brian Schmidt.

	nam	phone_1	entd_amt	doc_dat
1	Bill Baker	321-455-1836	200.00	2001-02-15 00:00:00.000
2	Bill Baker	321-455-1836	150.00	2002-09-30 00:00:00.000
3	Bill Baker	321-455-1836	150.00	2003-01-04 00:00:00.000
4	Bill Baker	321-455-1836	175.00	2004-07-21 00:00:00.000
5	Brian Schmidt	321-456-7788	150.00	2001-02-15 00:00:00.000

Use Aliases to Simplify SELECT Statements

Aliases allow you to use “shorthand” for table names in queries.

Here’s the query from the previous exercise:

```
SELECT ar_cust.nam, ar_cust.phone_1, ar_opn_item.entd_amt, ar_opn_item.doc_dat
        FROM ar_cust, ar_opn_item
        WHERE (ar_opn_item.doc_typ='p')
        AND (ar_opn_item.entd_amt>100)
        AND (ar_cust.cust_no=ar_opn_item.cust_no)
```

And here it is again using aliases:

```
SELECT A.nam, A.phone_1, B.entd_amt, B.doc_dat
        FROM ar_cust as A, ar_opn_item as B
        WHERE (B.doc_typ='p')
        AND (B.entd_amt>100)
        AND (A.cust_no=B.cust_no)
```

In this example, “A” represents AR_CUST and “B” represents AR_OPN_ITEM. Define the alias in the FROM statement of your query.

An alias can be any character(s), and is usually short since its primary purpose is to eliminate typing the same table name over and over in a query.

The UPDATE Command

Use to:
- Perform mass updates to all rows in a table
- Update multiple tables
- Perform calculations on data

Examples:

a) Change the Company checkbox for **Use passwords** to “N”

update SY_COMP set USE_PWDS = 'N'

b) Clear the menu code from a System Security Code

update SY_SEC_COD set MENU_COD = null

c) Change the item category from one value to another

update IM_ITEM set CATEG_COD = 'NEWCAT'
where CATEG_COD = 'OLDCAT'

Most statements use a WHERE clause to specify the updates. This is easily forgotten as you enter new SQL statements. Also, this statement will fail if the new code (NEWCAT in this example) has not already been defined.

d) Set an item attribute or profile code to a specific value

update IM_ITEM set ATTR_COD_1 = 'VALUE'
WHERE ATTR_COD_1 is null

update IM_ITEM set PROF_COD_1 = 'VALUE'
WHERE PROF_COD_1 is null

Use this if you activate a previously unused item attribute or profile. Since existing items will contain NULL for this column, use SQL to set the attribute value to something specific.

The ALTER TABLE Command

Use to:
- Add new columns to a table
- Delete columns from a table
- Change size or properties of a column

Examples:

a) Add the fields **Comment_1** and **Comment_2** to the Vendor table

```
alter table PO_VEND
   add COMMENT_1      T_COMMNT,
       COMMENT_2      T_COMMNT
```

b) Delete the fields **Comment_1** and **Comment_2** from the Vendor table

```
alter table PO_VEND
   drop column COMMENT_1,COMMENT _2
```

c) Rename the **Comment_1 and Comment_2** fields in the Vendor table to **Usr_Comment_1** and **Usr_Comment_2**

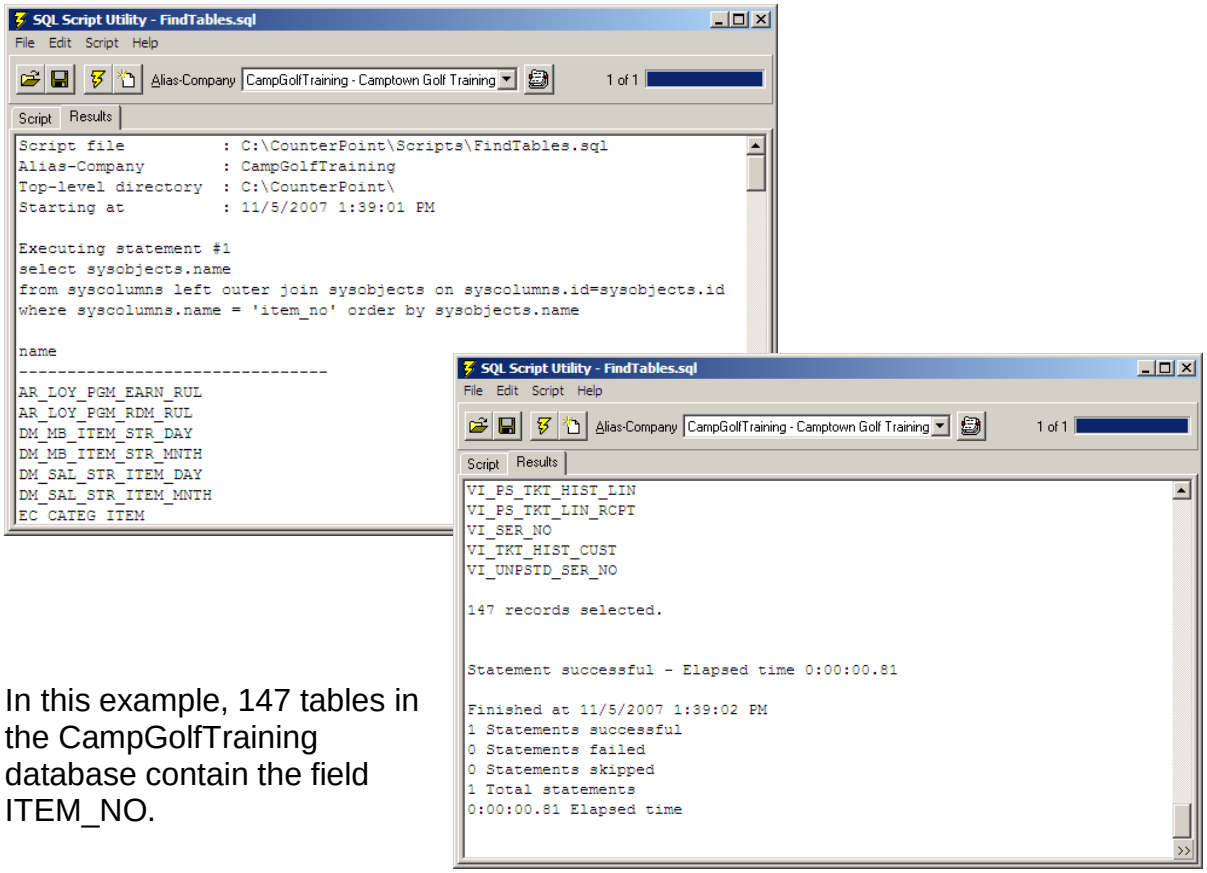
```
exec sp_rename 'PO_VEND.COMMENT_1', USR_COMMENT_1
exec sp_rename 'PO_VEND.COMMENT_2', USR_COMMENT_2
```

Finding all Tables that Contain a Column

If you plan to change a column in all tables that contain the column, you will need an easy way to determine which tables are involved.

This query returns all the table names that contain the column ITEM_NO. It can be run in the SQL Script Utility provided with CounterPoint SQL, in Microsoft's Query Analyzer, or with an OSQL command.

```
SELECT sysobjects.name
FROM syscolumns LEFT OUTER JOIN sysobjects ON syscolumns.id =
sysobjects.id
WHERE syscolumns.name = 'ITEM_NO'
ORDER BY sysobjects.name
```



In this example, 147 tables in the CampGolfTraining database contain the field ITEM_NO.

To further refine the search, you can next print the Column Description Report for just the tables reported by the above SQL script test. On that report, select to show the *Column Code* and *Display Label* columns.

Dangers of SQL Access

SQL is run 'outside' of Counterpoint

- No audit trail - No last maintenance information
- No transaction precautions unless specific steps are taken
- Potential for invalid data (sales rep not on file, prices < 0)

Transaction Precautions

Use an SQL transaction to ensure that all tables are updated before the transaction is committed. Build in error-checking so that if an error occurs while updating a table, the SQL transaction will be rolled back so that no columns are updated. In SQL, an error number of 0 indicates that the update was successful.

```
BEGIN TRANSACTION
UPDATE IM_ADJ_HIST SET ITEM_NO = (Select NEW_ITEM_NO from ITEM_RENUMBER) WHERE ITEM_NO =
(Select OLD_ITEM_NO from ITEM_RENUMBER)
        IF @@ERROR <> 0 GOTO ERROR_COND
UPDATE IM_ADJ_HIST_CELL SET ITEM_NO = (Select NEW_ITEM_NO from ITEM_RENUMBER) WHERE ITEM_NO =
(Select OLD_ITEM_NO from ITEM_RENUMBER)
        IF @@ERROR <> 0 GOTO ERROR_COND
UPDATE IM_ADJ_TRX SET ITEM_NO = (Select NEW_ITEM_NO from ITEM_RENUMBER) WHERE ITEM_NO = (Select
OLD_ITEM_NO from ITEM_RENUMBER)
        IF @@ERROR <> 0 GOTO ERROR_COND
…
…
…

COMMIT
GOTO FINISH_UP
ERROR_COND:
        ROLLBACK TRANSACTION
        GOTO FINISH_UP
FINISH_UP:
        DELETE FROM ITEM_RENUMBER
        DROP TABLE ITEM_RENUMBER
```

DANGEROUS Samples Using SQL:

a) Change the item category from one value to another:

update IM_ITEM set CATEG_COD = 'NEWCAT'
where CATEG_COD = 'OLDCAT'

Similar but DANGEROUS:

update IM_ITEM set CATEG_COD = 'NEWCAT'

DANGEROUS because it sets all items to the new category - probably not desired and something that cannot be undone. Use the WHERE clause to prevent this.

b) Remove every single customer,

delete from AR_CUST

or the entire customer table:

drop table AR_CUST

Sample Queries on Database Structure

DatabaseStructureQueries.sql (in CounterPoint/Scripts)

- example of different queries used to examine database structure
- execute queries in SQL Script Utility

Sample queries provided for:

- domains (including data type and length
- table and view list (names only)
- columns and data type for selected table
- definition of selected view
- indexes in selected table
- all foreign keys
- foreign keys from a selected table
- foreign keys to a selected table
- all triggers
- triggers on a selected table
- content of a selected trigger
- stored procedures (names only)
- parameters to selected stored procedure
- content of a selected stored procedure (with filtering)
- constraints in a selected table

SECTION 2: EXTENDED CUSTOMIZATIONS

Agenda

The mssql.sql Script

- Script that loads empty database with Counterpoint table and column layouts

When to modify mssql.sql:

- Field sizes for a new company need to be changed, usually increased
- Add new physical indexes or sort order (ascending or descending)
- Define new data types, tables, or columns (data types are per company database)

(# commands in below example are treated as commands only in Counterpoint SQL Script Utility and are seen only as comments in standard SQL)

```
#chk_db_empty
#begin dlg_ok_cancel This script will create a new CounterPoint SQL
   database.

Click OK to proceed or Cancel to abort.
#end



sp_addtype T_ACCT_NO, 'varchar(20)'
go


sp_addtype T_ACT_TYP, 'varchar(3)'
go

.
.
.

create table AR_CATEG_COD (
CATEG_COD            T_COD               not null,
DESCR                T_DESCR             null,
DESCR_UPR            T_DESCR             null,
LST_MAINT_DT         T_DT                null,
LST_MAINT_USR_ID     T_USR_ID            null,
LST_LCK_DT           T_DT                null,
ROW_TS               timestamp           null,
constraint PK_AR_CATEG_COD primary key  (CATEG_COD)
)
```

Impact of Customizations

Upgrading to future releases

Before executing UpdateFrom8.x.x script, review **Updating customizations** in Counterpoint Update Guide in online help. Other tools you will use:

1) **Database Customizations Report** - from the prior release, to compare your customized database to the prior version's standard database.
2) **CompareDBResultsFrom84x.htm –** in root directory of Counterpoint download for the new release, to see changes that may affect your customizations.

➡ **Is a table being dropped and a replacement table created?**

If you have added custom fields to the table, they will not automatically be copied to the replacement table and the data in those fields will be gone.

What you need to do: Before running the script, you will need to modify the script to 1) add your custom field to the new replacement table and 2) copy the custom field from the old table to the new one.

➡ **Are multiple tables being merged into a single table?**

What you need to do: If you have added custom fields to a table that is being merged into another table, your custom fields and the data within them will be gone. You will need to add your custom field to a different table and copy the data to the new field before running the script.

➡ **Is a table being dropped (and not replaced)?**

What you need to do: You will need to add your custom field to a different table and copy the data to the new field before running the script.

➡ **Is a new field being added whose name is the same as a custom field you have already added?**

What you need to do: You will need to rename your custom field before running the script. If you run the script first, the error results in the entire transaction being rolled back and other new fields may not get created either. (If you begin custom field and table names with USR_, you will avoid this problem.)

Using Custom Tables and Fields in a Multi-Site Environment

For Multi-Site systems, your schema changes for custom tables and columns need to be distributed to the replicating servers. Create a C:\Custom folder on the Hub server and each remote server, and put the SQL alter script used to create the custom tables and columns in that folder on each server, along with any custom replication rules you may create. The script and rules will be used automatically when Counterpoint is installed on the server or when you run the Activate or Update a Database utility for Multi-Site.

The following steps provide an overview of how to integrate your custom changes. Review the Multi-Site Online Help for complete instructions.

1. If it doesn't already exist, set up a development system that is completely separate from your live Multi-Site/Offline environment.

   The development system needs to have:

   MS SQL Server or SSE 2008 R2 or 2012

Counterpoint	(latest release, with the current service pack)
Counterpoint database	(create a new database, or use an existing one that has been updated to the current release)
DataXtend Studio	(install from Counterpoint Complete download. This provides you with the DXRE Designer which you'll use to modify the replication rules so data in your custom tables and columns replicate to other sites)

Schema Changes

2. Create a SQL alter script that, when applied to your development database, creates your custom tables and columns. Model the script after the standard UpdateFrom8.x.x scripts, but do not check or modify the DB_CTL version. Make a backup of your development database and then apply your script to the database.

3. Use the DataXtend Designer to update the replication rules to include your custom tables and columns. You will also use the DataXtend Designer to optionally assign the custom tables to work sets, as well as to identify any foreign key relationships for your custom tables and columns that you want honored during replication.

4. Copy the custom alter script and custom rules file to the C:\Custom folder on your Hub server and each remote server.

5. Install Counterpoint, selecting either the Hub server role or Remote server role. If Counterpoint is already installed, you can instead use the **Activate or Update Database** utility for Multi-Site (Start / All Programs / NCR Counterpoint / Utilities / Multi-Site / Activate or Update Database). Either of these methods will apply your custom alter script to each database that it creates or updates, and activate or upgrade replication of each database using the custom replication rules file.

Dictionary, Reports, Forms and Other "Top-level" Changes

6. Make any changes to the data dictionary, text dictionary, zoom dictionary, forms and reports on the Hub server so that the changes will exist in the "master" directories for the company.

   If you make dictionary changes on a different computer, you can instead use the DBSys.exe utility (in \Program Files (x86)\RadiantSystems\CounterPoint\CPSQL.1\Bin) to export specific dictionary entries from one system and then import them in to different dictionary files. This avoids simply overwriting entire dictionary files.

7. The File Sync feature of CP Services will automatically synchronize files in the TopLevel\ Configurations folder on the Hub server with each Remote server. Synchronization occurs at 1:00 am each day, as long as the files are not locked (i.e., no one is running Counterpoint).

   File Sync can be run "on demand" if desired from the Management Console.

Using Custom Tables and Fields with Offline Ticket Entry

For Offline Ticket Entry systems, your schema changes for custom tables and columns need to be distributed to the offline workstations.

1. After making the schema changes to the server's database, run the **Provision Database** utility (Start / All Programs / NCR Counterpoint / Utilities / Provision Database).

2. After the database is provisioned, start the Management Console. Highlight each store in the Object Explorer and, on the Summary tab, click the Rebuild button. The database of each offline workstation for a store will be replaced with the new schema after the workstation connects to the server and downloads the new package.

Expanding the Size of a Field

Before database is created

Edit **mssql.sql** script and increase “varchar” or “decimal” value for corresponding data type before executing script

```
sp_addtype T_ITEM_NO, 'varchar(20)'

sp_addtype T_CUST_NO, 'varchar(15)'

sp_addtype T_COST, 'decimal(15,4)'
```

Impact of changes:

- All columns that use the data type will automatically reflect the change.
- Columns on screens will scroll for longer values.
- May need to modify reports (using Crystal Reports) for longer values

Notes

- Do not decrease the size of an existing data type.
- Do not change a data type from “varchar” to “decimal” or vice versa.
- Do not increase the number of decimal digits to more than four; Delphi is limited to four decimals

Expanding the Size of a Field

After database exists

Use Query Editor to execute each step.

1. Create a new user-defined data type with the new column size.

   sp_addtype T_NEW_DESCR, 'varchar (50)'

2. Determine the tables that contain columns which use the old data type (^ = space).

   Select 'alter table^' + Table_Name + '^alter column^' + Column_name + '^T_NEW_DESCR'
   From Information_Schema.Columns
   Where DOMAIN_NAME='T_DESCR' and table_name not like 'VI%'

   This SQL command will generate one or more "alter" statements similar to:

   alter table PS_TKT_PROF_COD alter column DESCR T_NEW_DESCR

3. Modify all columns that use the old data type to the new data type.

   Cut and paste the "alter" statements from the previous command into the query window, and execute each of them.

4. Drop the old data type.

   sp_droptype T_DESCR

   A list of any columns still using the old data type will display, in case any were missed. Execute an "alter table …. alter column …" command for any that were missed, and then attempt to drop the old data type again.

5. If an error message indicates that any VIEWS are using the old data type, force MSSQL to recognize the change to the views.

   sp_refreshview <viewname>

6. Rename the new data type to match the old data type name.

   sp_rename T_NEW_DESCR, T_DESCR

   Renaming the data type to the old name ensures that future “update…” scripts (supplied with and run when installing new releases of Counterpoint) run successfully and use your new column size.

What is a “Custom field”?

Brand-new column in SQL database that is added to an existing System table

Where can the “Custom field” be used?

- Add/edit in standard menu selection for the involved System table*
- Reports and forms created or changed in Crystal Reports
- PS Receipts (OPOS, RDLC, and Crystal formats)
- Lookups (to look up valid field values in a new custom table)
- Filters
- Zoom windows

* custom fields can be added to the standard tables listed here, but cannot be maintained in the standard forms for those tables. They can be maintained in custom forms:

  PS Drawer Sessions (activating, counting, reconciling drawers)
  AR Open Items Apply-to detail
  IM Substitute Items
  IM Price Rules (quantity breaks area)
  IM Grid Definition
  IM Model Stock
  EC Categories

Refer to the Adding Custom Fields to PS Tables chart, later in this section, for similar limitations for some PS document tables and forms.

Adding Custom Fields

Checklist for adding a Custom field

Required steps

1. Add physical field to tables in the database.
2. Set initial value of field in existing records.
3. Update view definitions for associated tables.
4. Add new field to the data dictionary.

Optional steps

5. Regenerate the Quick Report for the table.
6. Use Crystal Reports to customize existing reports to add the field.
7. Customize the table's zoom window to add the new field.
8. Customize Point of Sale receipts to add the new field.
9. Customize filters to add the new field.
10. Customize lookups for the table to add the new field.
11. Customize "grid layouts" to add the new field to the line item areas of ticket entry, purchase request entry, cash receipts entry, or other similar functions.

Commonly Used Counterpoint Data types for columns*

Counterpoint Data Type	SQL Data Type
T_ADRS	varchar (40)
T_AP_VEND_NO	varchar (50)
T_BAT_TYP	varchar (5)
T_BOOL	varchar (1)
T_CITY	varchar (20)
T_CNTRY	varchar (20)
T_COD	varchar (10)
T_COMMNT	varchar (50)
T_COST	decimal (15,4)
T_CUST_NO	varchar (15)
T_DAT, T_DT	datetime
T_DESCR	varchar (30)
T_DOC_NO	varchar (15)
T_EMAIL_ADRS	varchar(50)
T_FILENAME	varchar (100)
T_FLG	varchar (1)
T_GRID_DIM_1 (_2, _3)	varchar (15)
T_GRID_DIM_TAG	varchar (10)
T_IMG	image
T_INT	int
T_ITEM_NO	varchar (20)

Counterpoint Data Type	SQL Data Type
T_LOC_ID	varchar (10)
T_MONEY	decimal (15,2)
T_NAM	varchar (40)
T_NOTE	text
T_NOTE_ID	varchar (20)
T_PCT	decimal (9,3)
T_PHONE	varchar (25)
T_PO_NO	varchar (20)
T_PRC	decimal (15,4)
T_PWD	varchar (15)
T_QTY	decimal (15,4)
T_REF	varchar (50)
T_SER_NO	varchar (30)
T_STATE	varchar (10)
T_TIM	datetime
T_TRX_TYP	varchar (3)
T_UNIT	varchar (15)
T_URL	varchar (100)
T_USR_ID	varchar (10)
T_VEND_NO	varchar (15)
T_WEIGHT	decimal (15,4)
T_ZIP_COD	varchar (15)

* See ***mssql.sql*** script for complete list of all Counterpoint-defined data types.

Varchar	Alphanumeric data, fields with mapping values (pre-defined selection lists), and checkboxes [varchar (1) or simply use T_BOOL]
Int	Whole numbers only, positive or negative. Limit range in Data Dictionary.
Decimal	Numeric data with decimals, positive or negative. “Decimal (15,4)” is 11 digits before the decimal and 4 digits after the decimal. Counterpoint does not support more than 4 digits after the decimal. Limit range in Data Dictionary.
Datetime	Date and time values. Use this for date values. Counterpoint does not support entry of time-only values.
Text	Long alphanumeric values, like notes. Will not display in lookups and cannot be used in filters.

Adding Custom Fields

Required steps –Example 1

1. Add physical field to tables in the database.

Use Query Editor or SQL Script Utility to add the new physical field with its appropriate data type to the table in the database. Avoid using spaces in your column names.

```
alter table  AR_CATEG_COD
     add  USR_COMMENT_1          T_COMMNT  null,
          USR_COMMENT_2          T_COMMNT  null,
          USR_COMMENT_3          T_COMMNT  null
```

2. Set the fields to appropriate initial values in existing records.

Since the initial value for all three fields is Null, it is not necessary to use the "update" command to specifically set the initial values in any existing record or to set a default constraint.

3. Update view definitions for associated tables.

All view definitions that use the table with the custom field need to be updated. Use Query Editor or the SQL Script Utility to load and execute the **RefreshViews.sql** script supplied with Counterpoint.

4. Add new field to the data dictionary.

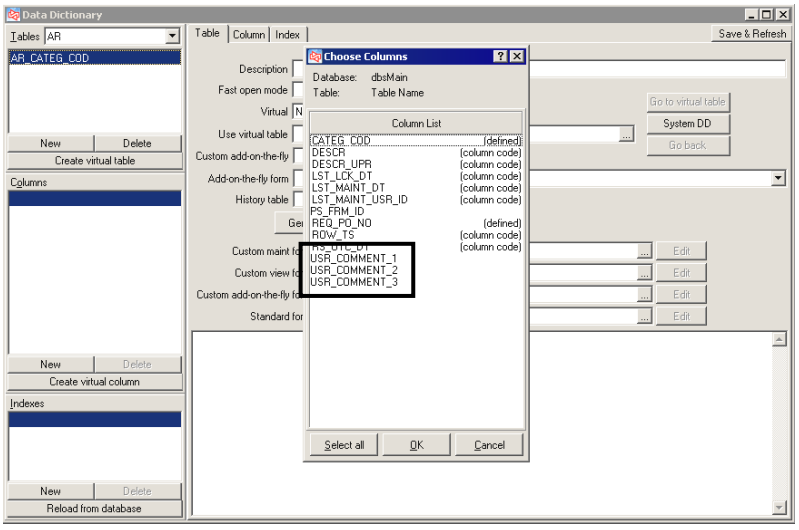
Select **Setup / System / Configuration / Data Dictionary**.

In the System Data Dictionary, select to customize the table.

Then, on the Data Dictionary form, click New under **Columns**.

If asked, specify a physical table name as a template.

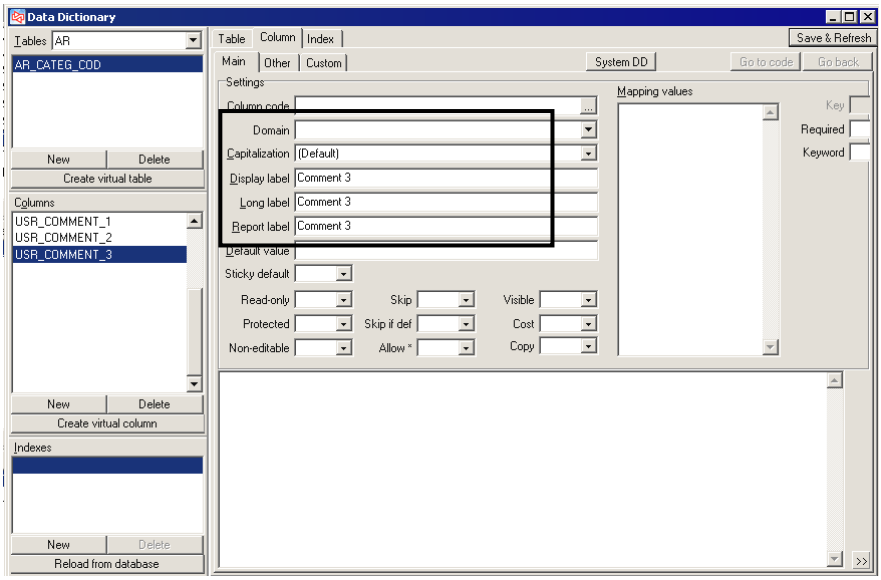
Then select your new column from the list.



On the Main tab:

Set the **Display label**, **Long label**, and **Report label** for the new column.

Also, if necessary, assign a **Domain**. A blank domain means that the field will allow any alphanumeric characters to be entered.

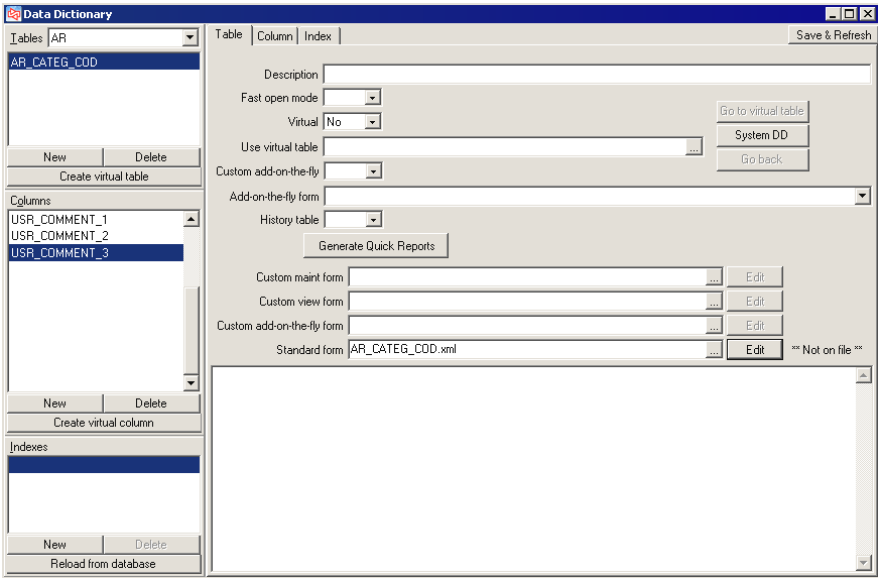


On the Table tab:

If you want the new field to appear on the standard form for the table, enter the name of a form layout file at **Standard form**.

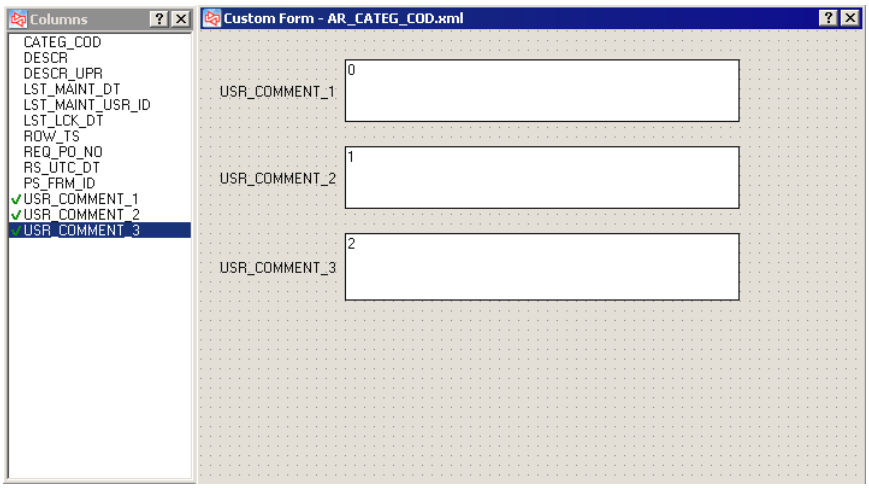
The form layout file will automatically be created if it does not already exist.

Click Edit to open the form layout editor.



Select the new field from the list of all fields in the table and drag it to the desired position on layout form. Change the size of the entry field if desired.

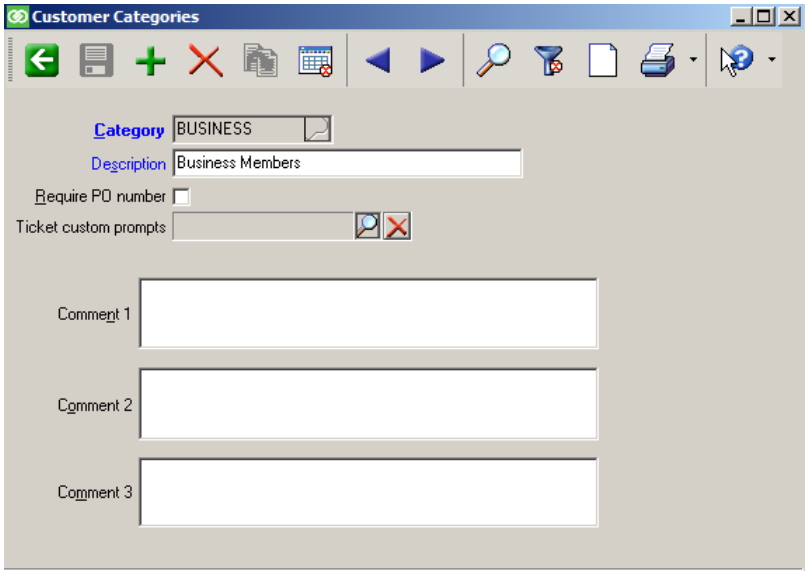
Close the layout form when done to save your changes.



Click Save & Refresh to save your entry to the Data Dictionary.

The new column should now appear in the menu selection that displays data from the involved table.

You may need to resize the form to see the new column.



Data Dictionary Domains

- Directs the type of control shown for a custom field

Counterpoint Data Type	Data Dictionary Domain	Use / Notes
<any "decimal" SQL type>	**NUMERIC**	Allows numeric value, including decimals
<any "varchar" SQL type>	**<Blank>**	Allows any string value
T_BOOL	**BOOLEAN**	Displays as checkbox
T_DAT or T_DT	**DATE or DATETIME**	Provides dropdown calendar
T_DAT or T_DT	**TRXDATE**	Requires the value to be in the authorized date range at time of entry
T_EMAIL_ADRS	**EMAIL**	Value displays in blue, underlined Double-click to launch email editor
T_FILENAME T_FILENAME50	**FILE**	Displays browse button, and stores full path and filename in field
T_IMG	**IMAGE**	Displays in box, proportionally scaled if too large. Right-click control to load .bmp, .png, .jpg file or to save as .bmp.
T_NOTE	**RTF**	Stored in rich text format; also stores as plain text if <fieldname>_TXT exists
T_PWD	**PASSWORD**	Data entry and display are masked with asterisks (stored in database as plain text)
T_TIM	**TIME**	Time entry for hours and minutes
T_TIM	**LONGTIME**	Time entry for hours, minutes, and seconds
T_URL	**URL**	Value displays in blue, underlined Double-click to open URL
<any "varchar" SQL type>	**PRINTER**	Displays default Windows printer, and printer selection list

Selecting columns

- List shows all columns that are physically in the database
- Form grid is 8 pixels by 8 pixels; cannot change setting or turn grid off
- Drag or double-click column to add to layout form
- Cannot select multiple columns from list at the same time
- Green check indicates column is already on form layout
- Right-click on column list to see display labels of columns
- Initial default size of column's control is based on type and size of column
- Display label will always appear to left of its control
- Number inside field denotes its tab order within all custom columns on the form; default order based on sequence in which fields were added to layout form
- "Not visible" columns can be added to layout, but won't show up on actual form
- "Cost" column can be added to layout, but won't appear for unauthorized users
- Close form to save changes (no Save button). Saved as XML file in TopLevel/ Configuration / <company> / CustomForms

Editing

- Change tab sequence of custom columns by right-clicking form. Choose "Guess tab order" (sets in left-to-right reading order), or choose "Set tab order" and then click controls in desired sequence (reselect "Set tab order" to exit). "Skip" settings in Data Dictionary still control whether tab stops at a column.
- Hovering over control shows display or field name (whichever isn't shown), as well as numerical position and size
- Clicking on column in list doesn't select the column on the layout form
- Can move a control by dragging it or using arrow keys
- Resize control by grabbing handle and dragging; 8-pixel increments horizontally, 1-pixel increments vertically
- Can multi-select controls by shift-clicking or dragging mouse around controls

Adding Custom Fields

How does user enter data into Custom field?

Type of Standard form	Custom Field appears
Maintenance with tabs (e.g., Inventory / Items, Customers / Customers)	On a “Custom” tab
Maintenance without tabs (e.g., Setup / Customers / Customer Categories)	Below other fields
Document header with tabs (e.g., Purchase Requests / Enter)	On a “Custom” tab
Document header without tabs	On a “Custom” tab

Special Forms	Access to Custom Field	
	Simple toolbar	Default toolbar
PS Ticket Header*	Additional Functions / Enter profile	Ticket / Ticket profile
PS Ticket Lines*	Additional Functions / Enter item prompts	Line / Enter item prompts
PS Ticket Payment*	Payment Validation	Payment Validation
PO Purchase Request Lines	Additional Functions / Line delivery and comments	Line / Other info
PO Receiving Lines	Additional Functions / Line comments	Line / Comments
AR Cash Receipt Lines (apply-to’s)	Additional Functions / Enter discount amount	Line / Enter discount amount
IM Transfer Out Lines	Additional Functions / Enter comments	Line / Comments

* Set "Prompt for custom fields?" to **Yes** in individual customer records, item records, or pay code records to force custom fields in the PS Ticket Header, PS Ticket Line, or PS Payment Validation to be automatically requested for a customer, item, or pay code.

Try it Yourself!

This exercise shows how to add custom fields to five different types of forms in Counterpoint. Use Query Editor and the DemoGolf database for your work.

Example 1: Maintenance Form without Tabs

Add "Comment-1", "Comment-2", "Comment-3" to **Setup / Customers / Categories**, for purposes of spelling out the return policy for each customer category.

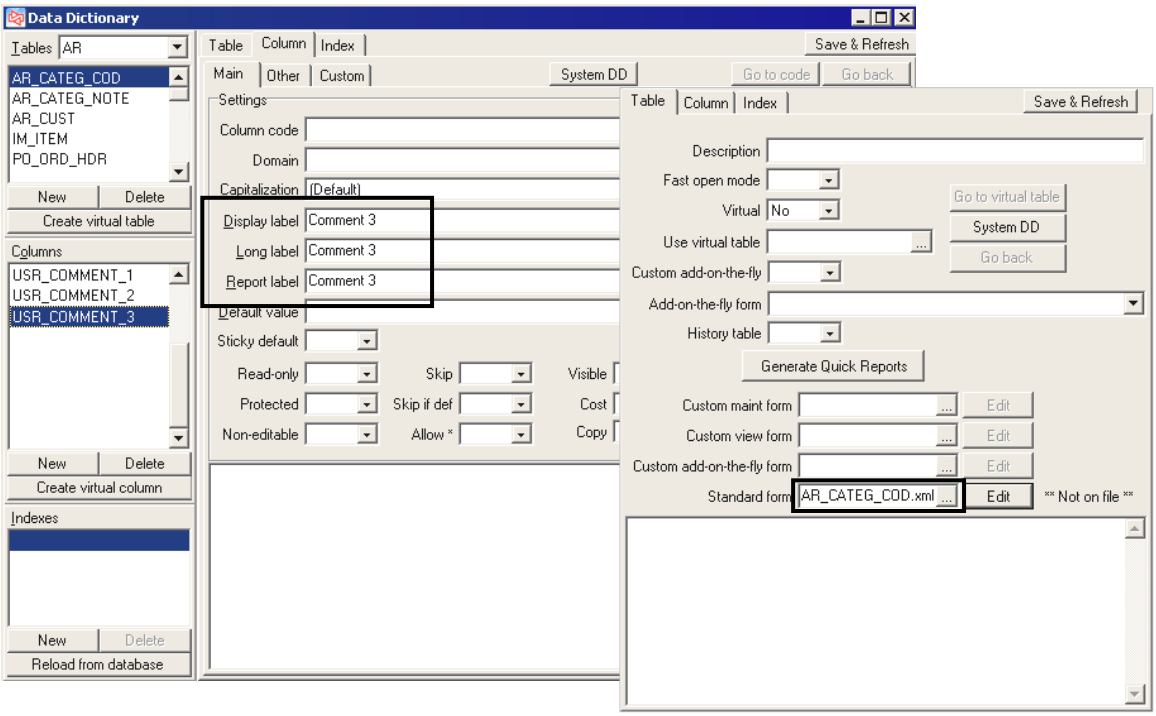
1. In Query Editor, add the three new fields to the AR_CATEG_COD table.

```
alter table  AR_CATEG_COD
       add  USR_COMMENT_1          T_COMMNT,
            USR_COMMENT_2          T_COMMNT,
            USR_COMMENT_3          T_COMMNT
```

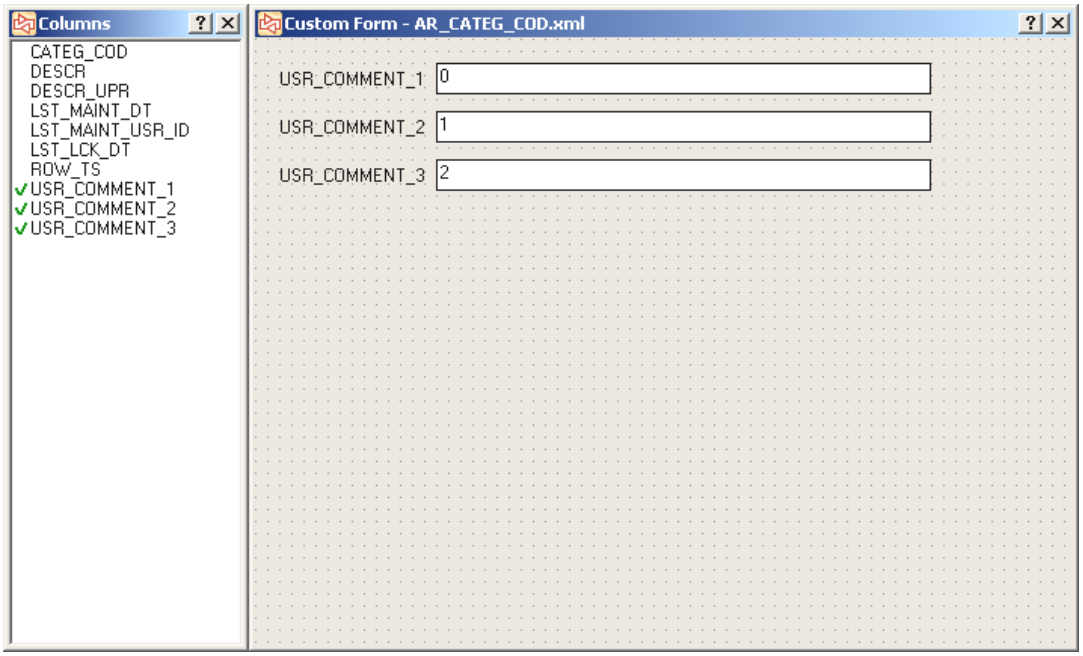
   Since the initial value for all three fields is Null, it is not necessary to set a default constraint or use the "update" command to specifically set the initial values.

   Load and execute **RefreshViews.sql** to update any views that use the AR_CATEG_COD table.

2. In **Setup / System / Configuration / Data Dictionary**, customize the AR_CATEG_COD table and add the three fields to the data dictionary.



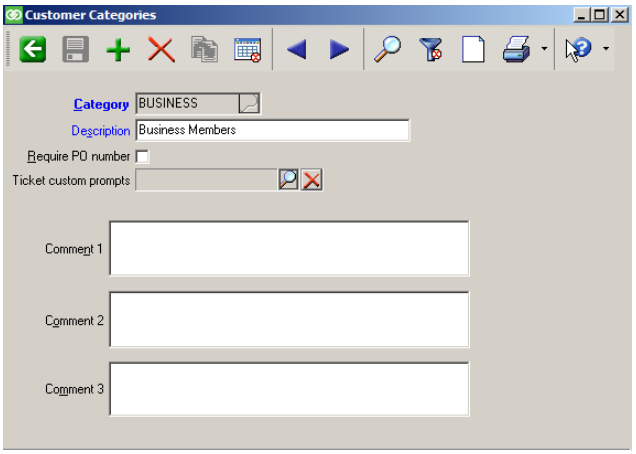
At "Standard form" on the Table tab, specify the form layout file, and click Edit to add all three fields to the layout.



Close the layout form and answer **Yes** to save your changes.

Click Save & Refresh when done.

3. Select **Setup / Customers / Categories** and look at the results. (You may have to resize the form to see the new fields.)



Example 2: Maintenance Form with Tabs

Add "Reference" and "Ref-Date" to **Customers / Customers**, to track who to speak with for a reference and the date they approved it.

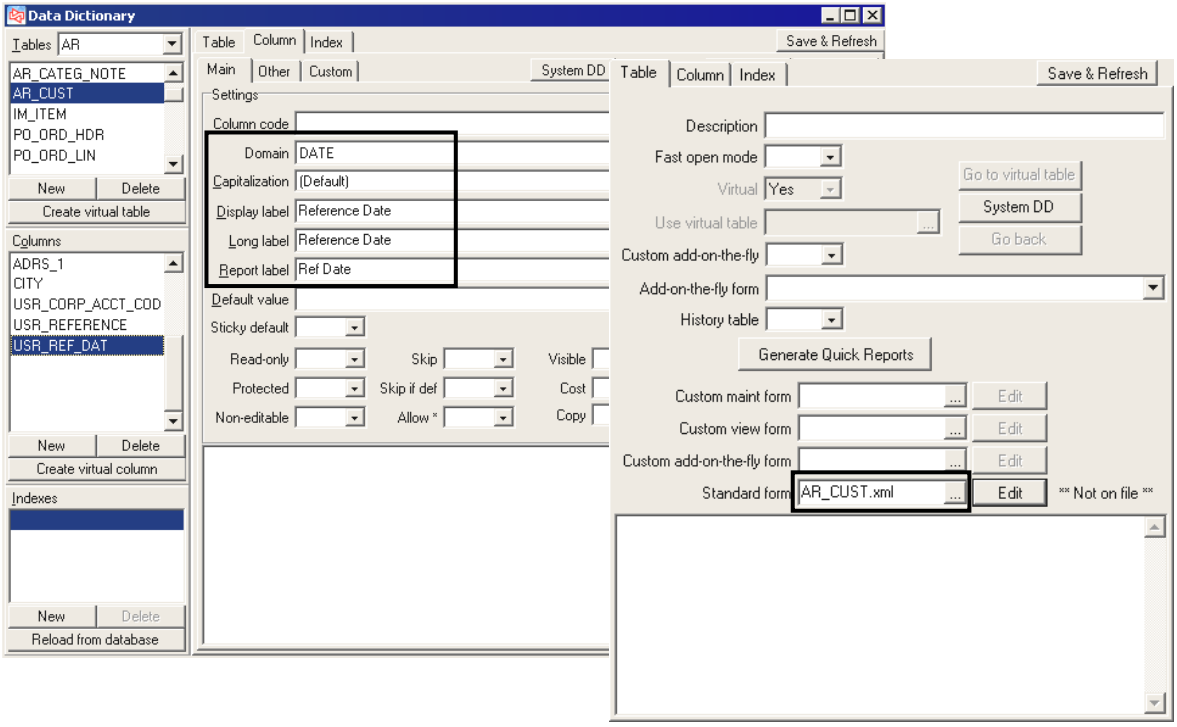
1. In Query Editor, add the two new fields to the AR_CUST table.

```
alter table  AR_CUST
     add  USR_REFERENCE          T_COMMNT,
          USR_REF_DAT            T_DAT
```

Allow the initial values to default to “Null”.

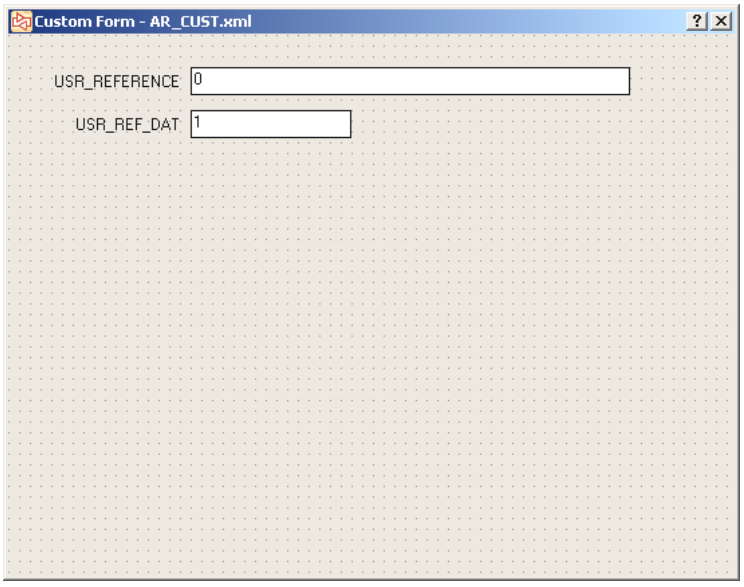
Load and execute **RefreshViews.sql** to update any views that use the AR_CUST table.

2. In **Setup / System / Configuration / Data Dictionary**, customize the AR_CUST table and add the two fields to the data dictionary.



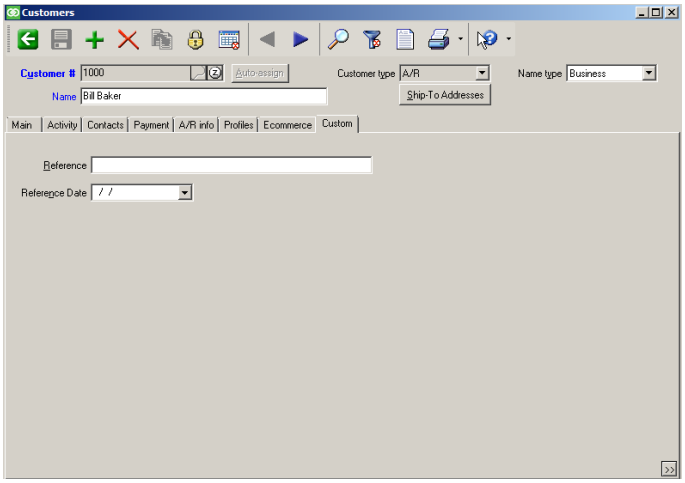
At "Standard form" on the Table tab, specify the form layout file, and click Edit to add the two new fields to the layout.

Close the layout form and answer **Yes** to save your changes.



Click Save & Refresh when done.

3. Select **Customers / Customers** or **Customers / Quick Customers** and display any customer record. Switch to the new Custom tab to see the new fields.



Example 3: Maintenance Form without Tabs

Add "Keyword_1", "Keyword_2", and "Keyword_3" to **Customers / Utilities / Category Notes** so that during lookups of category notes, notes can be found using a keyword.

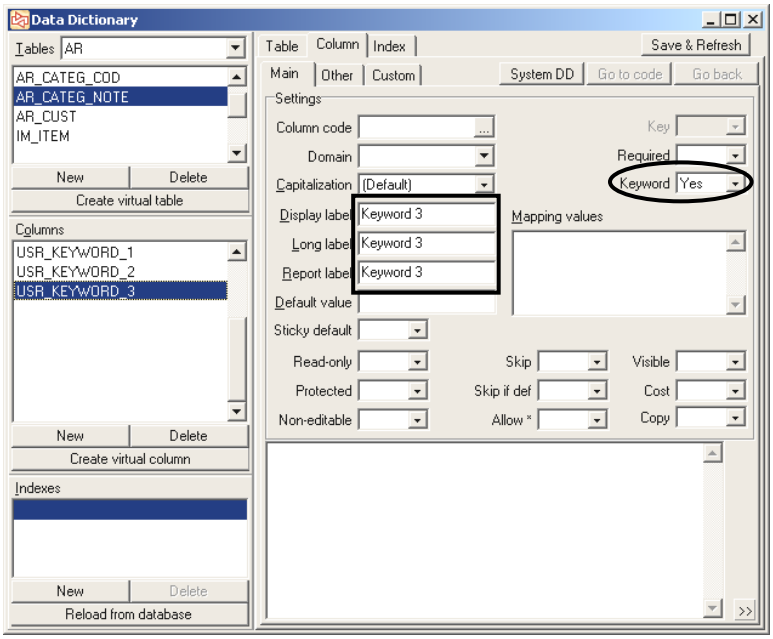
1. In Query Editor, add the three new fields to the AR_CATEG_NOTE table.

```
alter table  AR_CATEG_NOTE
     add  USR_KEYWORD_1          T_COMMNT,
          USR_KEYWORD_2          T_COMMNT,
          USR_KEYWORD_3          T_COMMNT
```

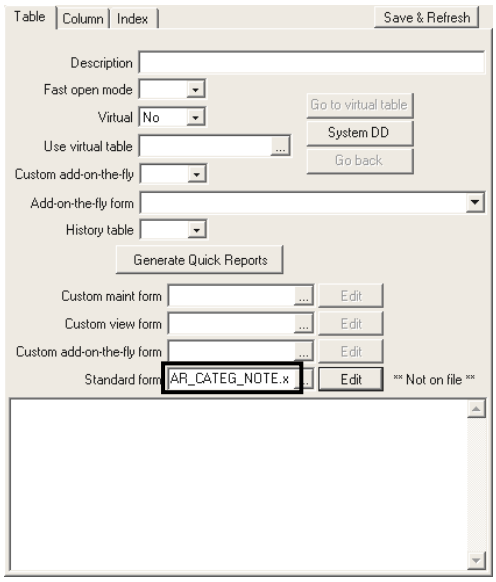
   Allow the initial values to default to "Null".

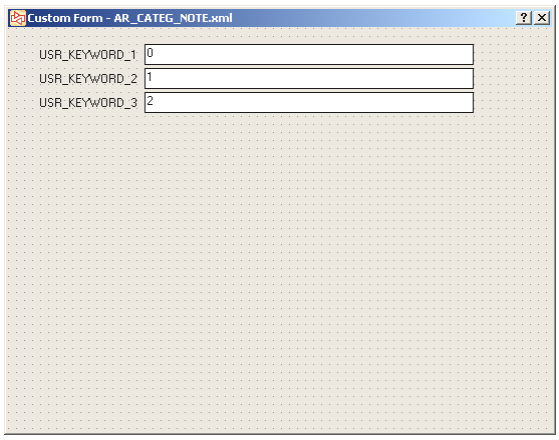
   Load and execute **RefreshViews.sql** to update any views that use the AR_CATEG_NOTE table.

2. In **Setup / System / Configuration / Data Dictionary**, customize the AR_CATEG_NOTE table and add the three fields to the data dictionary. Set each of the fields as a Keyword.



At "Standard form" on the Table tab, specify the form layout file, and click Edit to add the new fields to the layout.





Close the layout form and answer **Yes** to save your changes.

Click Save & Refresh when done.

3. Select **Customers / Utilities / Category Notes**. Add a note for the category **BUSINESS**, with a Note ID of RATE, with the text shown here.

   Enter the keywords **days rates corporate business 9holes 18holes**, separating the keywords with a space.

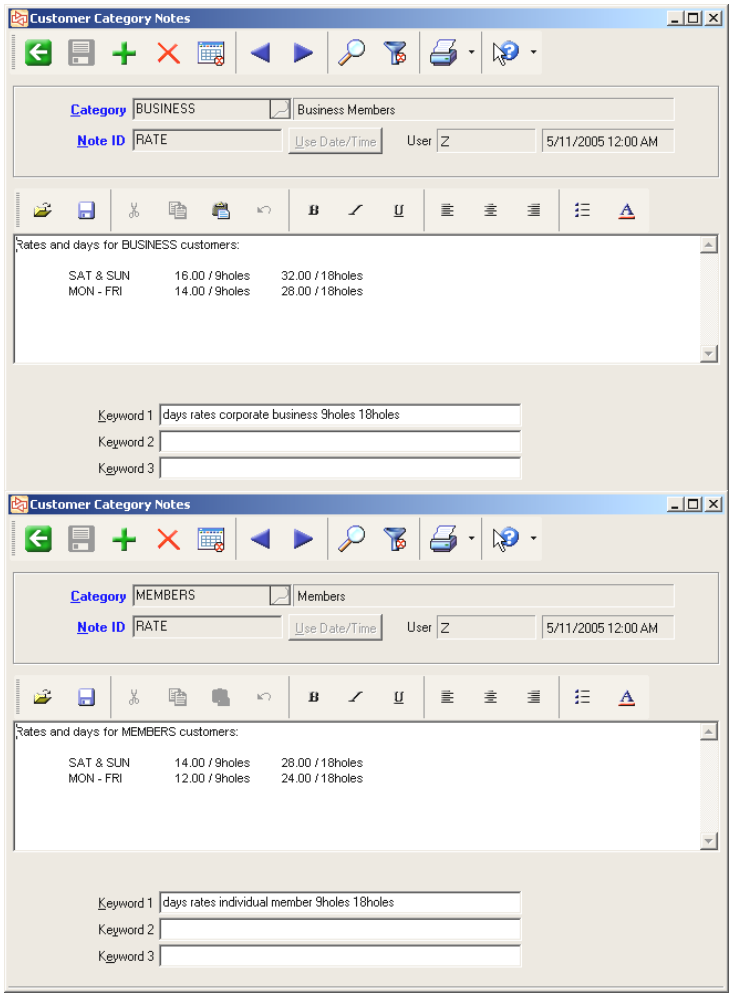
   Highlight the note text and click (or press Ctrl-C) to copy the text.

   Save the note.

4. Add a new note for the category **MEMBERS**, using a Note ID of RATE.

   Paste the text by clicking (or press Ctrl-V).

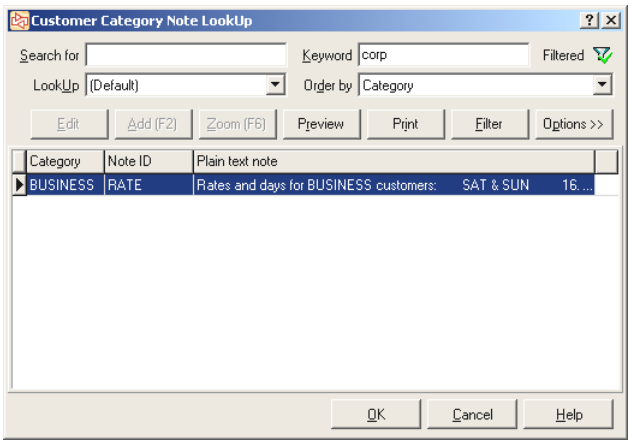
   Change the rates as shown here.



Enter the keywords **days rates individual member 9holes 18holes**, separating the keywords with a space.

Save the note.

5. Click  on the toolbar and enter **corp** at "Keyword". The lookup automatically filters the notes to the single note that has “corp” as a keyword.



Example 4: Document Header

Add “Shipping Instructions” and “Shipped by” to **Purchasing / Purchase Requests / Enter** to allow entry (and printing) of shipping information for the purchase order.

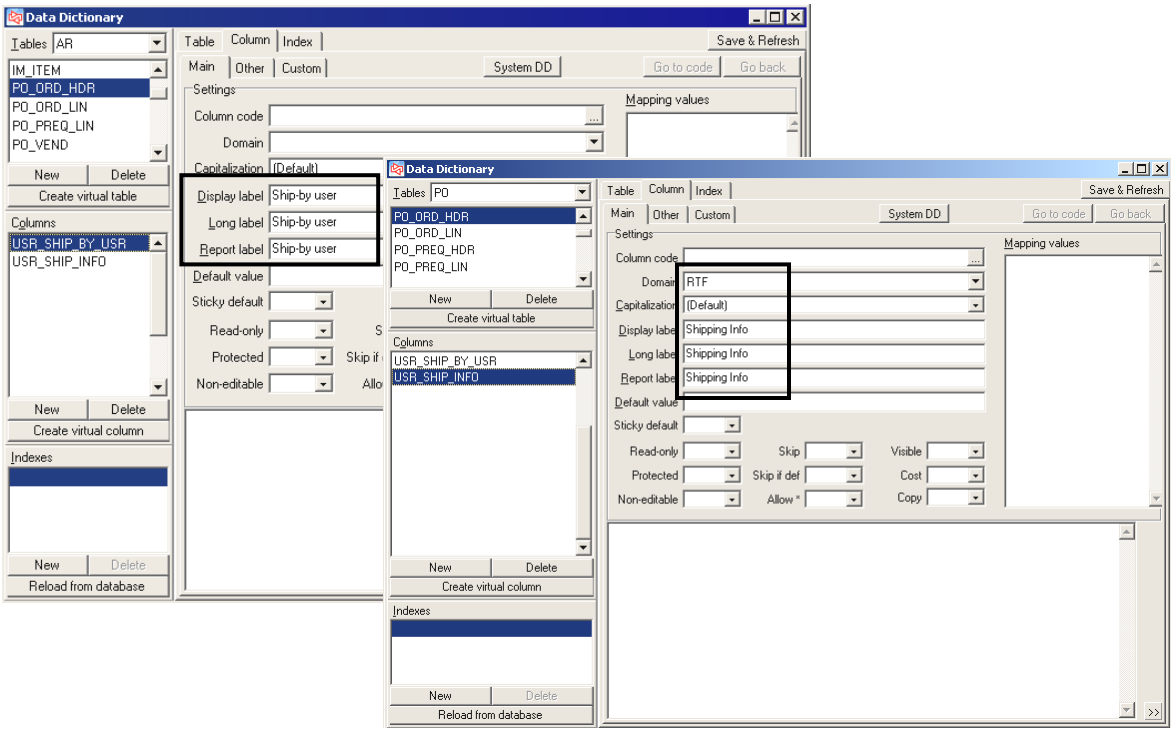
1. In Query Editor, add the two new fields to both the PO_PREQ_HDR and PO_ORD_HDR tables. By adding it to PO_ORD_HDR as well, the shipping information will be copied to it when the purchase request is posted so that the information is available for printing on purchase order forms.

   Note that the T_NOTE data type is used for the shipping instructions so that unlimited text can be entered in a scrolling notes window.

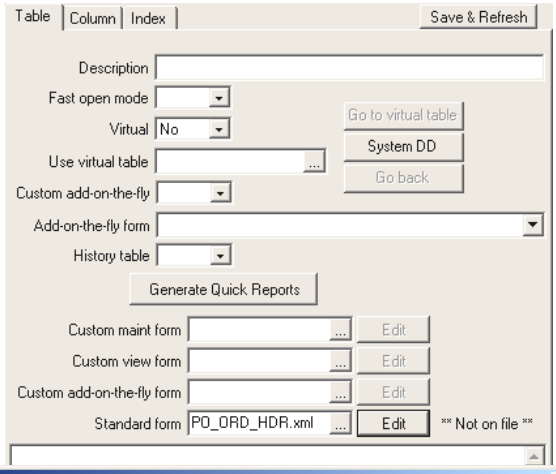
```
alter table  PO_PREQ_HDR
    add  USR_SHIP_INFO          T_NOTE,
         USR_SHIP_BY_USR        T_NAM

alter table  PO_ORD_HDR
    add  USR_SHIP_INFO          T_NOTE,
         USR_SHIP_BY_USR        T_NAM
```

   Load and execute **RefreshViews.sql** to update any views that use these two tables.

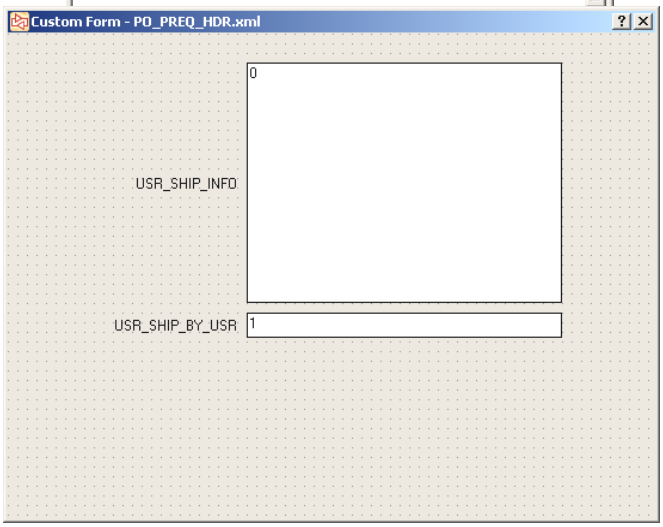
2. In **Setup / System / Configuration / Data Dictionary**, customize the PO_PREQ_HDR and PO_ORD_HDR tables and add the two fields to each table.



At "Standard form" on the Table tab for each table, specify the form layout file, and click Edit to add the new fields to the layout.



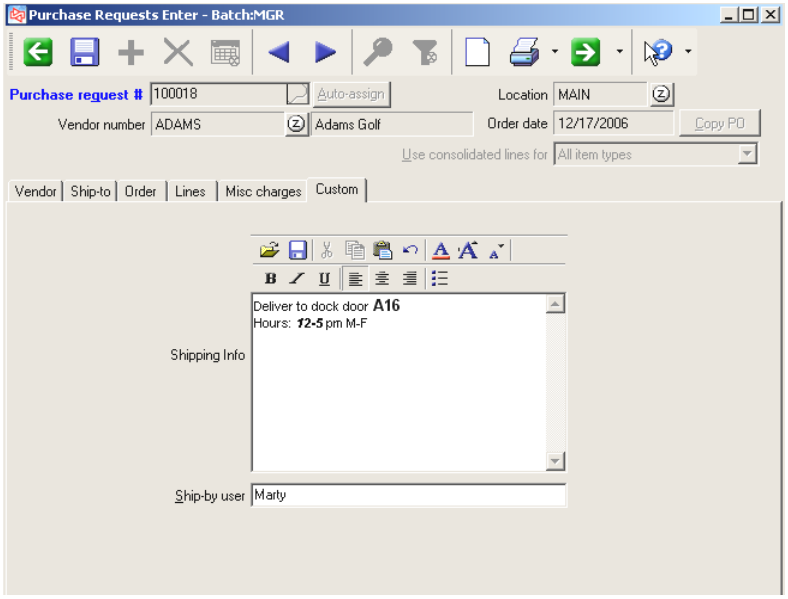
Close the layout form and answer **Yes** to save your changes.



Click Save & Refresh when done.

3. Select **Purchasing / Purchase Requests / Enter**.

Click  to display an unposted purchase request and switch to the Custom tab.



Example 5: Document Lines

Add "Special Information" and a "Review?" checkbox to **Purchasing / Purchase Requests / Enter** to allow entry (and printing) of special information for specific items on the purchase order. You could use the "Review?" checkbox in a custom Crystal report to show the purchasing information for only items that are marked for review.

1. In Query Editor, add the two new fields to both the PO_PREQ_LIN and PO_ORD_LIN tables.

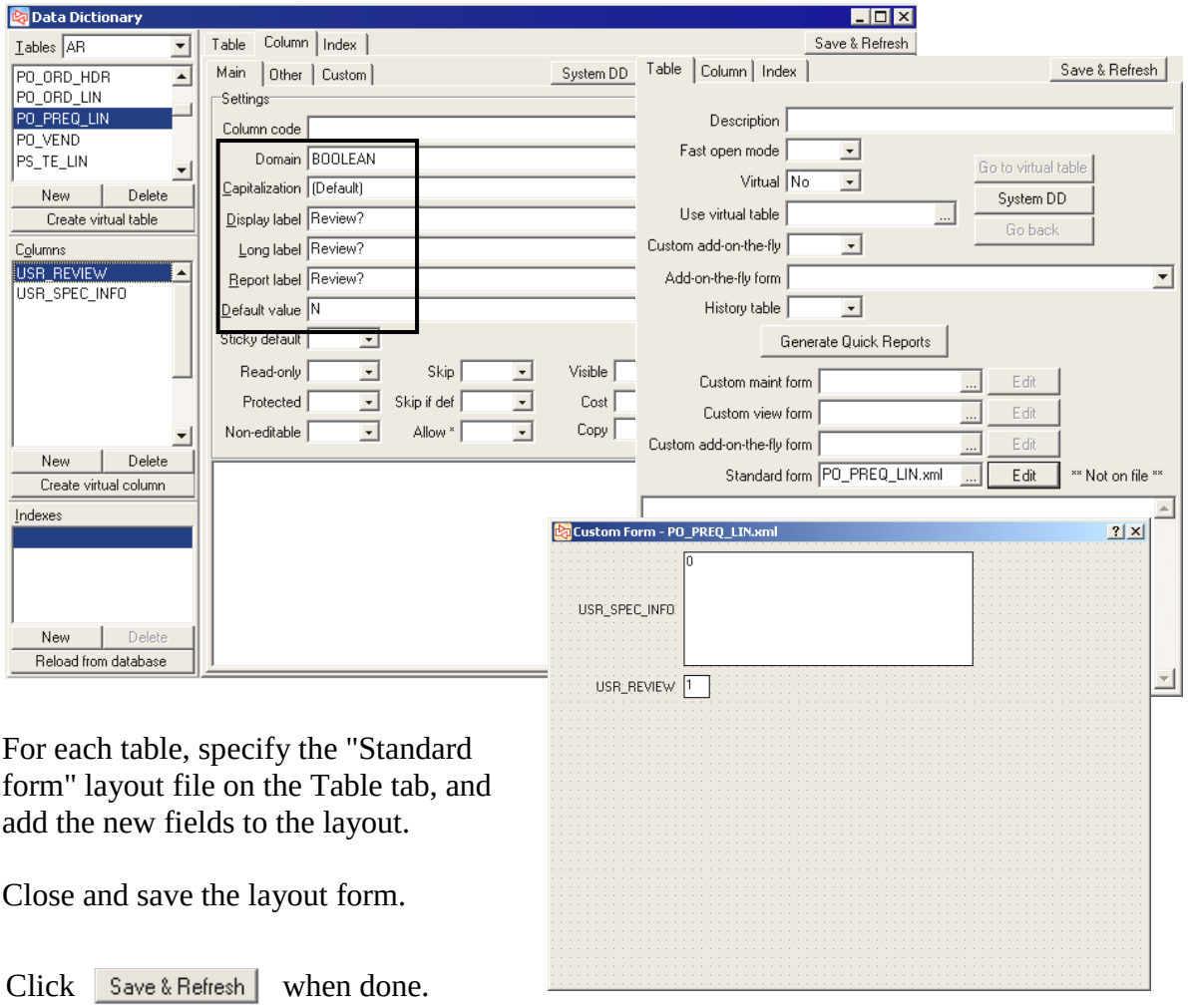
```
alter table  PO_PREQ_LIN
     add  USR_SPEC_INFO              T_NOTE,
          USR_REVIEW                 T_BOOL
           constraint USR_DF_PREQ_USR_REVIEW default 'N'


alter table  PO_ORD_LIN
     add  USR_SPEC_INFO              T_NOTE,
          USR_REVIEW                 T_BOOL
          constraint USR_DF_PO_USR_REVIEW default 'N'
```

   Adding the default constraints to the database ensures that the default value for USR_REVIEW will be set even if the PO line item is added outside of Counterpoint.

   Load and execute **RefreshViews.sql** to update any views that use these two tables.

2. In **Setup / System / Configuration / Data Dictionary**, customize the PO_PREQ_LIN and PO_ORD_LIN tables and add the two fields to each table.

   Set the "Domain" for the USR_REVIEW field to **BOOLEAN** so that it results in a checkbox, and set the "Default value" to **N**.



For each table, specify the "Standard form" layout file on the Table tab, and add the new fields to the layout.

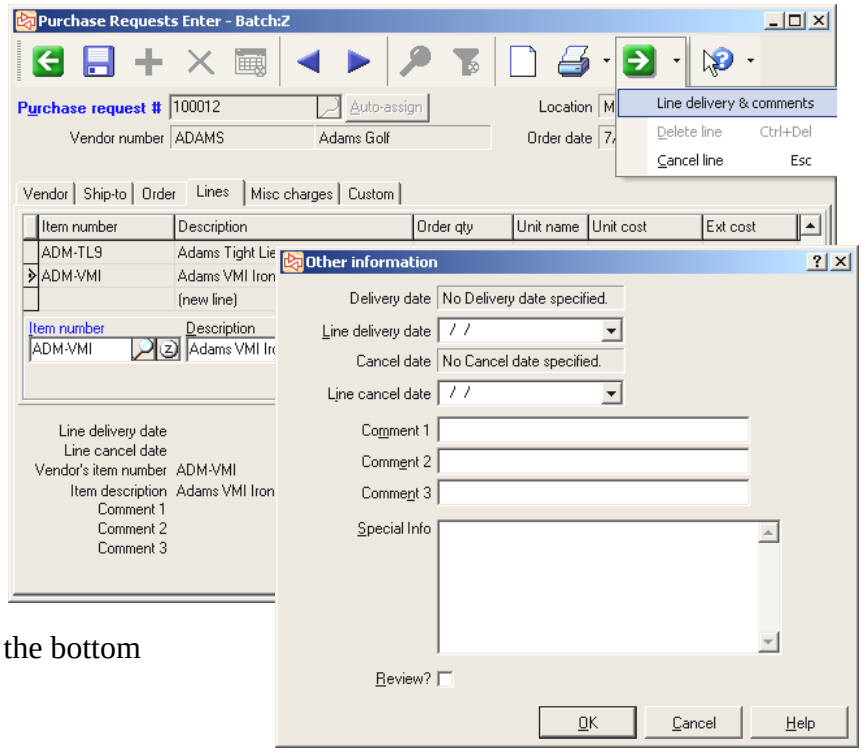
Close and save the layout form.

Click Save & Refresh when done.

3. Select **Purchasing / Purchase Requests / Enter**. Click ▶ to display an unposted purchase request.

Switch to the Lines tab and select to edit an existing line item.

Click ➔ for Additional Functions and select "Line delivery & comments".



The new fields display at the bottom of this window.

End of Exercise

"Live" Ticket Tables (add custom columns to **xxx_EXT** tables)

Data	Table name(s) (in database)	Data Dictionary table name*	Support for Custom fields?	Custom fields requested in Ticket Entry/ Touchscreen?
Header	PS_DOC_HDR PS_DOC_CONTACT PS_DOC_HDR_PROF PS_DOC_HDR_TOT PS_DOC_HDR_MISC_CHRG PS_DOC_HDR_PAY_IOA PS_DOC_HDR_HOLD_QUOT PS_DOC_HDR_ORIG_DOC PS_DOC_HDR_DOC_STAT PS_DOC_HDR_EC PS_DOC_HDR_LOY_PGM PS_DOC_HDR_**EXT** Primary Key: DOC_ID or DOC_ID_EXT	PS_TE_HDR (.XML only) VT_PS_DOC_HDR (template= PS_DOC_HDR_EXT)	Yes (in _EXT table only)	Yes (Ticket Profile dialog)
Line	PS_DOC_LIN PS_DOC_LIN_PRICE PS_DOC_LIN_PROMPT PS_DOC_LIN_PO PS_DOC_LIN_LOY PS_DOC_LIN_**EXT**	PS_TE_LIN (.XML only) VT_PS_DOC_LIN (template= PS_DOC_LIN_EXT)	Yes (in _EXT table only)	Yes (Item prompt dialog)
Cell	PS_DOC_LIN_CELL PS_DOC_LIN_CELL_**EXT**	PS_TE_LIN_CELL (.XML) VT_PS_DOC_LIN_CELL (template= PS_DOC_LIN_CELL_EXT)	Yes (in _EXT table only)	No
Serial	PS_DOC_LIN_SER PS_DOC_LIN_SER_PROMPT PS_DOC_LIN_SER_**EXT**	PS_TE_LIN_SER (.XML) VT_PS_DOC_LIN_SER (template= PS_DOC_LIN_SER_EXT)	Yes (in _EXT table only)	No
Payment	PS_DOC_PMT PS_DOC_PMT_CHK PS_DOC_PMT_CR_CARD PS_DOC_PMT_APPLY PS_DOC_PMT_PROMPT PS_DOC_PMT_**EXT**	PS_TE_PMT (.XML only) VT_PS_DOC_PMT (template= PS_DOC_PMT_EXT)	Yes (in _EXT table only)	Yes (Pay code Validations dialog)
Notes	PS_DOC_NOTE	PS_TE_NOTE.	No	No
Audit log	PS_DOC_AUDIT_LOG PS_DOC_AUDIT_LOG_TOT	PS_TE_AUDIT_LOG	No	No
Gift certificates	PS_DOC_GFC	PS_TE_GFC	No	No
Package Tracking #	PS_DOC_PKG_TRK_NO	PS_TE_PKG_TRK_NO	No	No
SVC	PS_DOC_SVC	PS_TE_SVC	No	No
Tax	PS_DOC_TAX	PS_TE_TAX	No	No

"History" Ticket Tables

Data	Table name(s) (in database)	Data Dictionary table name	Support for Custom fields?
Header (History)	PS_TKT_HIST PS_TKT_HIST_CONTACT PS_TKT_HIST_PROF PS_TKT_HIST_MISC_CHRG PS_TKT_HIST_PAY_IOA PS_TKT_HIST_HOLD_QUOT PS_TKT_HIST_ORIG_DOC PS_TKT_HIST_EC PS_TKT_HIST_LOY_PGM PS_TKT_HIST_EXT Primary Key: BUS_DT, DOC_ID	PS_TKT_HIST PS_TKT_HIST_CONTACT PS_TKT_HIST_PROF PS_TKT_HIST_MISC_CHRG PS_TKT_HIST_PAY_IOA PS_TKT_HIST_HOLD_QUOT PS_TKT_HIST_ORIG_DOC PS_TKT_HIST_EC PS_TKT_HIST_LOY_PGM PS_TKT_HIST_EXT	Yes (in _EXT table only)
Lines (History)	PS_TKT_HIST_LIN PS_TKT_HIST_LIN_PRICE PS_TKT_HIST_LIN_PROMPT PS_TKT_HIST_LIN_PO PS_TKT_HIST_LIN_LOY PS_TKT_HIST_LIN_EXT	PS_TKT_HIST_LIN PS_TKT_HIST_LIN_PRICE_COST PS_TKT_HIST_LIN_PROMPT PS_TKT_HIST_LIN_PO PS_TKT_HIST_LIN_LOY PS_TKT_HIST_LIN_EXT	Yes (in _EXT table only)
Cells (History)	PS_TKT_HIST_LIN_CELL PS_TKT_HIST_LIN_CELL_EXT	PS_TKT_HIST_LIN_CELL PS_TKT_HIST_LIN_CELL_EXT	Yes (in _EXT table only)
Serial numbers (History)	PS_TKT_HIST_LIN_SER PS_TKT_HIST_LIN_SER_PROMPT PS_TKT_HIST_LIN_SER_EXT	PS_TKT_HIST_LIN_SER PS_TKT_HIST_LIN_SER_PROMPT PS_TKT_HIST_LIN_SER_EXT	Yes (in _EXT table only)
Payments (History)	PS_TKT_HIST_PMT PS_TKT_HIST_PMT_CHK PS_TKT_HIST_PMT_CR_CARD PS_TKT_HIST_PMT_PROMPT PS_TKT_HIST_PMT_EXT	PS_TKT_HIST_PMT PS_TKT_HIST_PMT_CHK PS_TKT_HIST_PMT_CR_CARD PS_TKT_HIST_PMT_PROMPT PS_TKT_HIST_PMT_EXT	Yes (in _EXT table only)
Notes (History)	PS_TKT_HIST_NOTE	PS_TE_NOTE	No
Audit log (History)	PS_TKT_HIST_AUDIT_LOG PS_TKT_HIST_AUDIT_LOG_TOT	PS_TKT_HIST_AUDIT_LOG PS_TKT_HIST_AUDIT_LOG_TOT	No
Gift certificates (History)	PS_TKT_HIST_GFC	PS_TKT_HIST_GFC	No
Stored value cards (History)	PS_TKT_HIST_SVC	PS_TE_PKG_TRK_NO	No
Package tracking #s (History)	PS_TKT_HIST_PKG_TRK_NO	PS_TKT_HIST_PKG_TRK_NO	No
Tax (History)	PS_TKT_HIST_TAX	PS_TKT_HIST_TAX	No

Similar table changes for PS_LWY_HIST
PS_ORD_HIST
PS_VOID_HIST

Required steps – Example 2

1. Add physical field to tables in the database.

```
alter table  PS_DOC_LIN_EXT
     add USR_GRD_LVL                    T_FLG    null
go

alter table  PS_TKT_HIST_LIN_EXT
     add USR_GRD_LVL                    T_FLG    null
go

alter table  PS_VOID_HIST_LIN_EXT
     add USR_GRD_LVL                    T_FLG    null
go
```

For PS_DOC tables, add your fields to the corresponding _EXT table.

If you add the same field to the corresponding history tables, Counterpoint will automatically copy the value to history when the transaction is posted.

While not required, including “go” after each statement allows the SQL server to refer to a specific statement number if a failure occurs in one of the commands.

2. Set the fields to appropriate initial values in existing records.

Use method a) if new column is “not null” and method b) if new column is “null”.

a) When adding the column, include a constraint that sets the default value:

```
alter table  PS_DOC_LIN_EXT
          add USR_GRD_LVL     T_FLG   not null
          constraint USR_DF_USR_GRD_LVL default 'K'
```

b) After adding the column, update the table and set the value of the column:

```
update PS_DOC_LIN_EXT
     set USR_GRD_LVL = 'K'
go

update PS_TKT_HIST_LIN_EXT
     set USR_GRD_LVL = 'K'
go

update PS_VOID_HIST_LIN_EXT
     set USR_GRD_LVL = 'K'
go
```

Be sure to set initial values in existing records for Boolean fields, where data type is T_BOOL ("Y" or "N"), and for fields with mapping values.

An initial value of "Null" is assumed if not specifically set.

3. Update view definitions for associated tables.

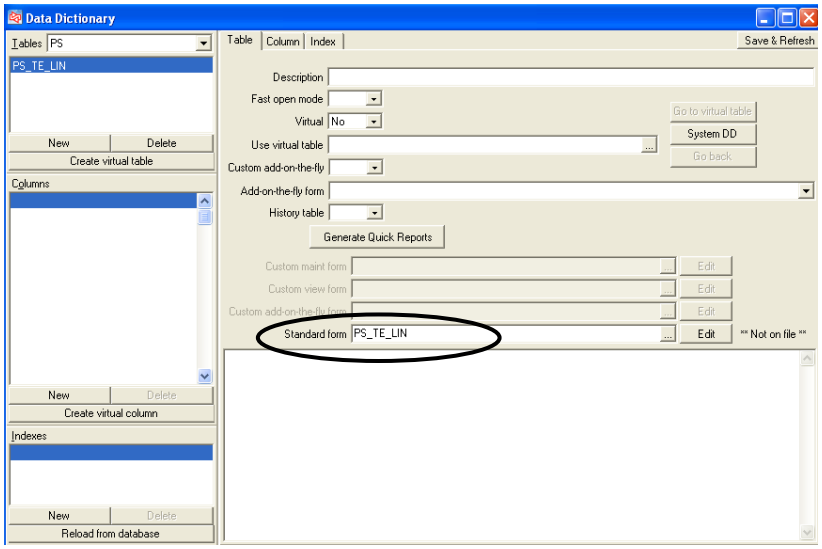
Load and execute **RefreshViews.sql** to update any views that use the tables that you've modified.

4. Add new field to the data dictionary.

In **Setup / System / Configuration / Data Dictionary**, customize the table. For a PS_DOC_xxx _EXT table, select to customize the corresponding PS_TE_xxx table.

On the Table tab:

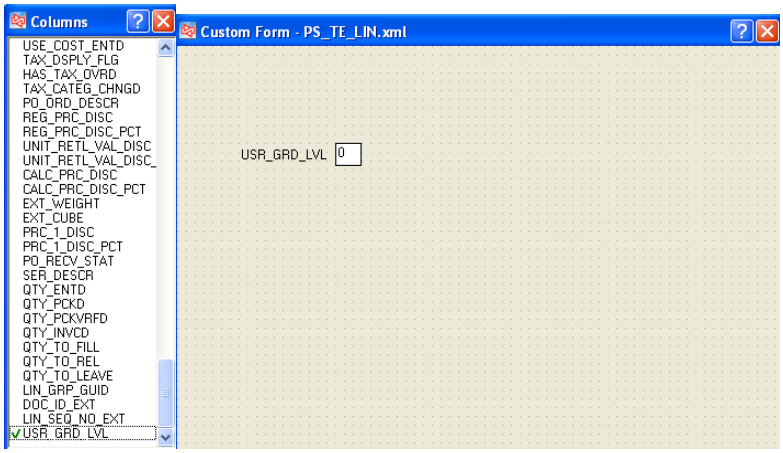
If you want the new field to appear on the standard form for the table, enter the name of a form layout file at **Standard form**.



The form layout file will automatically be created if it does not already exist.

Click Edit to open the form layout editor.

Select the new field from the list of all fields in the table and drag it to the desired position on layout form.



Close the layout form when done to save your changes.

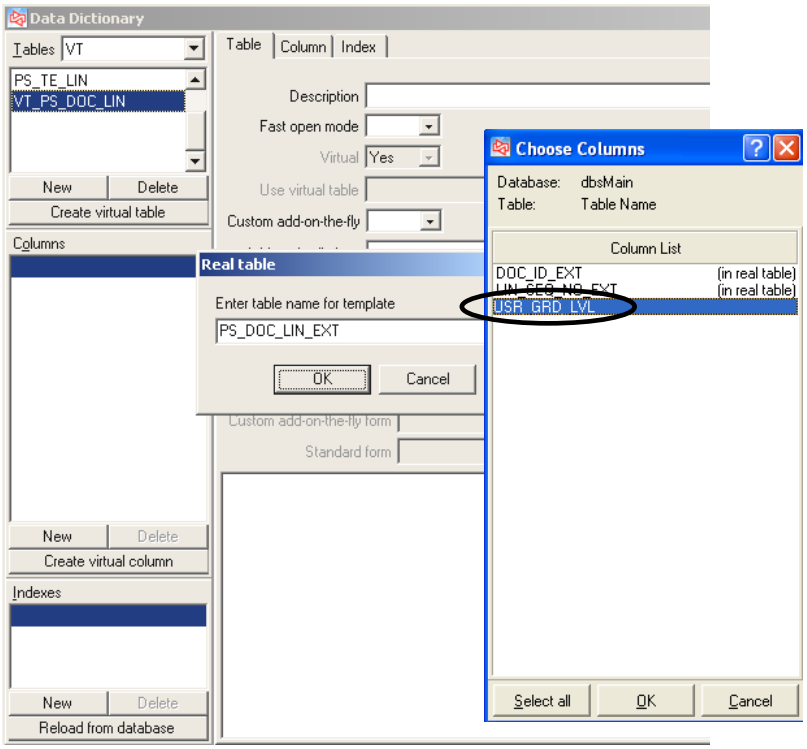
Next, set the properties of the column.

For **non PS_DOC_xxx** tables, set column properties for the same table where you defined the form layout. For **PS_DOC_xxx** tables, first select to customize the corresponding VT_PS_DOC_xxx table and set the column properties for that table.

On the Data Dictionary form, click New under **Columns**.

If asked, specify a physical table name as a template. For a VT_PS_DOC_xxx table, use the corresponding PS_DOC_xxx table.

Then select your new column from the list.

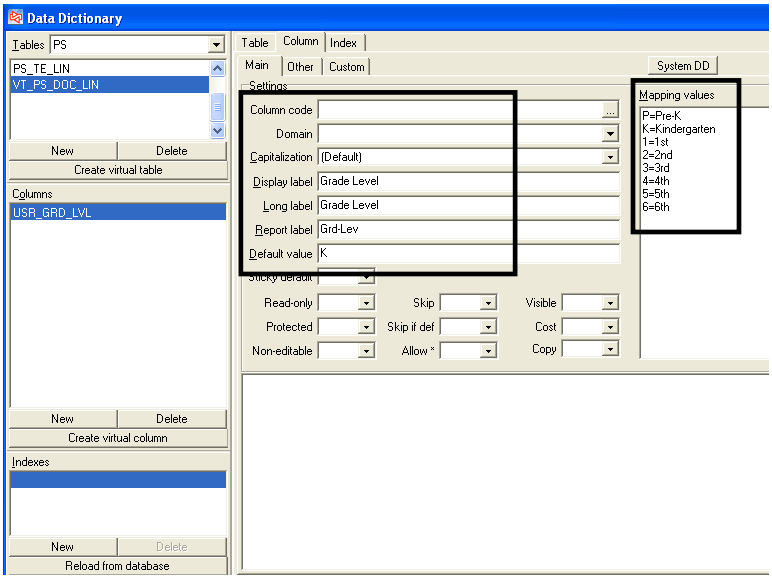


<u>On the Column Main tab:</u>

Set the **Display label**, **Long label**, and **Report label** for the new column.

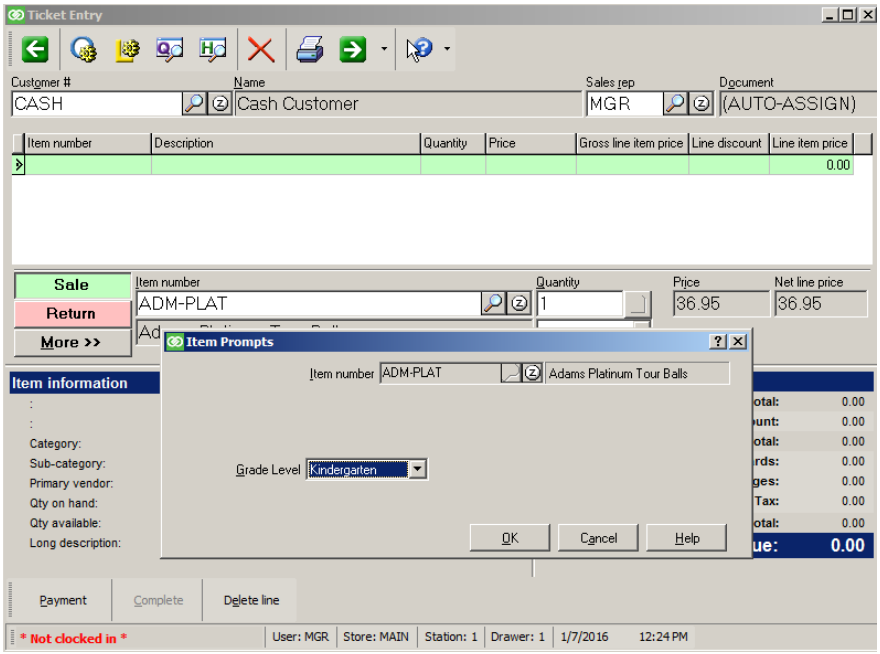
If necessary, assign a **Domain** and **Default value**.

Enter **Mapping values** to provide the user a list of choices to pick from.



Click Save & Refresh to save your entries to the Data Dictionary.

The new column should now be available in the function that displays data from the involved table. You may need to exit and restart Counterpoint for the changes to be available when entering tickets.



Note that it is NOT necessary to add Data Dictionary entries for the new column to the associated history tables (in this case, to PS_TKT_HIST_LIN_EXT and PS_VOID_HIST_LIN_EXT). As long as the columns physically exist in the tables, the data will automatically be updated to the history tables when tickets are posted.

- Use to set default value for a column
- Use to restrict column value to a specific list of values
- Constraints in database are also enforced outside of Counterpoint

Setting a Default Constraint

Add with new column (sets defaults in all existing records)

```
alter table PS_DOC_LIN_EXT
      add USR_GRD_LVL        T_FLG   not null
      constraint USR_DF_USR_GRD_LVL default 'K'
```

Add for existing column (does not set defaults in existing records)

```
alter table IM_ITEM
      add constraint USR_DF_WARR_UNIT_1 default 'D' for WARR_UNIT_1
```

Setting a Check Constraint

Add with new column

```
create table dbo.IM_ITEM
      (ITEM_TYP                  T_FLG              not null
       constraint CK_IM_ITEM_ITEM_TYP check (ITEM_TYP in ('I','N','S','D'))
```

Add for existing column (rolls back all if existing records have invalid values)

```
alter table IM_ITEM
      add constraint CK_IM_ITEM_WARR_UNIT_1 check (WARR_UNIT_1 in
      ('D','M','Y'))
```

Also set mapping values in Data Dictionary

Adding Variable Custom Fields

What is a “Variable Custom field”?

Custom column that can appear for specific items, customers, or pay codes (e.g., prompt whether Mustard is desired on a Hamburger, but not for Pizza)

Where can “Variable Custom fields” be used?

Only in Ticket Entry or Touchscreen, to prompt for information for individual items, customers, or pay codes

Checklist for adding a Variable Custom field

Required steps

1. Add physical field to appropriate PS _DOC_XXX_EXT table in the database.
2. Update view definitions for associated tables.
3. Add new field to the data dictionary.
4. Set up custom layout form
5. Assign custom layout form to items, customers, or pay codes

The first three steps are almost identical to the steps used when defining a regular custom field.

1. **Add physical field to appropriate PS _DOC_XXX_EXT table.**

For variable fields for	Add fields to this table
Customers & Customer categories	PS_DOC_HDR_EXT
Items, Item Categories, & Item sub-categories	PS_DOC_LIN_EXT
Pay codes	PS_DOC_PMT_EXT

Use Query Editor or SQL Script Utility to add the new fields to the table.

```
alter table PS_DOC_LIN_EXT
add
  -- Things you may want on a hamburger
  MUSTARD  T_BOOL   null,
  KETCHUP  T_BOOL   null,
  PICKLES  T_BOOL   null,
  MAYO     T_BOOL   null,
  -- Things you may want on a pizza
  PEPPERONI T_BOOL  null,
  SAUSAGE   T_BOOL  null,
  MUSHROOM  T_BOOL null,
  ANCHOVY   T_BOOL null,
  GREENPEPPER T_BOOL null,
  -- Things you may want on BOTH
  TOMATO   T_BOOL   null,
  ONION    T_BOOL   null
```

2. **Update view definitions for associated tables.**

Run RefreshViews.sql.

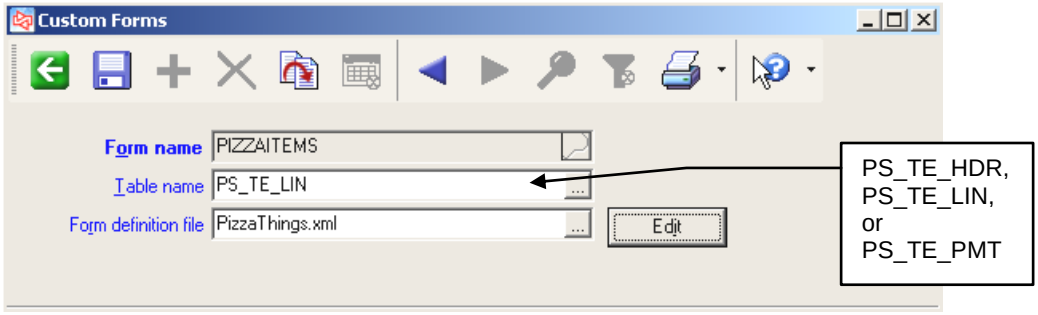
3. **Add new fields to the data dictionary.**

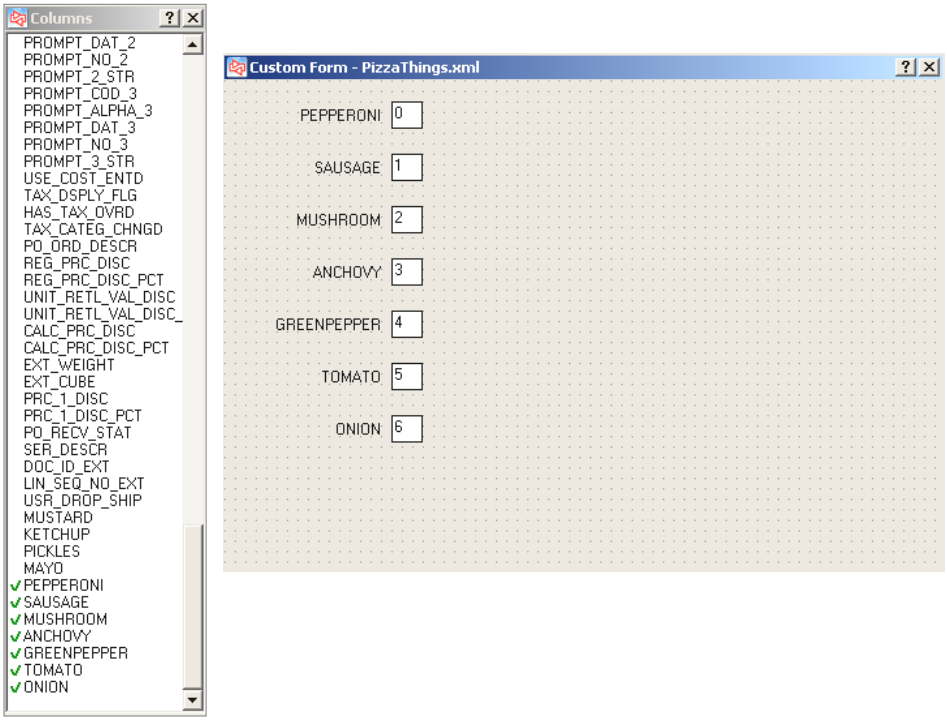
Use **Setup / System / Configuration / Data Dictionary** to provide display labels for the variable custom fields by customizing the corresponding VT_PS_DOC_xxx table. (Use PS_DOC_xxx_EXT as the template table.)

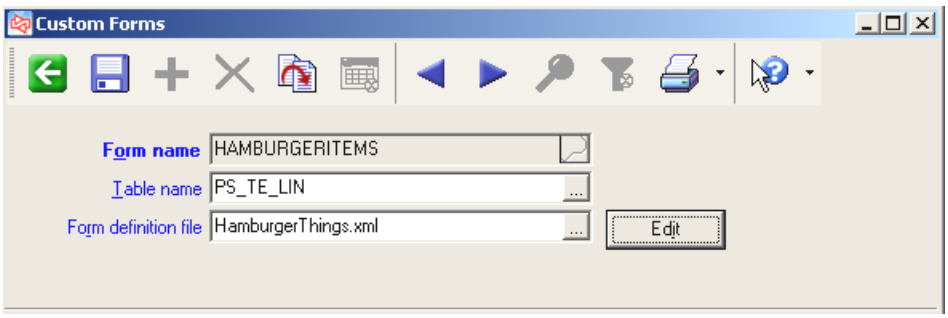
Only supply a display label for each variable custom field. Do not mark the fields as required and do not give them a default value.

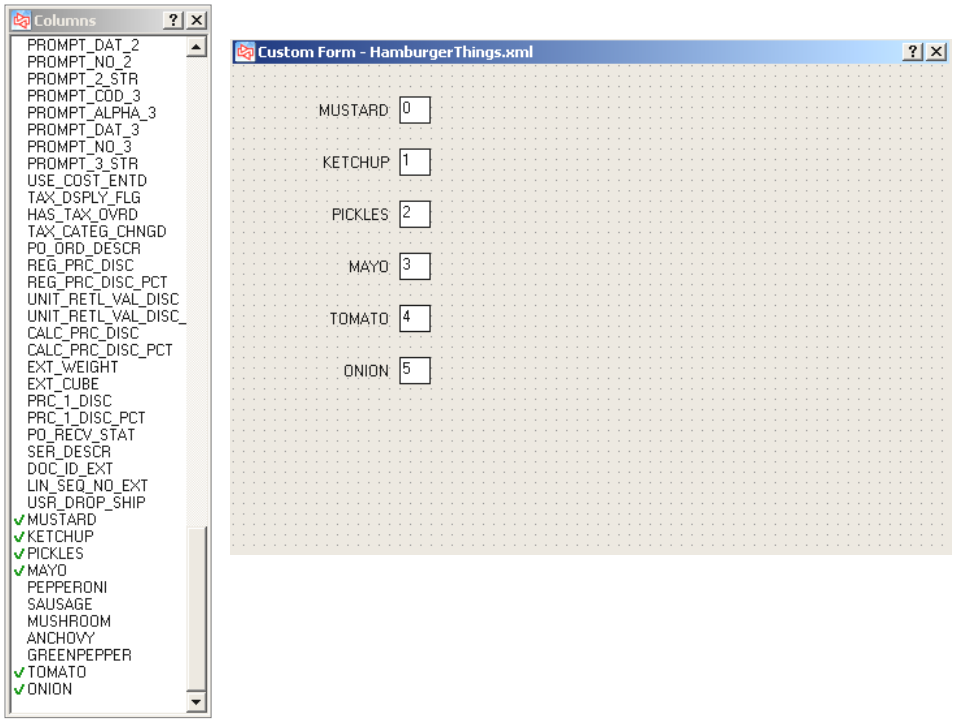
4. Set up a custom layout form for each group of variable custom fields.

Rather than doing this in the Data Dictionary, use **Setup / System / Custom Forms** to build the layout form.

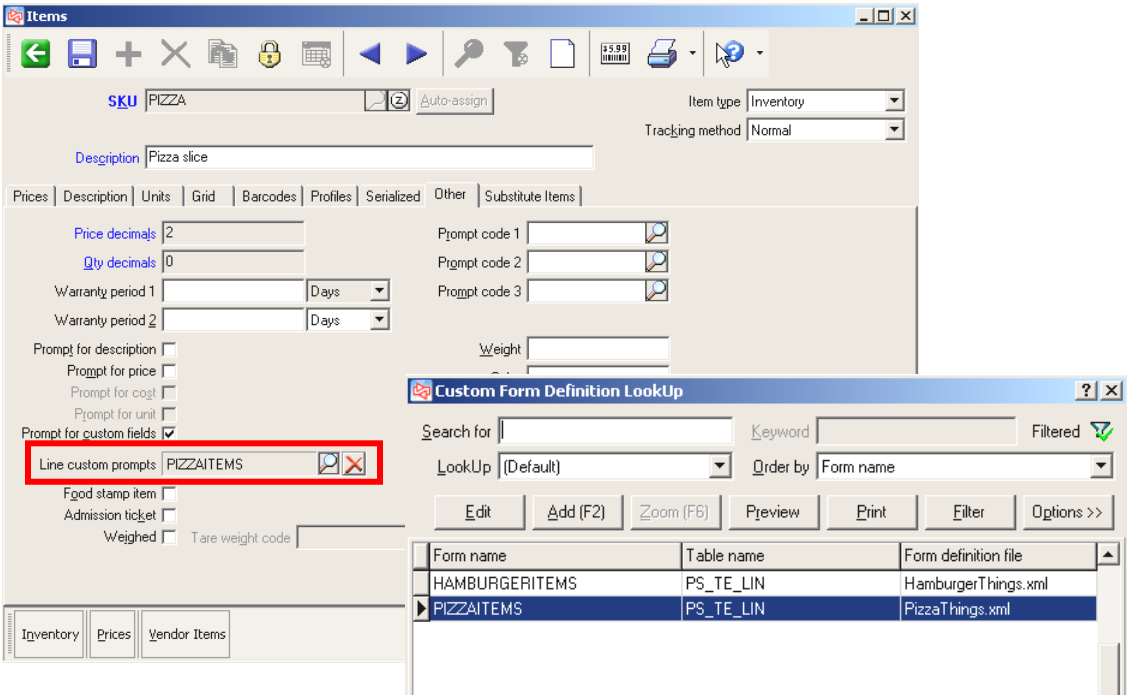








5. Assign custom layout form to items, customers, or pay codes



Custom layout forms can be assigned in several functions. Which takes precedence?

---- Displays fields in first-found custom form assigned to ----		
Line Items	**Customers**	**Pay Codes**
Item	**Customer**	**Pay code**
Sub-category	**Category**	**Station**
Category	**Station**	**Store**
Data Dictionary (for PS_TE_LIN)	**Store**	**Data Dictionary (for PS_TE_PMT)**
	Data Dictionary (for PS_TE_HDR)	

Notes on Variable Custom Fields

- Variable custom fields appear only when entering tickets.
- "Prompt for custom fields" setting for an item, customer, or pay code still controls whether variable custom fields are automatically requested
- Use the  button to clear a custom layout form name

  In table view, simply blank out the custom layout form name.

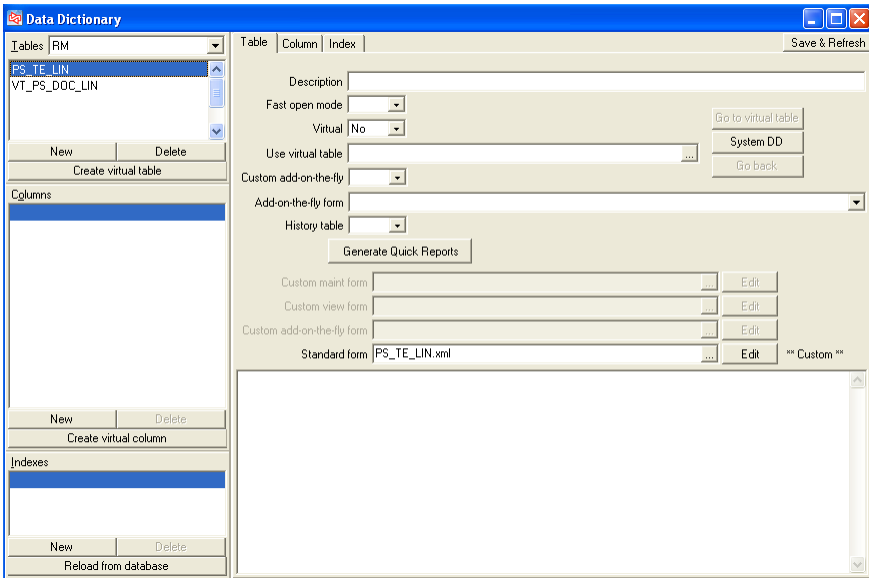
- All XML layout files are stored in company's CustomForms folder, including those defined in Data Dictionary
- Variable custom fields are automatically deployed from the site server to Offline Ticket Entry workstations
- Variable custom fields are automatically included in Multi-Site replication. XML layout files are deployed with other Top-Level files, using FileSync.

Try it Yourself!

In this exercise, you will change the location of the custom layout form that you previously defined to request Grade Level so that it only appears for a single item instead of for all items.

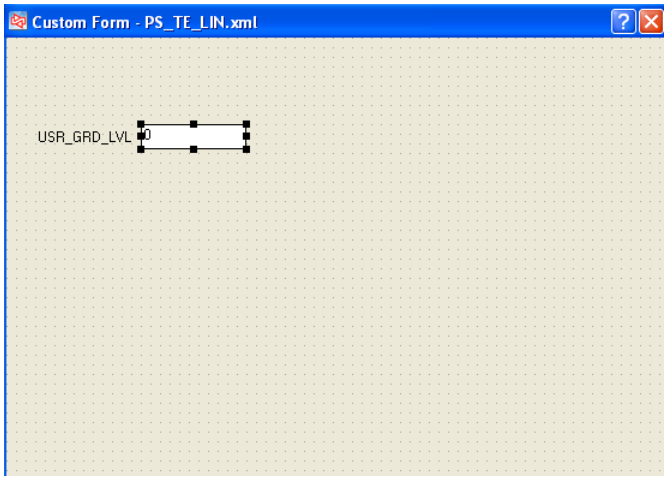
Since you previously added the field to the PS_DOC_LIN_EXT table and provided a display label for it, you'll only need to work on the layout form.

1. Open the Data Dictionary Editor and display the definition you previously built for the PS_TE_LIN table.



Click Edit to open the PS_TE_LIN.xml layout.

In the layout window, delete the USR_GRD_LVL field.

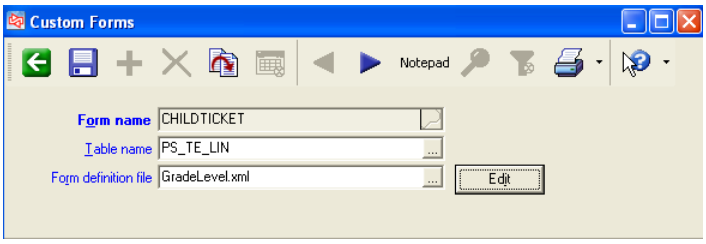


Close the window and save the changes.

2. Select **Setup / System / Custom Forms** from the Counterpoint menu.

Add a new form named **CHILDTICKET**.

Enter **PS_TE_LIN** as the name of the table, and **GradeLevel** as the name of the Form definition file.

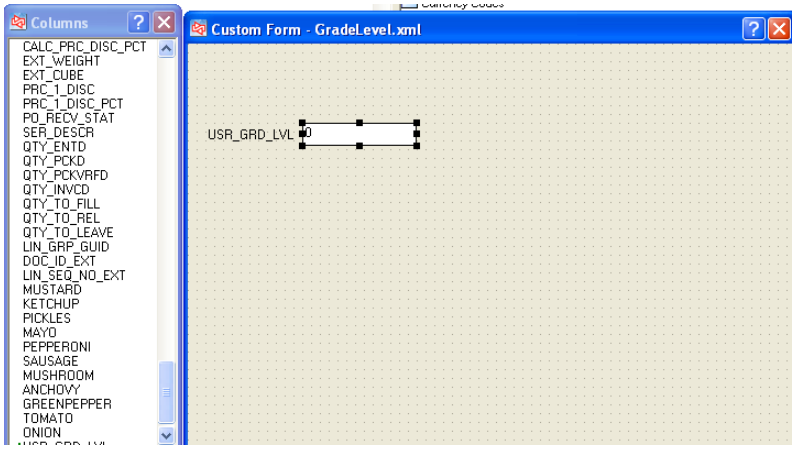


Click  to open the GradeLevel.xml layout.

Position USR_GRD_LVL on the layout window.

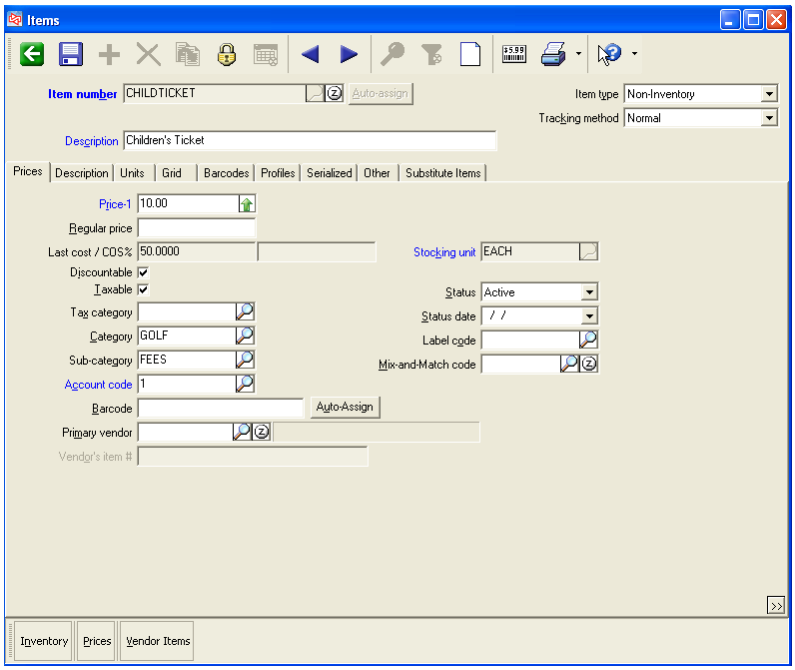
Close the window and save the changes.

Then save the Custom Form.



3. Select **Inventory / Items** from the Counterpoint menu.

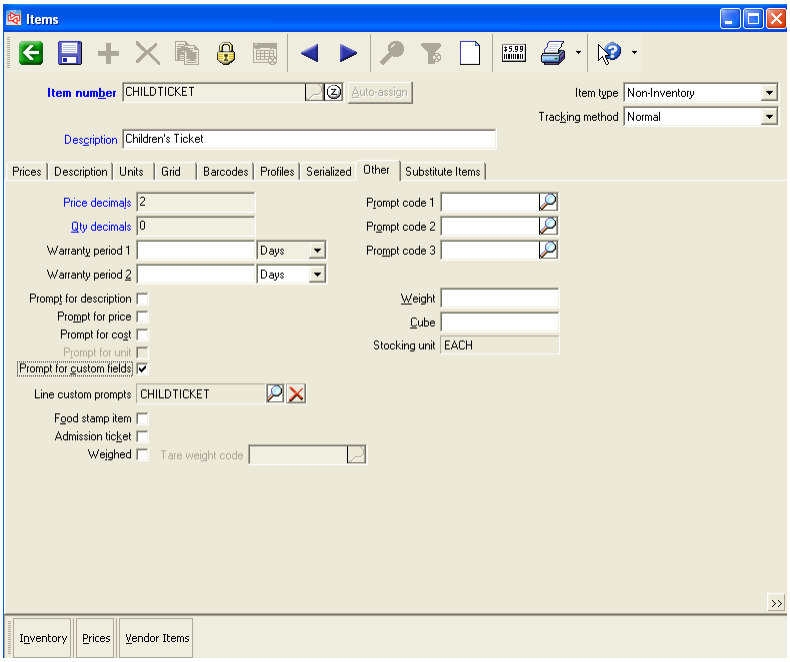
Set up a new item named CHILDTICKET as a non-inventory item, so that it looks similar to this:



On the Other tab, enable **Prompt for custom fields**.

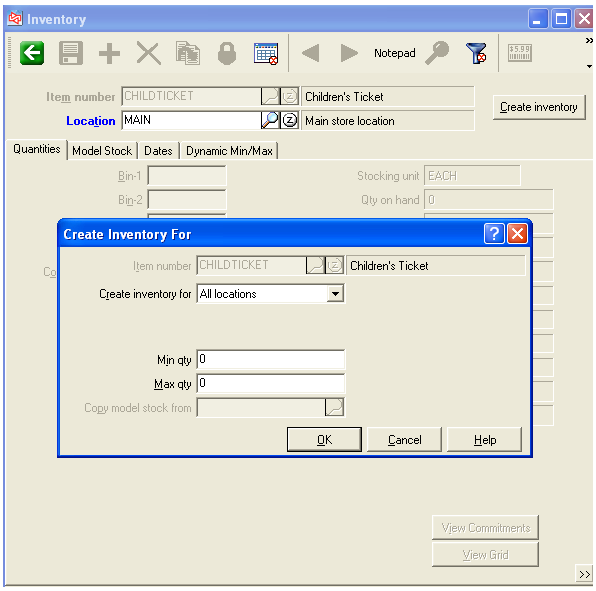
At **Line custom prompts**, look up and select the **CHILDTICKET** custom form.

Save the item record.

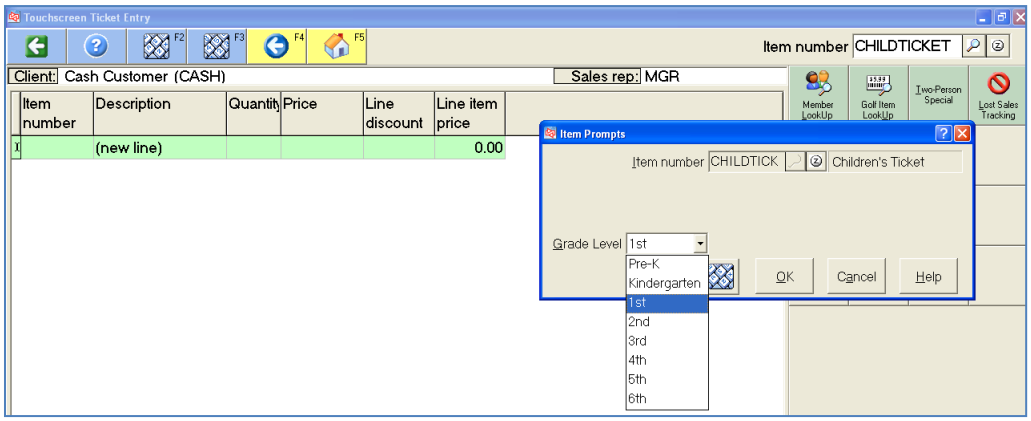


When the Inventory window opens, click Create inventory and select to create inventory for All locations.

After the inventory records are created close the Inventory window and the Items window.



Now, when you sell the CHILDTICKET item, the item prompts window will automatically open and the mapping values for Grade Level will appear in the dropdown list.

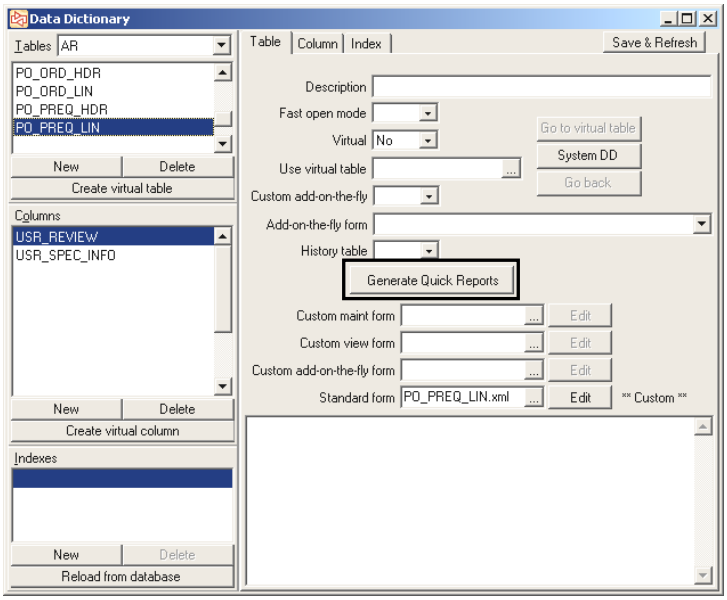


Optional steps

5. Regenerate the Quick Report for the table, if you want to include the new field.

On a system with Crystal Decisions Report Creation API License:

Switch to the Table tab, and click Generate Quick Reports.



On a system without the Creation API License, open the standard Quick Report in Crystal Reports and:

- set the datasource location so that the new field becomes available in Crystal,
- select File / Options and ensure that "Insert Detail Field Headings" is enabled on the Layout tab,
- drag the new field to the Details band of the quick report*, and
- save the report in CounterPoint / <company> / QuickReports / <tablename>.rpt.

* For a field with mapping values, this will show just the storage value. To show the associated display value, create a formula named REPLCP_<Table>.<Field>_MV that contains the value {<Table>.<Field>}. Position the formula on the report in the Details band, instead of the field itself.

6. Use Crystal Reports to customize existing reports to add the new field.

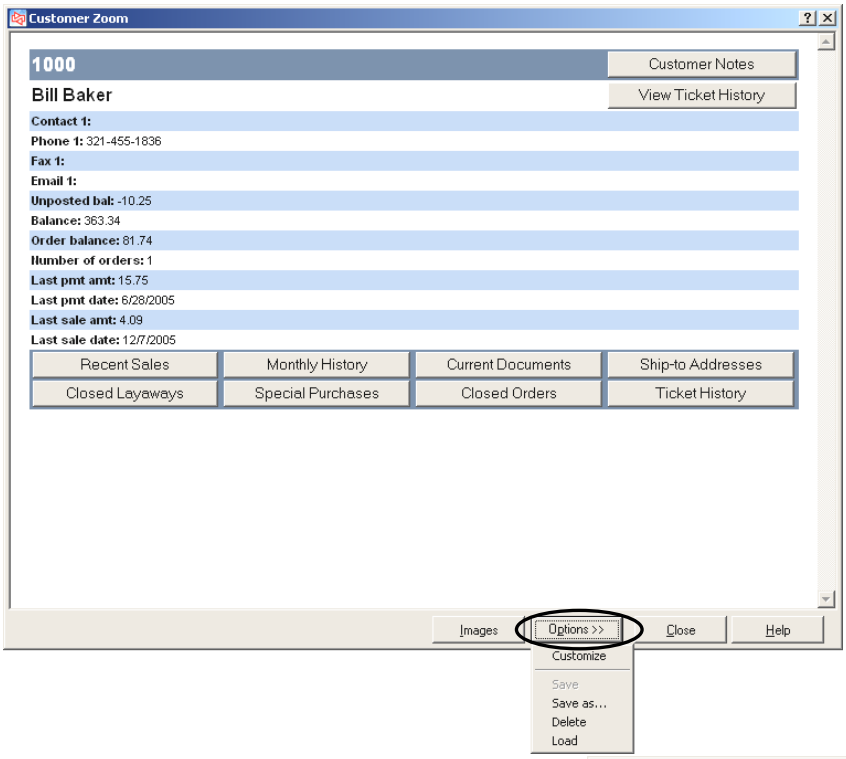
Open the existing report in Crystal Reports.

First, use **Database / Set Datasource Location** to “remap” the tables to your database. (New columns won’t appear until you do this.) Then, in the Field Explorer, select your new field and position it in the report.

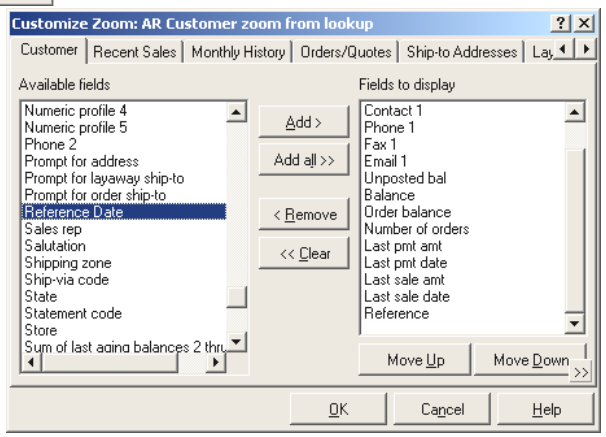
7. Customize the table's zoom window to add the new field.

(You may need to exit Counterpoint and restart it in order for the new fields to appear in the "Available fields" list when customizing.)

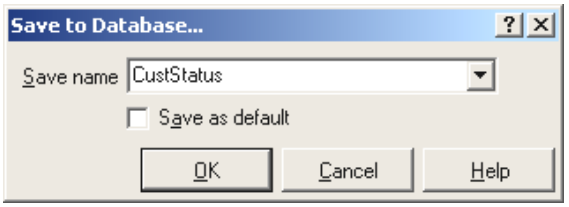
Simple customization of Zoom window



Select the new field from the corresponding table's tab and click OK when done.



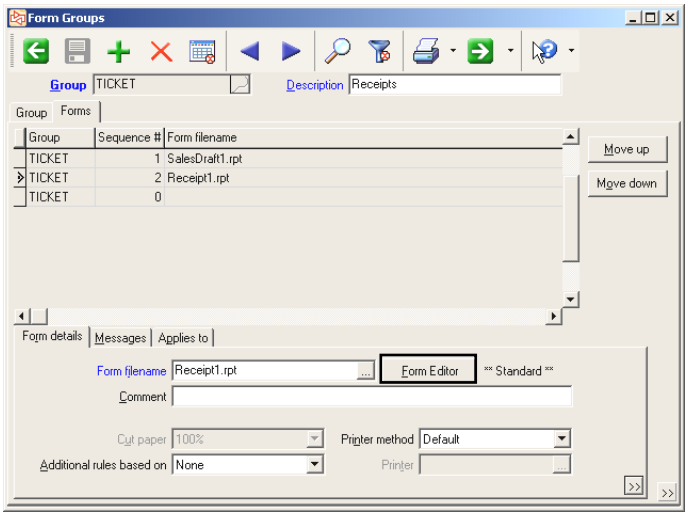
Under Options, select **Save as…** to save the zoom.



8. Customize Point of Sale receipts to add the new field.

Use **Setup / Point of Sale / Form Groups** and display each form group that uses a receipt form to which the new field must be added.

For each group, switch to the Forms tab, select the form to be changed and click Form Editor to start the Form Editor (for OPOS or RDLC forms) or Crystal Reports (for Crystal forms).



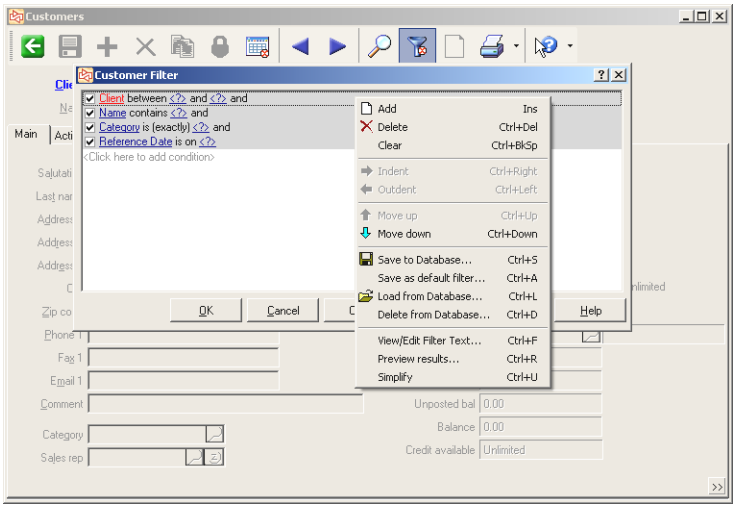
Position the new field on the form. (For Crystal forms, you will need to reset the datasource location to make your new fields available.)

Also, for Crystal forms, remember to make your changes to the History version of the form (e.g., Receipt1History.rpt). Do not copy and rename the non-history version of the form.

9. Customize filters to add the new field.

To add the new field to the default filter for a function, click Options >> to customize the filter, and add the conditions for the new fields. Then right-click and select **Save as default filter**.

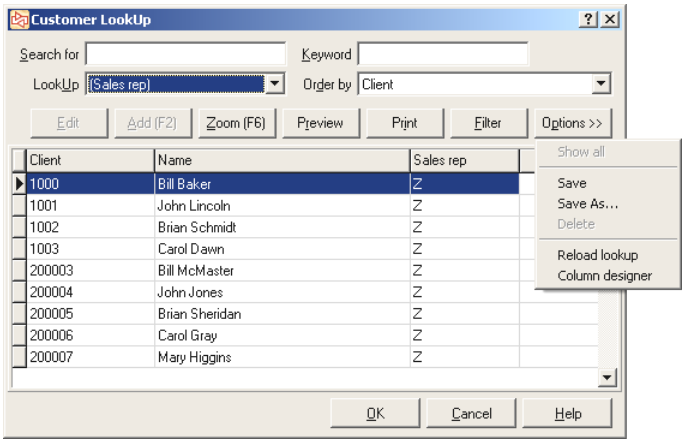
To add the new field to other non-default filters, first select **Load from Database**, customize the filter, and then select **Save to Database**.



10. Customize lookups for the table to add the new field.

To add the new field to the default lookup for a table, open the Lookup for the table, click Options >> and use the Column Designer to add the field to the list of Visible columns. Select **Save** when completed.

To add the new field to other non-default lookups, first select the desired lookup at "LookUp", use the Column Designer to add the field, and then select **Save**.



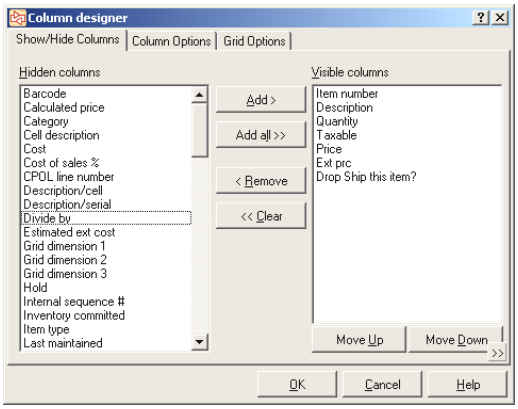
11. Customize "grid layouts" to add the new field to the line item areas of ticket entry, purchase request entry, cash receipts entry, or other similar functions.

Run the entry function and right-click any existing column in the line item area.

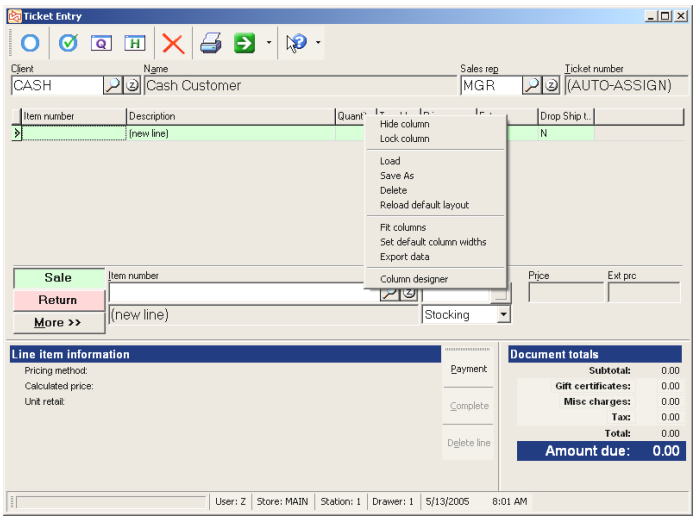
Use the **Column Designer** to add the field to the list of Visible columns.

When completed, click OK.

At the line items area, right-click and select **Save As** and save it with the name "(Default)" to make it the default line item columns that initially display.



To add the new field to other line item column formats that do not display by default, first use **Load** to load the format, use the Column Designer to add the field, and then select **Save As** to resave the format.



Adding Custom Tables

What is a “Custom table”?

Brand-new table and columns in an existing SQL database

What can be done with a “Custom table”?

- Form (menu selection) to maintain or just view the table’s data
- Custom toolbar in table’s menu selection(s)
- Toolbar buttons on existing forms that access the new table
- Reports and forms created in Crystal Reports
- PS Receipts (Crystal formats only)
- Lookups
- Filters
- Zoom windows

Maximums

- Up to 2 billion custom tables
- Up to 1024 columns per table
- Up to 8060 bytes per record
- One primary index per table
- Up to 249 additional indexes per table

Checklist for adding a Custom table

Required steps

1. Add the new physical table and its columns to the database, and identify the primary key and any foreign keys.
2. Create any additional indexes for the table.
3. Add the new table and its columns to the data dictionary.
4. Add a new maintenance or view selection to the Counterpoint menu.

Optional steps

5. Generate a Quick Report template for the new table.
6. Use Crystal Reports to customize existing reports to add fields from the new table.
7. Build a zoom window for the new table and customize other zooms to include new fields.
8. Customize Point of Sale receipts to add fields from the new table (Crystal receipts only).
9. Build a filter for the new table.
10. Build a lookup for the new table.

Adding Custom Tables

Details for Adding a Custom Table

Required steps

1. Add the new physical table and its columns to the database, and identify the primary key for the table and any foreign keys, using Query Editor or the SQL Script Utility.

   The below example illustrates creating a table to track the customer's item number and description for your merchandise. In addition to fields for the customer's item number and description, the new table must contain fields for your customer number and item number.

```
create table  dbo.USR_CUST_ITEMS
( CUST_NO                       T_CUST_NO  not null,
  ITEM_NO                       T_ITEM_NO  not null,
  CUST_ITEM_NO                  T_ITEM_NO  null,
  CUST_ITEM_DESC                T_DESCR    null,
   LST_MAINT_DT                 T_DT       null,
   LST_MAINT_USR_ID             T_USR_ID   null,
   ROW_TS                       timestamp  null,
constraint PK_USR_CUST_ITEMS primary key (CUST_NO, ITEM_NO),
constraint FK_USR_CUST_ITEMS_IM_ITEM foreign key (ITEM_NO)
      references IM_ITEM (ITEM_NO),
constraint FK_USR_CUST_ITEMS_AR_CUST foreign key (CUST_NO)
      references AR_CUST (CUST_NO)
)
go
```

   Auto-updated if fields are present

   A table can only have one primary key constraint. All columns specified within a primary key must be defined as NOT NULL.

   Foreign key constraints are optional, but their use ensures that a record in your table is not 'orphaned' by the deletion of the record to which it is connected.

2. Create any additional indexes for the table.

   Additional indexes provide the ability to sort and look up values in the table in an order other than the primary key.

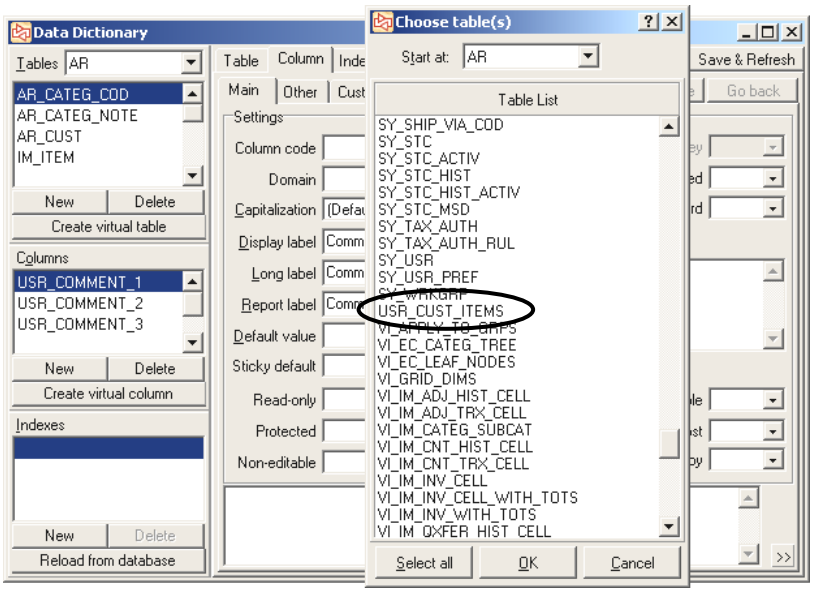
```
create index  USR_CUST_ITEMS_X1 on USR_CUST_ITEMS
( CUST_ITEM_NO,
  CUST_NO,
   ITEM_NO
)
go
```

3. Add the new table and its columns to the data dictionary.

Select **Setup / System / Configuration / Data Dictionary**.

Select any table from the Data Dictionary (not the System Dictionary).

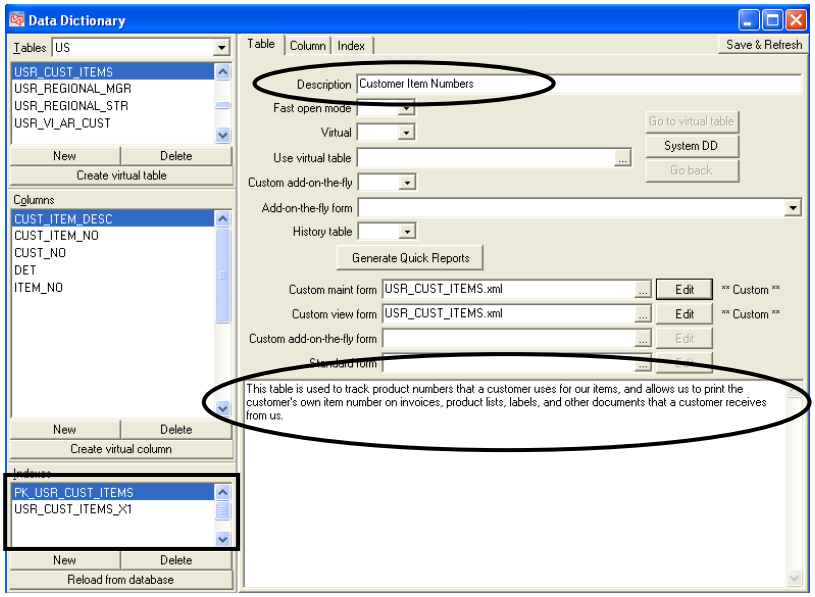
Then click New under Tables and select the new table.



Enter the name of the table for **Description**.

Record the purpose of the table in the blank window. This information prints on the Column Description Report.

The primary key and other indexes you created are also shown.



Click New under Columns to add each of the four fields for the new table.

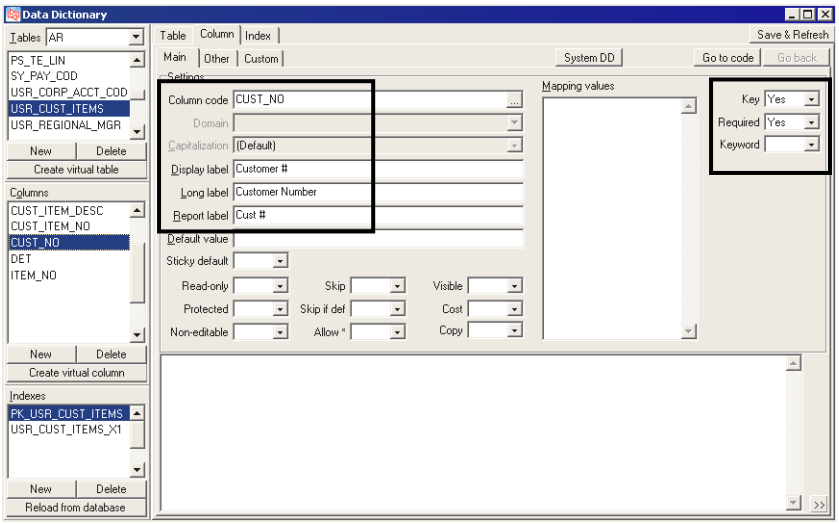
<u>On the Main tab</u> for each column, enter a **Display label**, **Long label**, and **Report label**.

CUST_NO and ITEM_NO each use a **column code**.

Since CUST_NO and ITEM_NO are components of the primary key, they also need to be defined as **Keys** and **Required**.

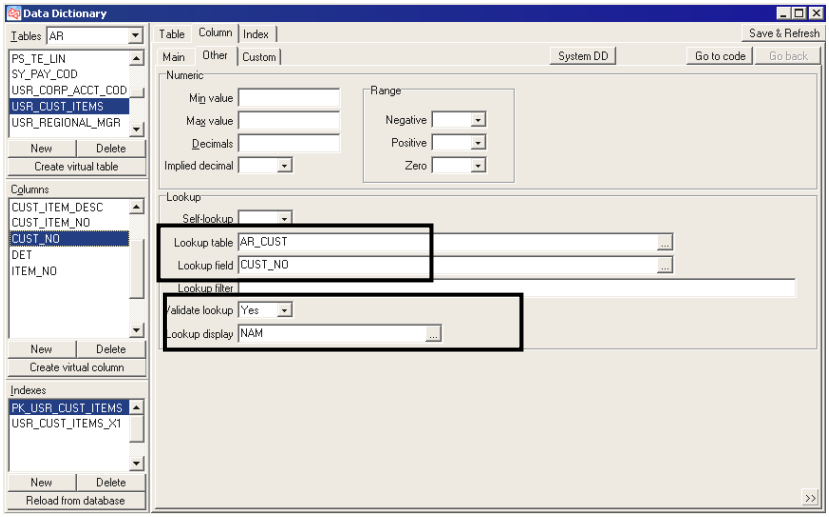
CUST_ITEM_NO is defined as **Required**.

CUST_ITEM_DESCR is defined as **Required** and **Keyword**.



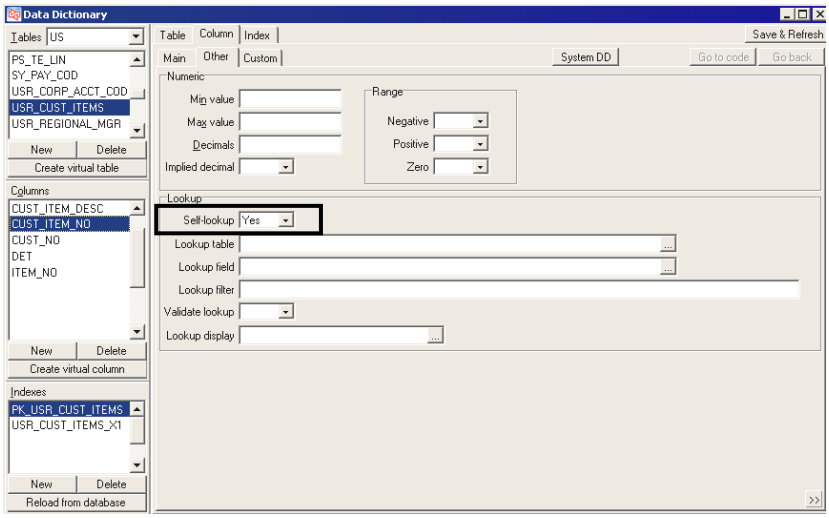
<u>On the Other tab</u> for CUST_NO and ITEM_NO, the usual System table and field are entered for **Lookup table** and **Lookup field** so that lookup controls will be available for both fields.

In addition, for CUST_NO and ITEM_NO, set **Validate lookup** to Yes to ensure that only on-file customer or item numbers can be entered. Also set **Lookup display** to show the customer name or item description from the lookup table.



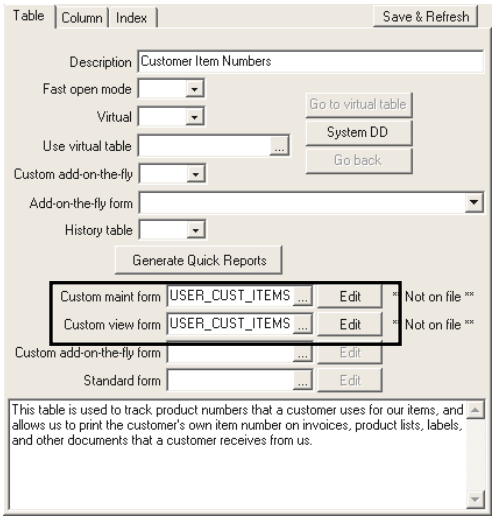
For CUST_ITEM_NO, set **Self-lookup** to Yes since it will be looked up in this custom table with the same column name.

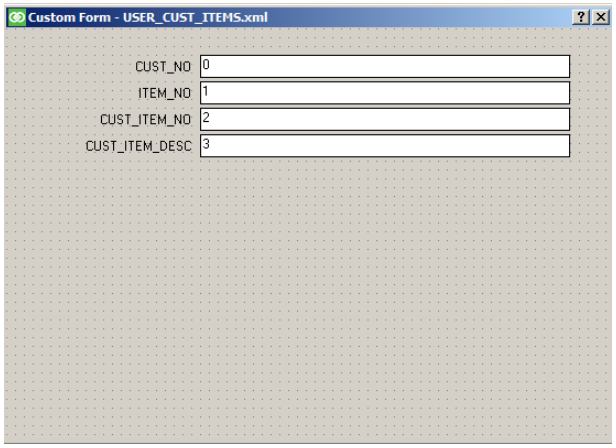
Do not identify a lookup table or lookup field for a "self-lookup" column, nor should you validate lookup for it (this would prevent the ability to add a new customer item number record).



Last, on the Table tab, specify a layout file for **Custom maint form** and **Custom view** and position all of the fields that need to display in the maintenance and view functions (which you will add to the menu). Extend the sizes of each field to allow the customer name and item description to appear.

You can use the same layout file for both the maintenance and view functions.





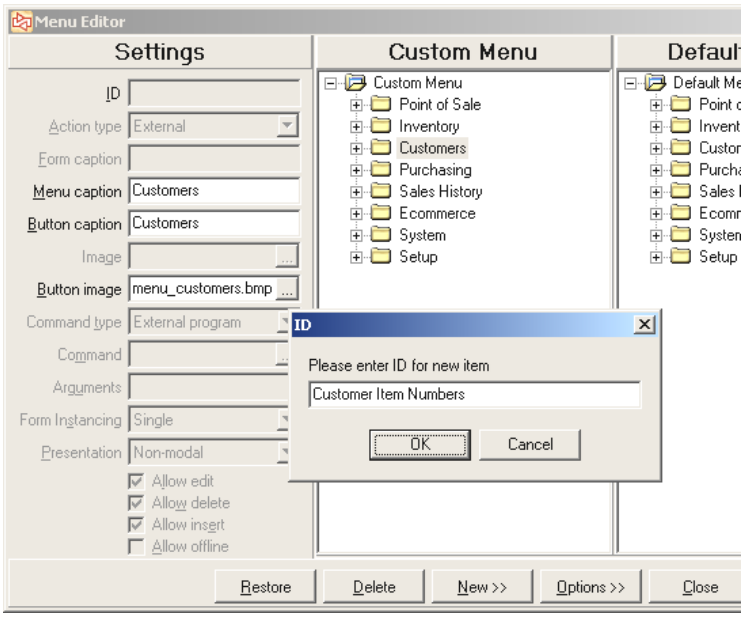
4. Add a new maintenance or view selection to the Counterpoint menu, using **Setup / System / Menu Codes**.

   The menu selections can be used to maintain or view the data in the new table.

Select the **Customers** folder under Custom Menu to place the new function under this folder.

Then click New .

Select **Menu item** and enter an ID of your choice.

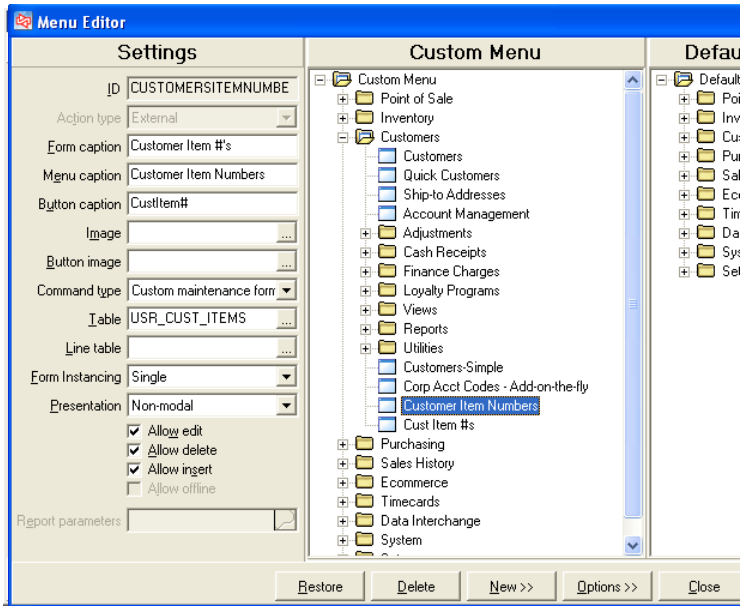


Enter the **Form caption**, **Menu caption**, and **Button caption**.

At **Command type**, select "Custom maintenance form".

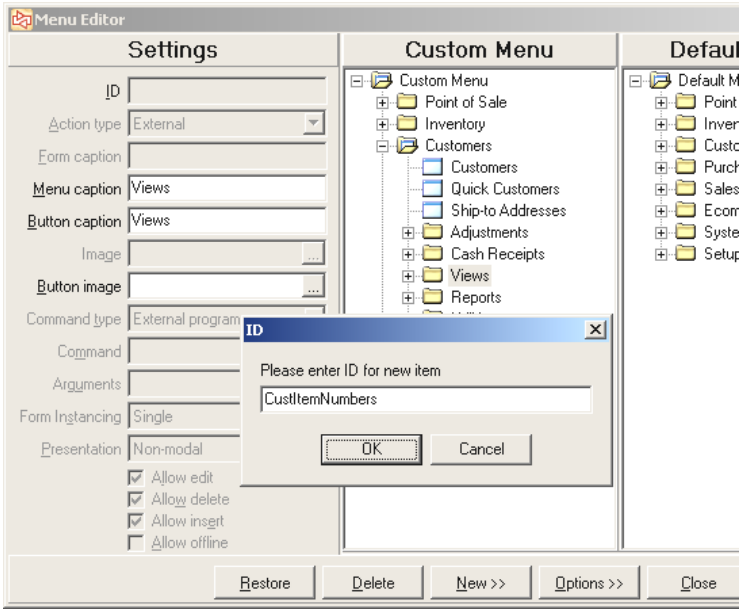
At **Table**, look up and select the table for this function.

Since this function will be used to enter, edit, and delete customer item numbers, all three checkboxes are selected. However, this function will not be available during offline ticket entry.



To create another function for just viewing customer item numbers, select the **Views** folder under **Customers**.

Click New and enter an ID (it must be different than the other menu selection's ID).

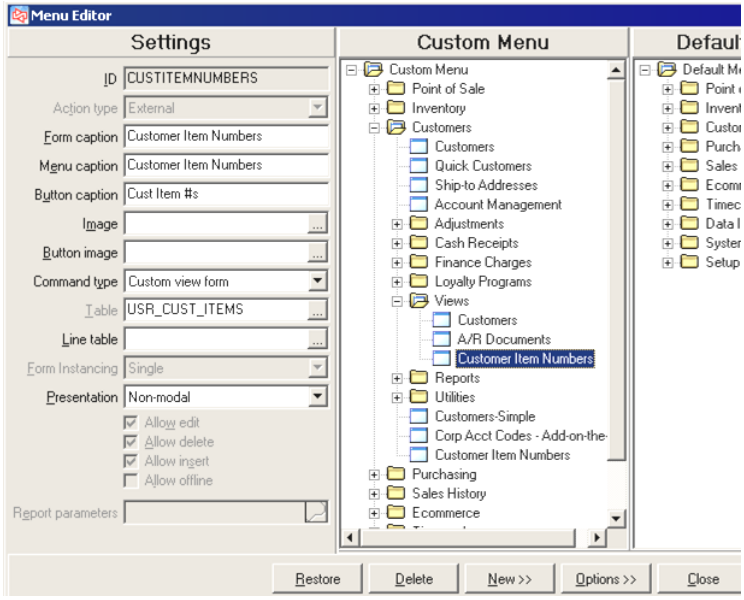


Enter the **Form caption**, **Menu caption**, and **Button caption**.

At **Command type**, select "Custom view form".

At **Table**, look up and select the table for this function.

Since this is a "Custom view form", the checkboxes for **Allow edit, Allow insert,** and **Allow delete** are automatically disabled.



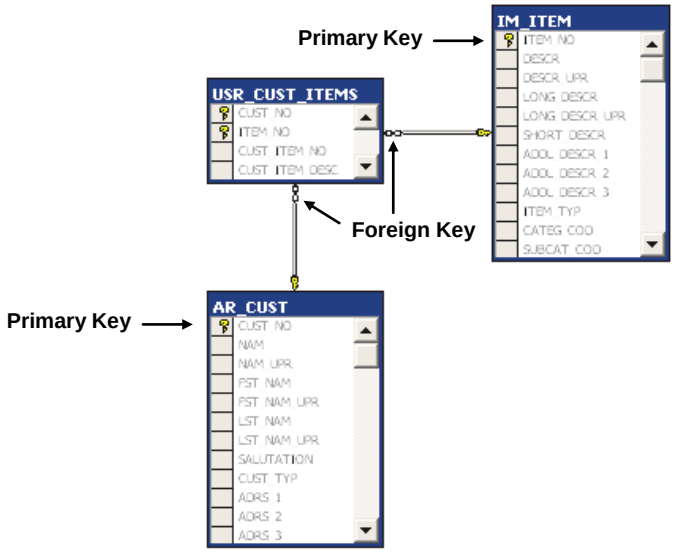
Click  when completed defining both new menu selections, and then save the menu code.

Users who are assigned this menu code through their security code will need to have **Enable customizations** selected in their user preferences (**Setup / System / User Preferences**) in order to run the new menu selections.

Foreign Keys in Custom Tables

Foreign key: One table's primary key in another table

Foreign Keys maintain "referential integrity" in the database by ensuring that changes cannot be made to the primary table if those changes invalidate the link to data in the foreign key table.



To change or delete a record in the primary key table, you must first either:

- delete the foreign key data in the foreign key table (e.g., delete the record from USR_CUST_ITEMS), or

- link the foreign key in the foreign key table to a different primary key (e.g., redefine the record in USR_CUST_ITEMS and associate it with a different item or customer)

Try it Yourself!

This exercise shows how to add a custom table to a database in Counterpoint. Use Query Editor (or the SQL Script Utility) and the DemoGolf database for your work.

The exercise illustrates the following business application: All customers who work for a particular company are A/R charge customers, and each is responsible for their own charges and payments. However, the merchant would also like the ability to report total charges and payments for each 'parent' company. This will be accomplished by:

(1) defining a new Corporate Account Codes table that contains a record (a code ID and description) for each 'parent' company,

(2) creating a new menu selection for maintaining the Corporate Account Codes, and

(3) adding a Corporate Account Code field to the Customers table and requiring a valid entry in this field when adding a new customer in the "CORP" category.

Perform the following steps in the exercise.

1) Use Query Editor to add a new custom table, the Corporate Account Code table (USR_CORP_ACCT_COD), containing these two fields:

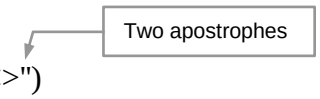
CORP_ACCT_COD	T_COD	not null,
CORP_ACCT_COD_DESCR	T_DESCR	null

Also add the three columns to track last maintenance information. The primary key for the table is CORP_ACCT_COD.

Also add the custom field USR_CORP_ACCT_COD to the AR_CUST table. Run the **RefreshViews.sql** script to ensure that views using AR_CUST are updated.

2) Use **Setup / System / Configuration / Data Dictionary** to add the new USR_CORP_ACCT_COD table and its columns to the Data Dictionary, and to build a layout form that positions the columns.

Also add the field USR_CORP_ACCT_COD to the AR_CUST table with this constraint to require an entry if the customer's category is "CORP".



(COL ('CATEG_COD') <> 'CORP') or (V<>'')

For the constraint error, enter **%f is required for customer category CORP**.

Allow a lookup from this column into the USR_CORP_ACCT_COD table to the CORP_ACCT_COD field. Validate the lookup and display the description of the corporate account code next to the code. (Don't forget to add the new column to the "Standard form" layout for the AR_CUST table.)

3) Use **Setup / System / Menu Codes** and add (or edit) the TRAIN menu code. Add a Corporate Account Codes maintenance function to the **Setup / Customers** menu. Ensure that the MGR security code uses the TRAIN menu code.

4) Select the new menu item, **Setup / Customers / Corporate Account Codes**, and define two corporate account codes, one named ACME ("Acme account") and the other named UNV ("Universal Sports account").

5) Use **Setup / Customers / Categories** and define a new category code of CORP with a description of "Corporate Customers".

6) Select **Customers / Customers** and display any existing customer. Change the category of the customer to CORP and attempt to save the record. You should receive the constraint error message "Corp Account Code is required for customer category CORP".

   Double-click the error message so that you are automatically moved to the "Corp Account Code" field on the **Custom** tab. Look up and select one of the Corporate Account codes that you defined and then save the customer record.

Solutions are provided in Appendix 1.

End of Exercise

Optional steps

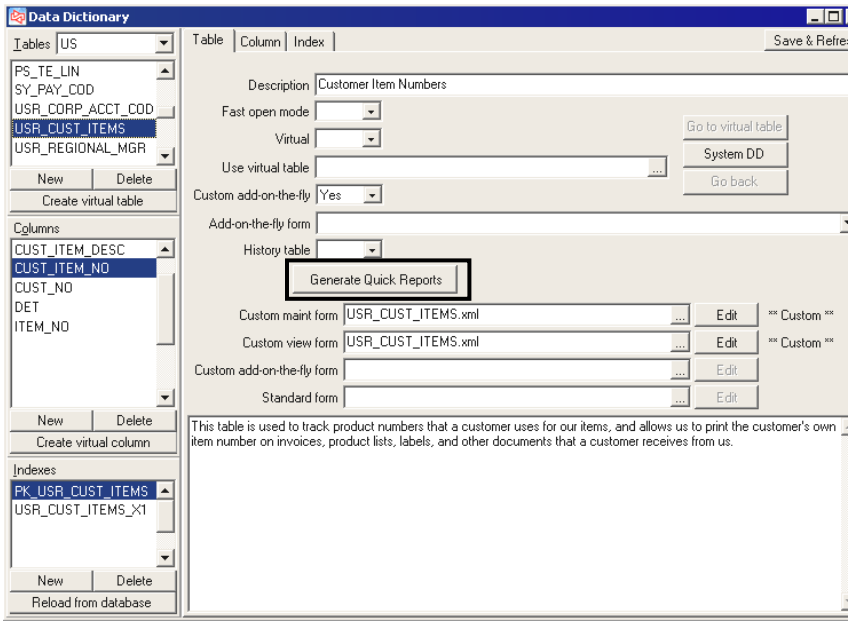
5. Generate a Quick Report template for the new table.

The Quick Report template will be used when anyone clicks Preview or Print to run a Quick Report from a lookup of the new table.

Select **Setup / System / Configuration / Data Dictionary** and select your new table.

On a system with Crystal Decisions Report Creation API License, switch to the Table tab and click Generate Quick Reports .

A new file named <Tablename>.rpt (e.g., **USR_CUST_ITEMS.rpt**) will be produced in the QuickReports folder for the company. Modify this template file using Crystal Reports if you want to change the appearance.



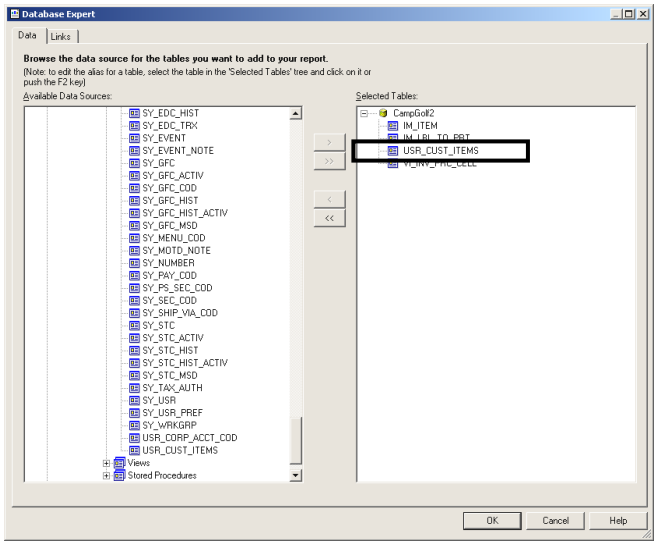
On a system without the Creation API license, start Crystal. Open **Template for Quick Reports.rpt** (located in System / Quick Reports) and:

- select File / Options and ensure that "Insert Detail Field Headings" is enabled on the Layout tab,
- use Database Expert to add your custom table to the report,
- from Field Explorer, multi-select all of the fields for your table and drag them to the Details band (their order in the Details band will not matter), and
- save the report in TopLevel / Configuration / <company> / QuickReports / <tablename>.rpt.

6. Use Crystal Reports to customize existing reports to add fields from the new table.

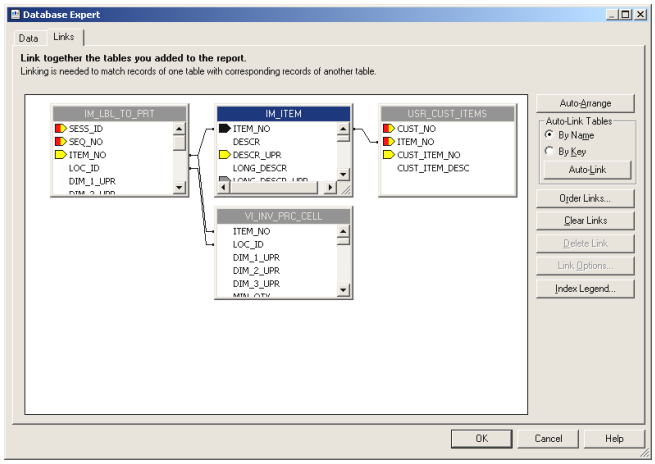
Open the existing report in Crystal Reports. First, use **Database / Set Datasource Location** to "remap" the existing report tables to your database.

Then use **Database / Database Expert** and select the new table.



On the Links tab, link from **ITEM_NO** in the IM_ITEM table to **ITEM_NO** in the USR_CUST_ITEMS table.

Click OK when completed.



Then, in the Field Explorer, select the desired fields from the USR_CUST_ITEMS table and position it in the report or label.

7. Build a zoom window for the new table and customize other zooms to include new fields.

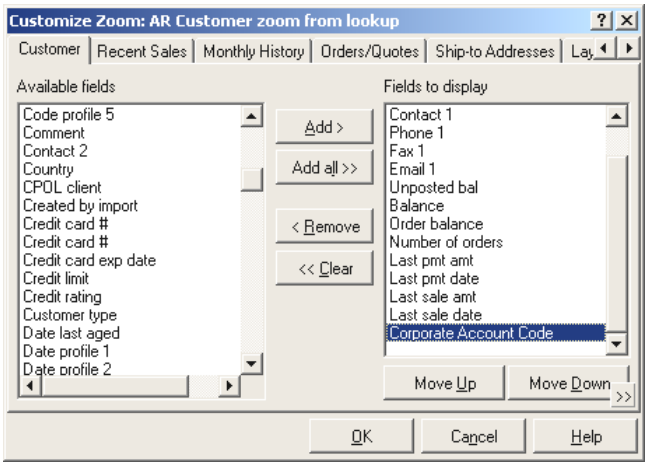
To add new fields in a table to the existing zoom for the table, do a simple customization.

Perform a lookup for the corresponding table. Highlight any record and click Zoom (F6) (or press F6).

In the Zoom window, click Options >> and select **Customize**.

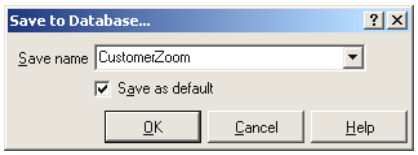
Add the new field to the list of fields to display.

Click OK when completed.



To save the changed zoom window, click Options >> and select **Save as…**.

Name the zoom and indicate if you wish it to be the default zoom.



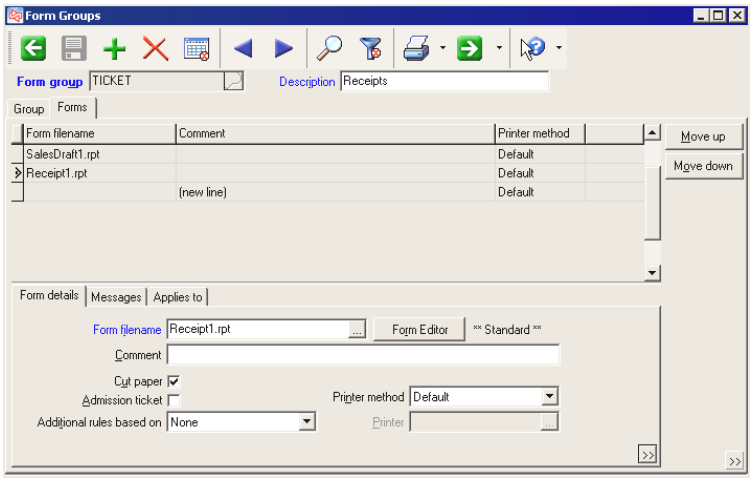
To build a brand-new zoom for a table (such as the Customer Items table), select **Setup / System / Configuration / Zoom Dictionary** and add a new zoom dictionary entry. Refer to *Customizing Zoom Windows* later in this section of the manual for instructions on customizing the Zoom dictionary.

Adding Custom Tables – Customizing Receipts

8. Customize Point of Sale receipts to add fields from the new table (Crystal receipts only).

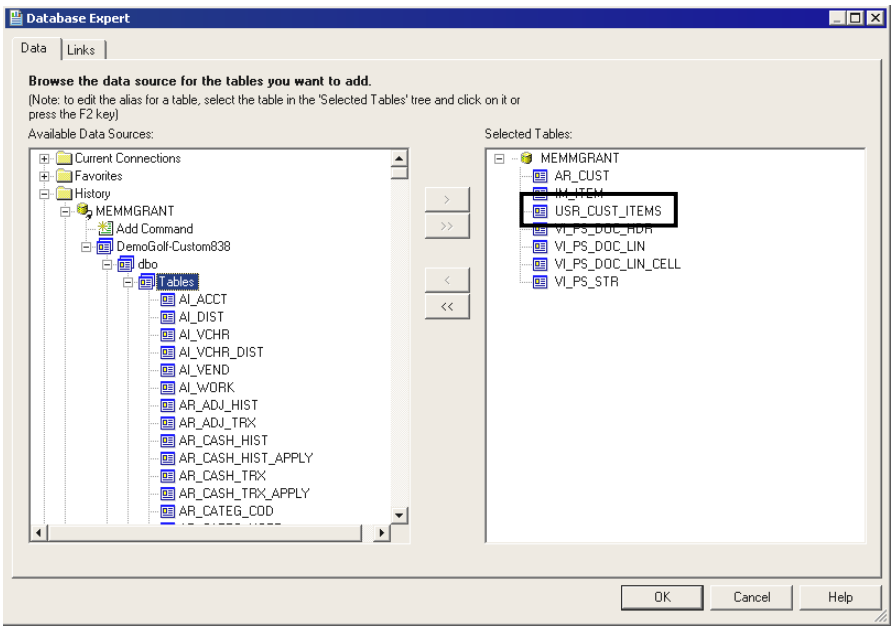
Use **Setup / Point of Sale / Form Groups** and display each form group that uses a receipt form to which the new fields must be added.

For each group, switch to the Forms tab, select the form to be changed (it must be a ".rpt" form) and click Form Editor to customize the form using Crystal Reports.



Using Crystal Reports, first use **Database / Set Datasource Location** to "remap" the existing tables in the form to your database.

Then use **Database / Database Expert** and select the new table.

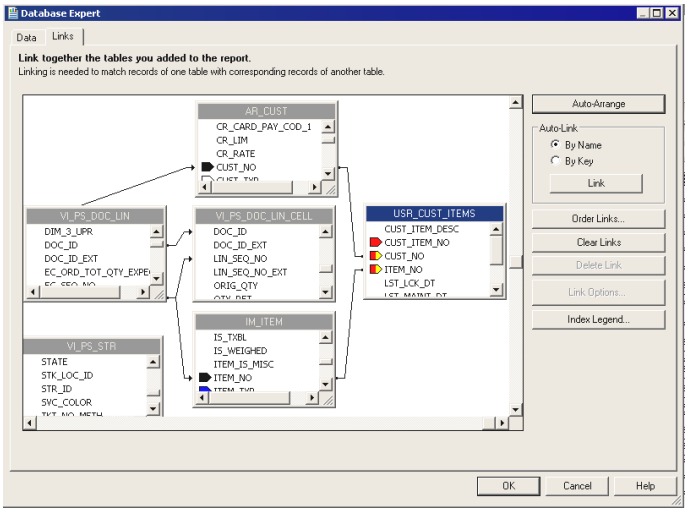


On the Links tab, link from **ITEM_NO** in the IM_ITEM table to **ITEM_NO** in the USR_CUST_ITEMS table.

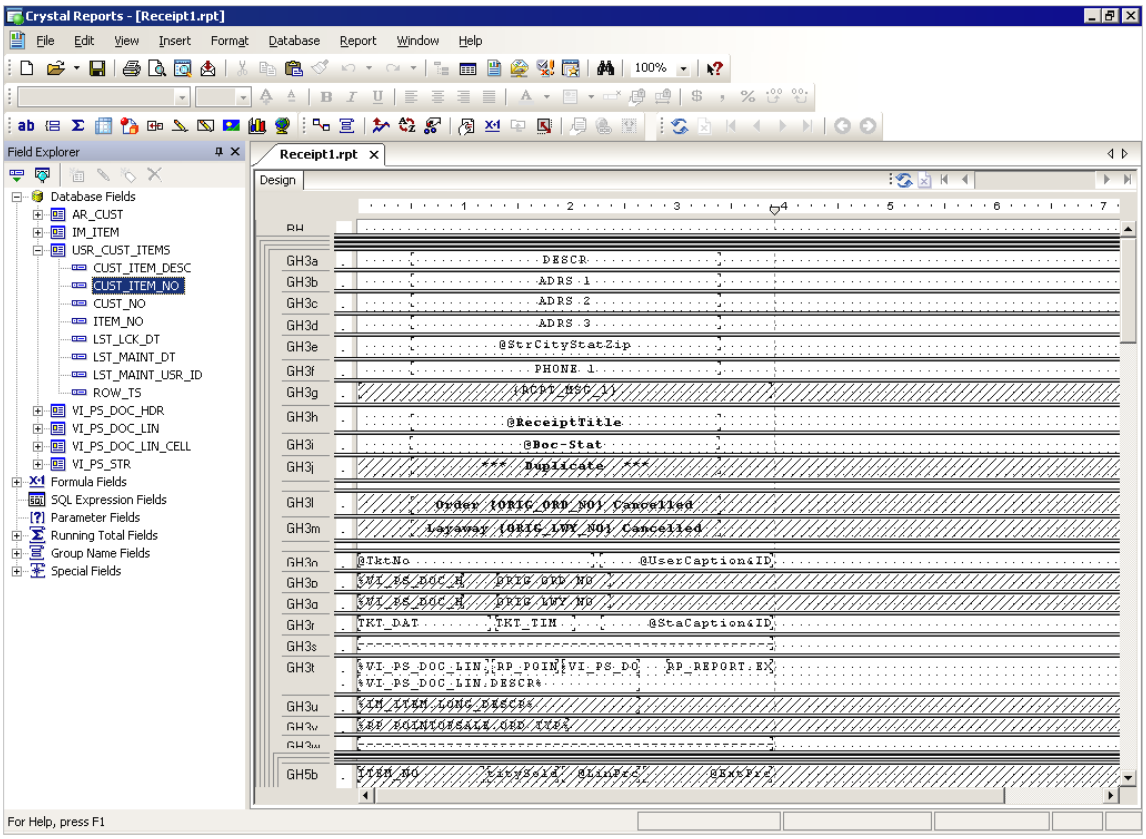
Also link from **CUST_NO** in the AR_CUST table to **CUST_NO** in the USR_CUST_ITEMS table.

Do not change any existing links between other tables.

Click OK when completed.



Finally, in the Field Explorer, select any field from the new table and position it in the form.



Save your changed form, using the default filename, in the PSForms folder for your company on the server.

Make the same changes to Receipt1History.rpt so that the new fields also print on receipts that are printed from history.

Adding Custom Tables – Customizing Filters

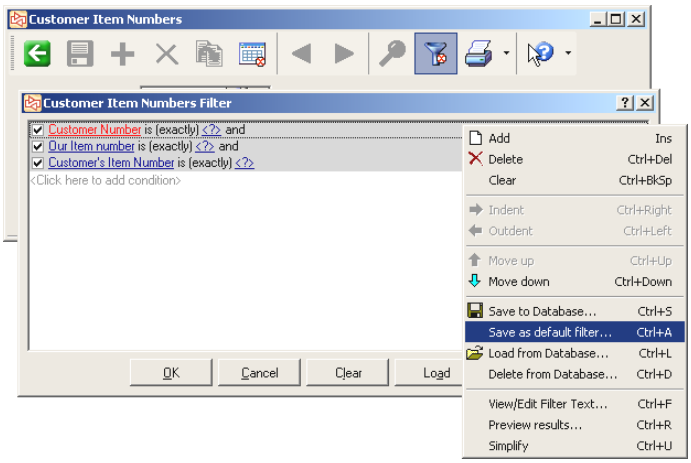
9. Build a filter for the new table.

Run the new maintenance or view menu selection for the table and click  on the toolbar to set a filter.

When the Filter window opens, right-click and select **Customize**. Add the desired fields to the filter.

When finished, click 

and select **Save as default filter** to save it as the default filter for the new table.

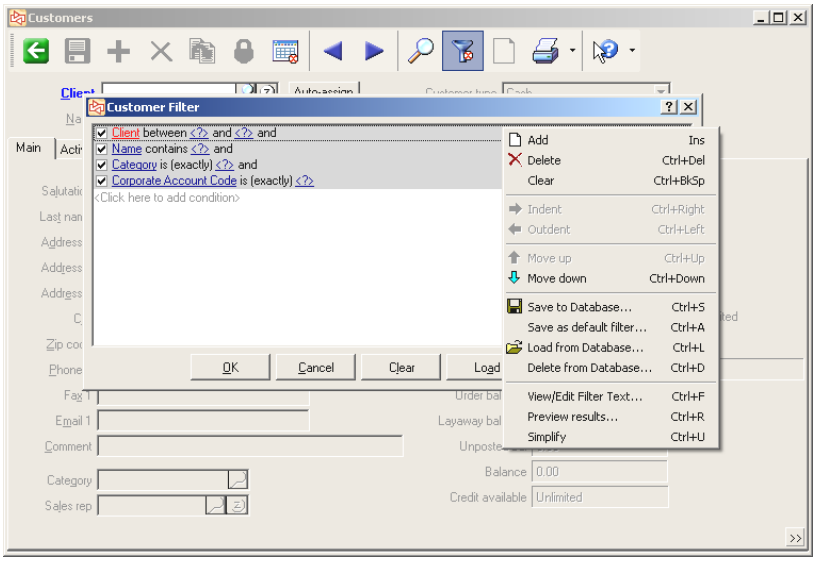


If you added a field from the new table to another table (e.g., AR_CUST) and you want to include the new field in a filter for the second table, run the maintenance menu selection for the second table (**Customers / Customers**).

Right-click and select to **Customize**.

Add a condition for the new field to the default filter, and then select **Save as default filter**.

To add the new field to a filter that is not the default, first use **Load from Database** to load the existing filter. Add the new condition and then use **Save to Database** to resave it using the same name or a different name.



10. Build a lookup for the new table.

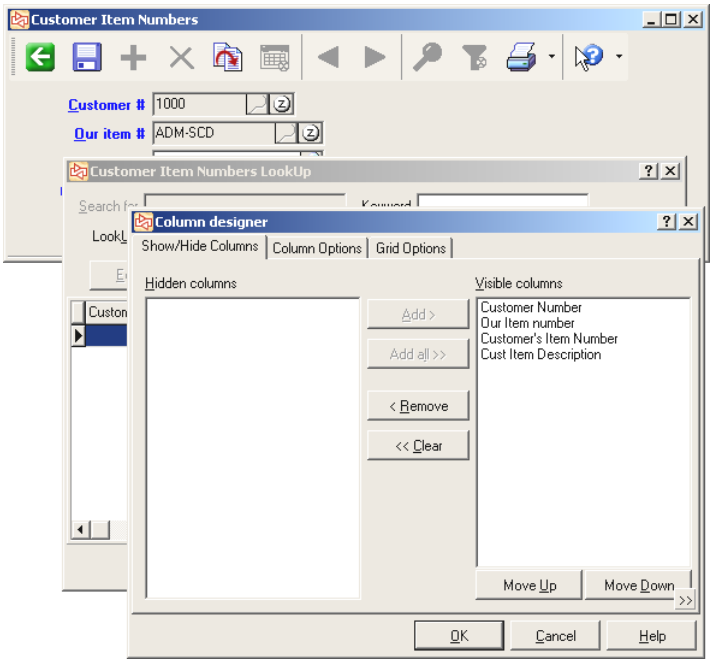
Select the maintenance or view menu selection for the new table and select to do a lookup.

Click Options >> and use the Column Designer to add or remove fields in the list of Visible columns.

Click OK when done.

Then click Options >> and select **Save**.

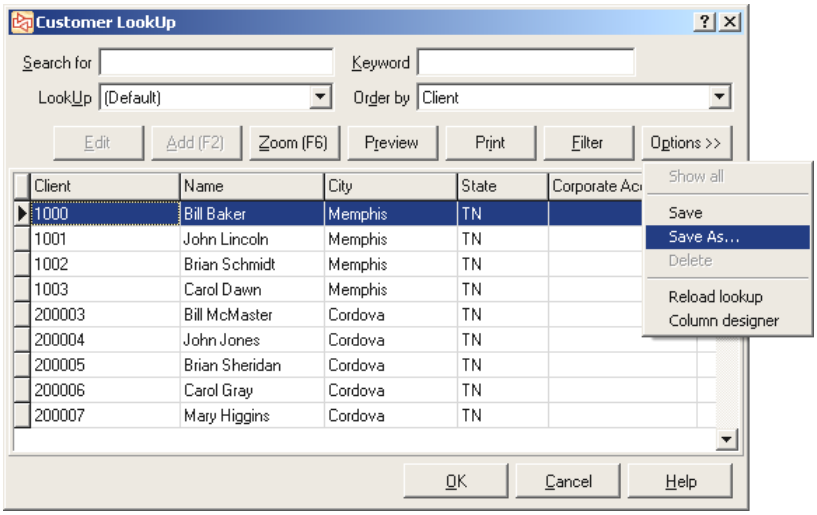
To build other non-default lookups for the table, customize the lookups and save them with a different name.



To add a new field to an existing lookup for a table (such as adding "Corporate Account Code" to the Customer Lookup), select the maintenance or view menu selection for that table and load the corresponding Lookup.

Click Options >> and use the Column Designer to add the field to the list of Visible columns.

Click Options >> and select **Save** when completed to resave the lookup, or use **Save as** to save it with a different name.



Adding Custom Tables

Try it Yourself!

In the prior exercise, you added the Corporate Account Code table to the Camptown Golf database, created a menu selection to maintain the codes, and added the Corporate Account Code field to the Customer table.

In this exercise, continue the customization by performing the following steps.

1. Either generate or use Crystal to create a Quick Report template for the Corporate Account Code table so that users will be able to preview or print a list of the records in this new table.

   Also regenerate or use Crystal to modify the Quick Report template for the Customers table so that the new Corporate Account Code field is included.

   Skip this step if Crystal Reports is not installed on the computer you are using in class.

2. Customize the Zoom for the Customers table so that the new Corporate Account Code field displays.

3. Customize the receipt (RECEIPT1.rpt) so that the description of the customer's Corporate Account Code prints next to the customer's phone number in the group footer.

   Skip this step if you do not have Crystal Reports installed.

4. Customize the default filter for the Customers table to add the new Corporate Account Code field.

5. Build a default lookup for the new Corporate Account Code table.

   Also add the Corporate Account Code field in the Customers table to the default lookup for the Customers table.

Solutions are provided in Appendix 1.

End of Exercise

Custom Tables have some limitations

A. When you customize a toolbar for a specific custom form, rather than for a group of forms, the toolbar is saved under the form caption (title bar) name. If you later change the Form Caption name in the menu code, the toolbar previously associated with it be “disconnected”. You will have to redefine the toolbar and save it so that the new form caption is captured.

B. Current “Help” choices will not display help on custom forms. However, you can customize the toolbar to replace the standard System “Help” choices with custom actions that open your custom .hlp file.

C. While you can control the position and tab sequence of custom fields on a form, you cannot control the location of the field label. Field labels are always displayed on the left of the associated control.

D. You cannot enable or disable a field conditionally based on the entry made in a different field.

Adding a Custom View Table

View Table an alternate “view” of an existing table, with its own data dictionary settings

- Use to produce a maintenance form or provide a lookup with
  - selected fields that have different settings than the “normal” maintenance form or lookup
  - fields from multiple tables in the database
  - data is not actually stored in view table

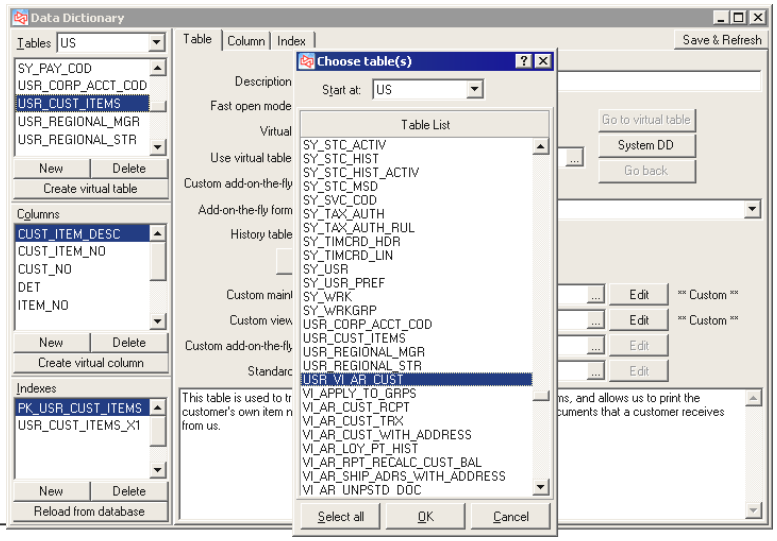
1. Create the View Table in the database, using Query Editor.

```
create view dbo.USR_VI_AR_CUST as
select  CUST_NO, NAM,
        ADRS_1, ADRS_2, ADRS_3,
        CITY, STATE, ZIP_COD,
        CATEG_COD, SLS_REP, STR_ID
from
       AR_CUST
go
```

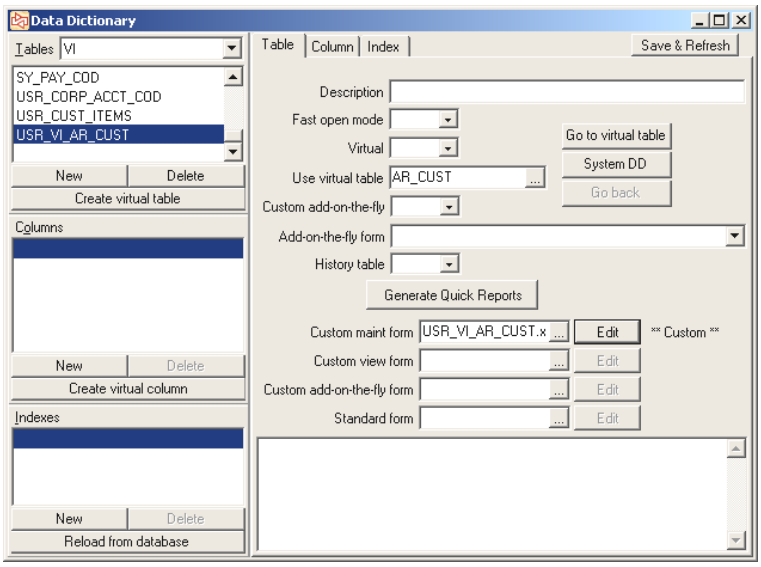
   It is not necessary to identify a primary key in the database for a view table. Also, do not add constraints to the view table; put constraints in the ‘real’ table instead.

2. Use **Setup / System / Configuration / Data Dictionary** to add the view table and its columns to the dictionary.

   Click New under Tables and select the new view table from the list.



On the Table tab, indicate if column settings should be obtained from a virtual table.



Column settings

It is only necessary to add columns in the view to the data dictionary if the setting for any column needs to vary from the standard setting in the virtual table or from an existing column code.

XML layout form

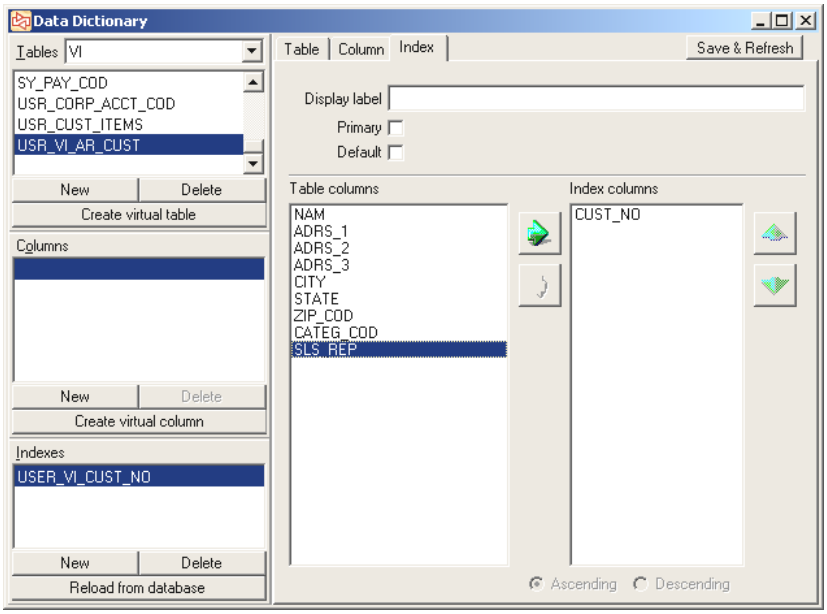
If you plan to provide a menu selection to maintain data using the view table, specify and build the XML layout form at **Custom maint form**.

(Note that it may not be possible to maintain data using the view if data is pulled from multiple tables, if it groups records, or does math [e.g., totals].)

On the Index tab, add a logical index for the view.

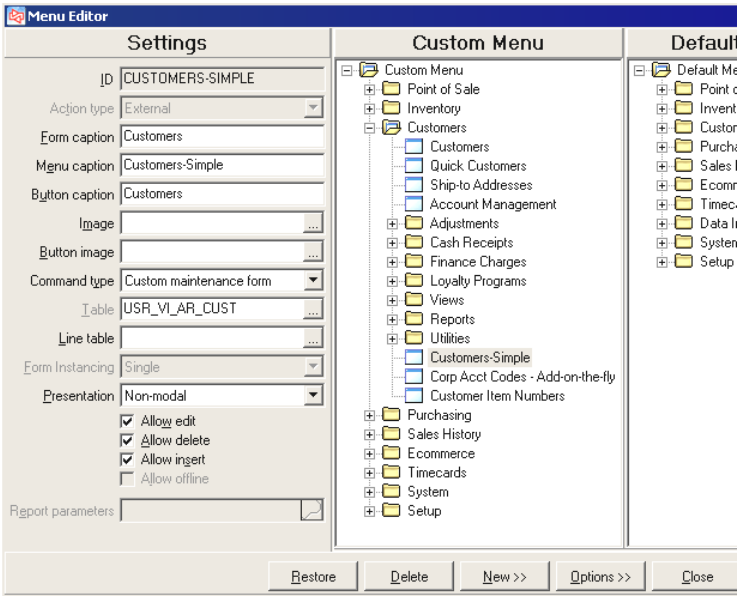
Click New under Indexes and name the index.

Use the same columns that are in the primary key for the 'normal' table(s).



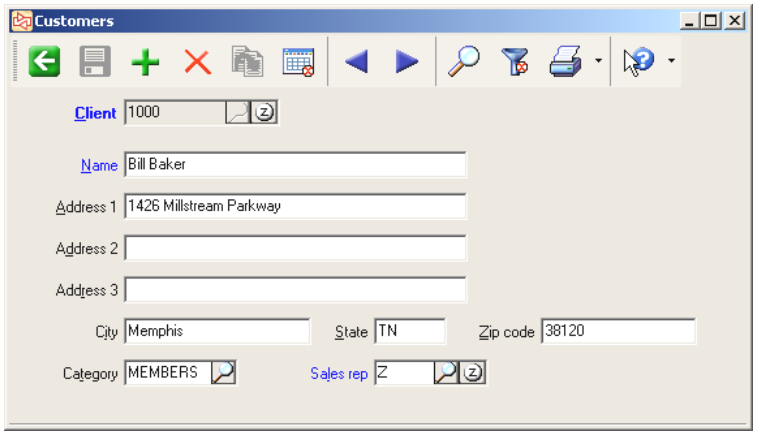
3. Use **Setup / System / Menu Codes** to add a new menu item that allows maintenance of the fields in the new view table.

Select ”Custom maintenance form” for **Command type** and identify the view table for **Table**.



You can remove the regular maintenance selection (e.g., **Customers / Customers**) from this menu code, to allow the user to only access the simple maintenance form for the new view table.

When the user runs the new menu selection, it displays this form where they can add or edit customers.



Try It Yourself!

In this exercise, you will create a view table of AR_CUST data and set up a “Simple Customers” menu selection to maintain customer data using the view table.

Use the preceding pages for assistance.

Adding Custom Fields to .rdlc Forms

VI_RECEIPT_HEADER
VI_RECEIPT_LINE_ITEMS

Two standard view tables for use to add new fields to .rdlc forms

View Table	Predefined Columns
VI_RECEIPT_HEADER	DOC_ID
VI_RECEIPT_LINE_ITEMS	DOC_ID, LINE_SEQ_NO

This example shows how to add the user's name (SY_USR.NAM) to .rdlc forms.

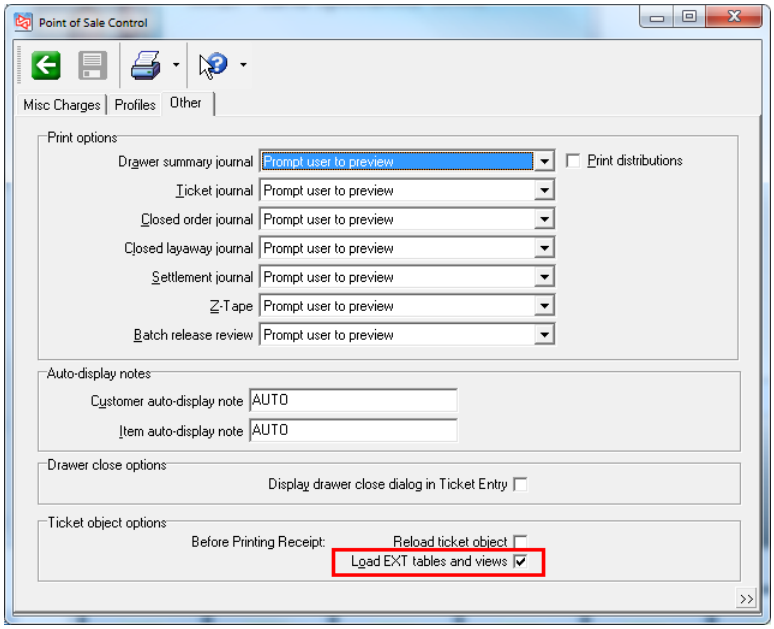
1) Use Query Editor or SQL Script Utility to add fields to corresponding view table

```
ALTER view dbo.VI_RECEIPT_HEADER as
Select PS_DOC_HDR.DOC_ID, SY_USR.NAM
From PS_DOC_HDR, SY_USR
Where PS_DOC_HDR.USR_ID = SY_USR.USR_ID
```

2) Start Counterpoint and select Setup/Point of Sale/Control.

Enable the Load EXT tables and views check box.

This forces data to load from the view, rather than just from memory on a workstation.

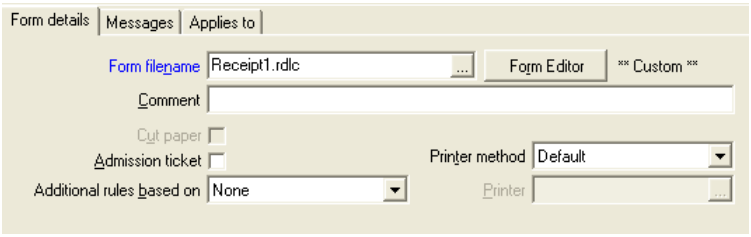


3) Run Ticket Entry or Touchscreen and enter a ticket.

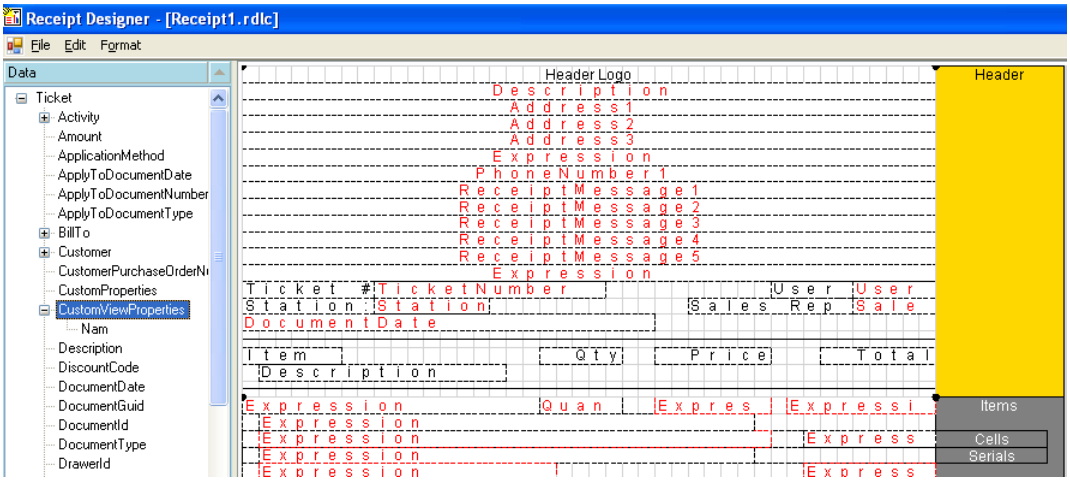
You will need to enter a ticket and print (or preview) an .rdlc receipt.

This step is necessary to actually load the view table so that Counterpoint "sees" the new field in the view table.

4) Use Setup/Point of Sale/Form Groups and use the Form Editor to edit the appropriate .rdlc form.



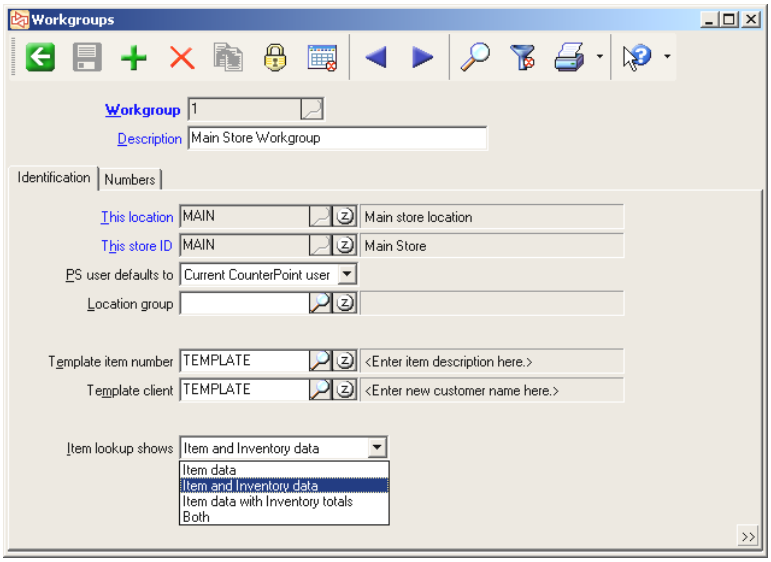
The new field will appear under the CustomViewProperties group for the Ticket or TicketLineItems data browser. You can position and set the properties of the custom field on the .rdlc form in the usual way.



Using a Custom View for Item Lookups

VI_IM_ITEM_CUSTOM_LKUP Custom View

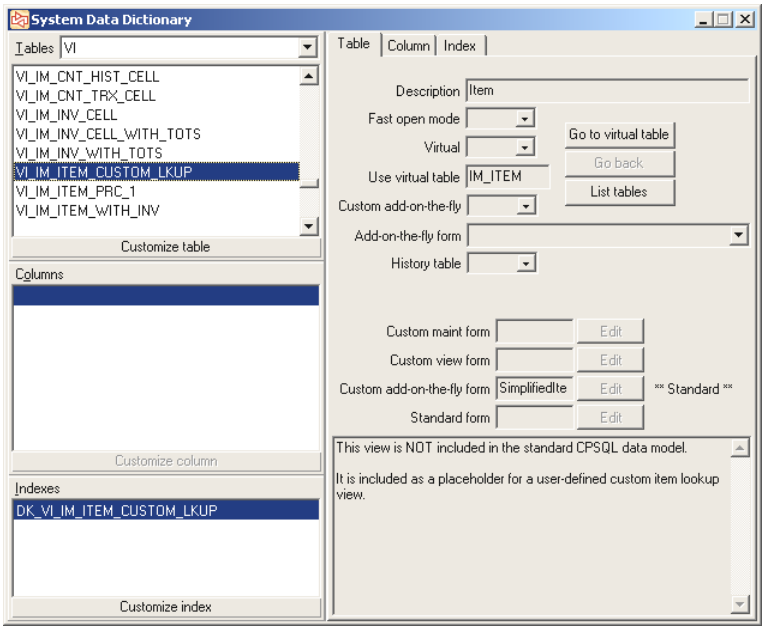
If VI_IM_ITEM_CUSTOM_LKUP exists in database, **Setup / System / Workgroups** includes choice "Custom Lookup" at **Item lookup shows**.



Use Query Editor or SQL Script Utility to add VI_IM_ITEM_CUSTOM_LKUP with desired columns to database.

System Data Dictionary already includes entries for this table.

Don't need to add entries for columns in data dictionary if existing column settings for IM_ITEM are used.



Adding the VI_IM_ITEM_CUSTOM_LKUP Custom View

Try It Yourself!

In this exercise, you will create the VI_IM_ITEM_CUSTOM_LKUP view table that combines data from the IM_ITEM, IM_INV, and PO_VEND_ITEM tables.

You will then set your workgroup to use that custom view table for item lookups and you will use that lookup when entering the line items on a purchase request.

1. Create the VI_IM_ITEM_CUSTOM_LKUP view table, using Query Editor.

```
Create view dbo.VI_IM_ITEM_CUSTOM_LKUP as
Select  I.*,
        V.LOC_ID, V.MIN_QTY, V.MAX_QTY, V.QTY_COMMIT,
        V.QTY_ON_HND, V.QTY_ON_PO, V.QTY_ON_BO,
        V.QTY_ON_XFER_OUT, V.QTY_ON_XFER_IN, V.QTY_AVAIL,
        V.NET_QTY, V.LST_COST as INV_LST_COST,
        V.COST_OF_SLS_PCT, V.FST_RECV_DAT,
        V.LST_RECV_DAT as INV_LST_RECV_DAT,
        V.FST_SAL_DAT, V.LST_SAL_DAT, V.LST_ORD_DAT,
        V.NXT_EXPECTD_DAT, V.NXT_EXPECTD_PO, V.AVG_COST,
        V.QTY_ON_ORD, V.QTY_ON_LWY, V.QTY_ON_SO,
        P.VEND_ITEM_NO as VEND_VEND_ITEM_NO,
        P.PUR_UNIT,  P.MIN_ORD_QTY, P.ORD_MULT,
        P.LEAD_DAYS, P.UNIT_COST
FROM IM_ITEM I
        inner join IM_INV V
        on V.ITEM_NO = I.ITEM_NO
        left join PO_VEND_ITEM P
        on P.VEND_NO = I.ITEM_VEND_NO
        and P.ITEM_NO = I.ITEM_NO
```

Notice that this command does not contain a WHERE clause to connect the tables. The WHERE clause works well when the tables are based on equality – that is, where there's at least one matching record in each table.

In this case, however, an item may not have an IM_INV record. Or ITEM_VEND_NO may be blank for an item, Or, even if ITEM_VEND_NO is not blank, there may not be a matching record in the PO_VEND_ITEM table.

So instead of using a WHERE clause, you modify the FROM clause to use INNER JOIN for the IM_INV table (to eliminate items without inventory records) and LEFT JOIN for the PO_VEND_ITEM table (to show an item even if there is no matching PO_VEND_ITEM record).

2. Since many of the columns in the VI_IM_ITEM_CUSTOM_LKUP view table already have column codes in the System Data Dictionary, you will only need to add Data Dictionary settings for the columns without column codes in order to give them display labels.

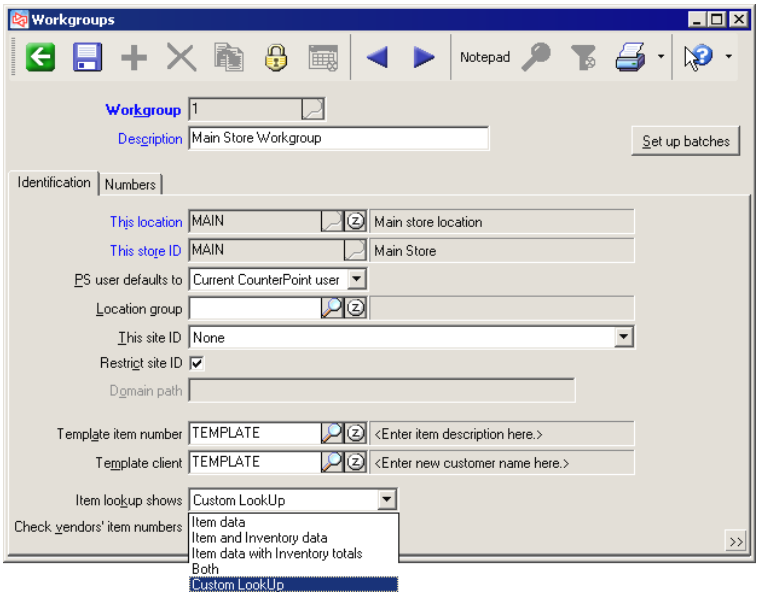
   One way to determine which columns need display labels is to view them in the Column Designer after doing a lookup that uses the custom view table.

   Continue with the next two steps of the exercise to get the list of columns that need to have display labels added in the Data Dictionary.

3. Select **Setup / System / Workgroups**.

   For workgroup 1, set "Item lookup shows" to **Custom Lookup**.

   Save your change.

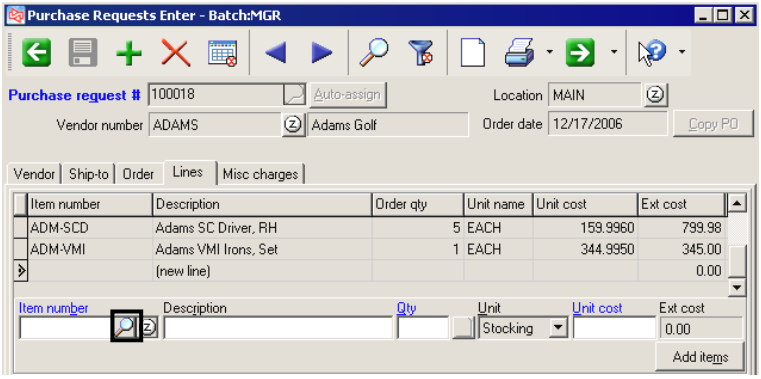


4. Select **Purchasing / Purchase Requests / Enter**.

   If there's already an unposted purchase request available, display it and switch to the Lines tab.

   If there is no unposted purchase request, create one and switch to the Lines tab. (You'll delete this purchase request when finished.)

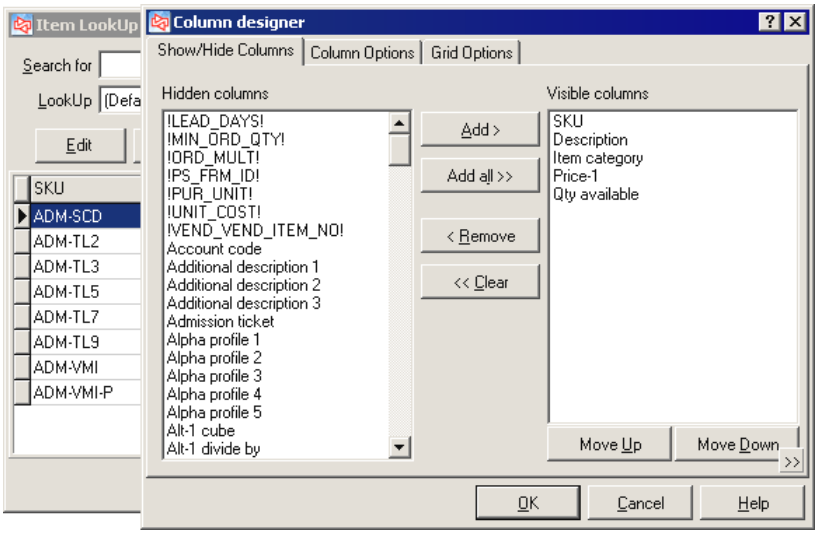
   Click  at "Item number" to do an item lookup.



When the Item Lookup appears, select **Options / Column Designer**.

Scroll to the top of the Hidden Columns list to see the columns with "!" surrounding the name. These are the columns that are missing display labels.

Jot down the names of these columns.



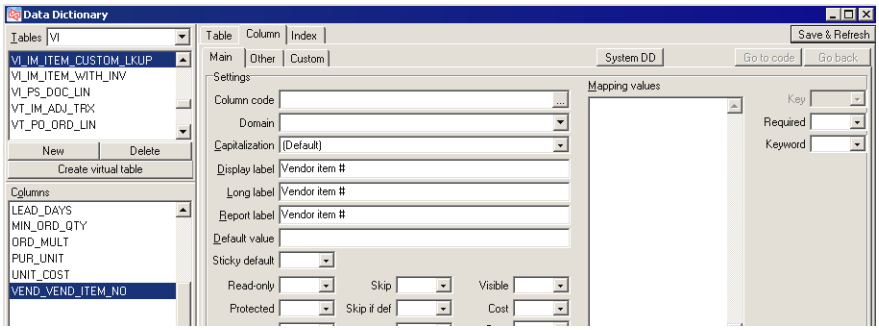
Close the Column Designer and Item Lookup when finished noting the names.

5. Select **Setup / System / Configuration / Data Dictionary**.

From the System Data Dictionary, select to customize VI_IM_ITEM_CUSTOM_LKUP.

From the Data Dictionary, click New under Columns and add each column that is missing a display label. (You can CTRL-click to select them.)

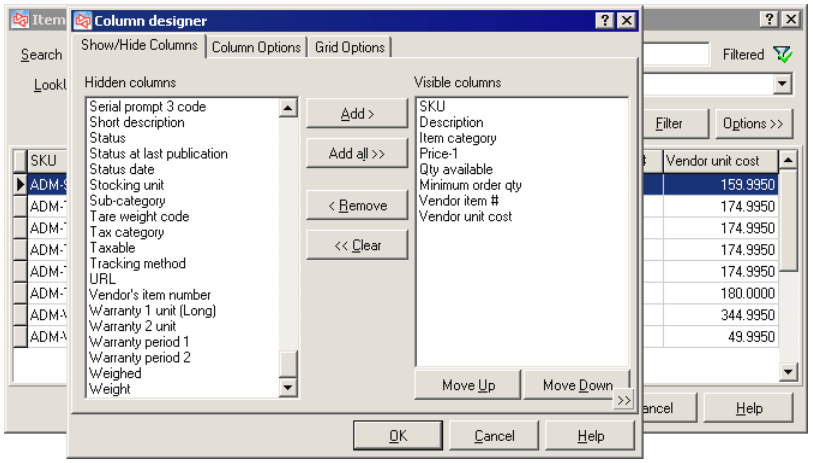
Set a display label for each column.



6. Return to **Purchasing / Purchase Requests / Enter**.

On the Lines tab, use the Column Designer from the Item Lookup to see that display labels appear for all columns.

Modify the item lookup to use fields from your view table.



End of Exercise

Customizing Zoom Windows

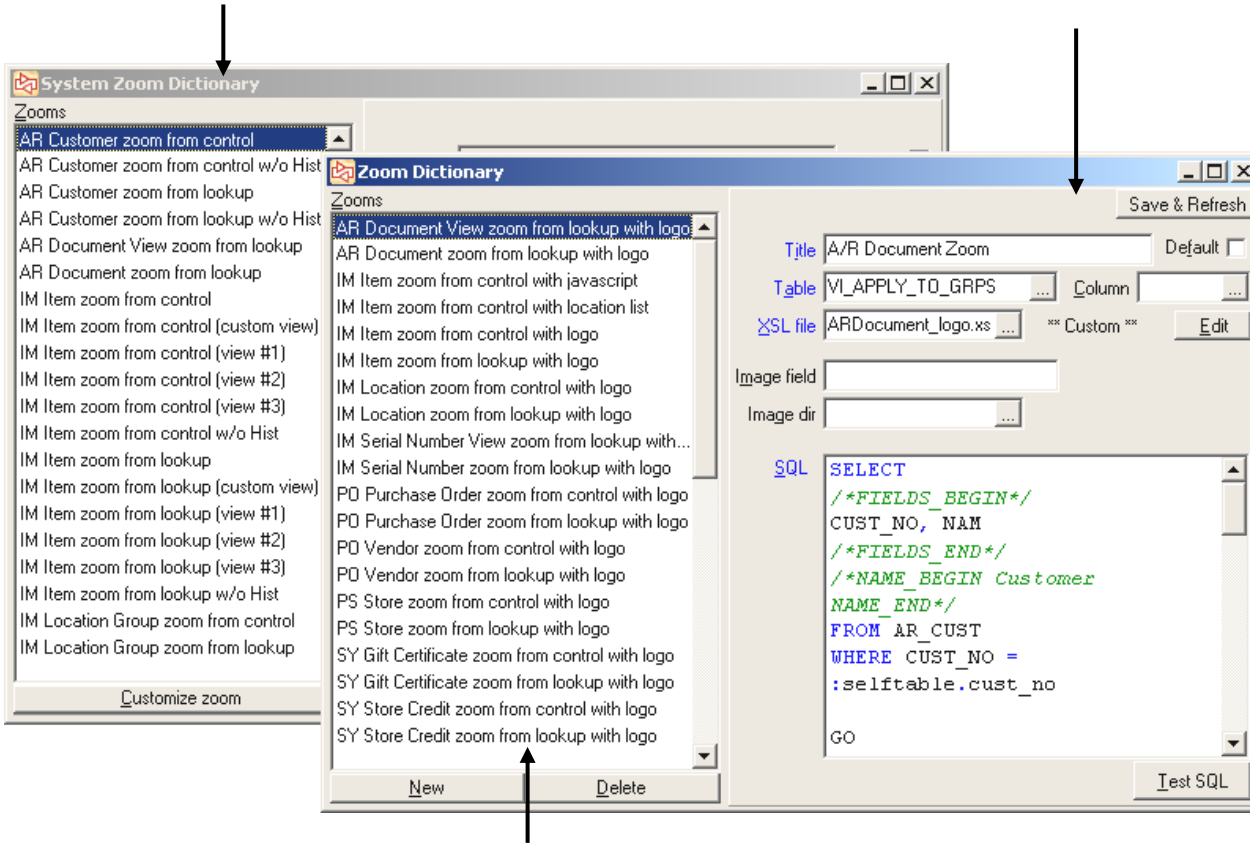
Select Setup / System / Configuration / Zoom Dictionary

- Use to build a new zoom or change appearance of a zoom
- For new zoom, adds Zoom (F6) in lookup and  at column
- Security code must allow you to customize zoom dictionary
- User preferences must have customizations enabled

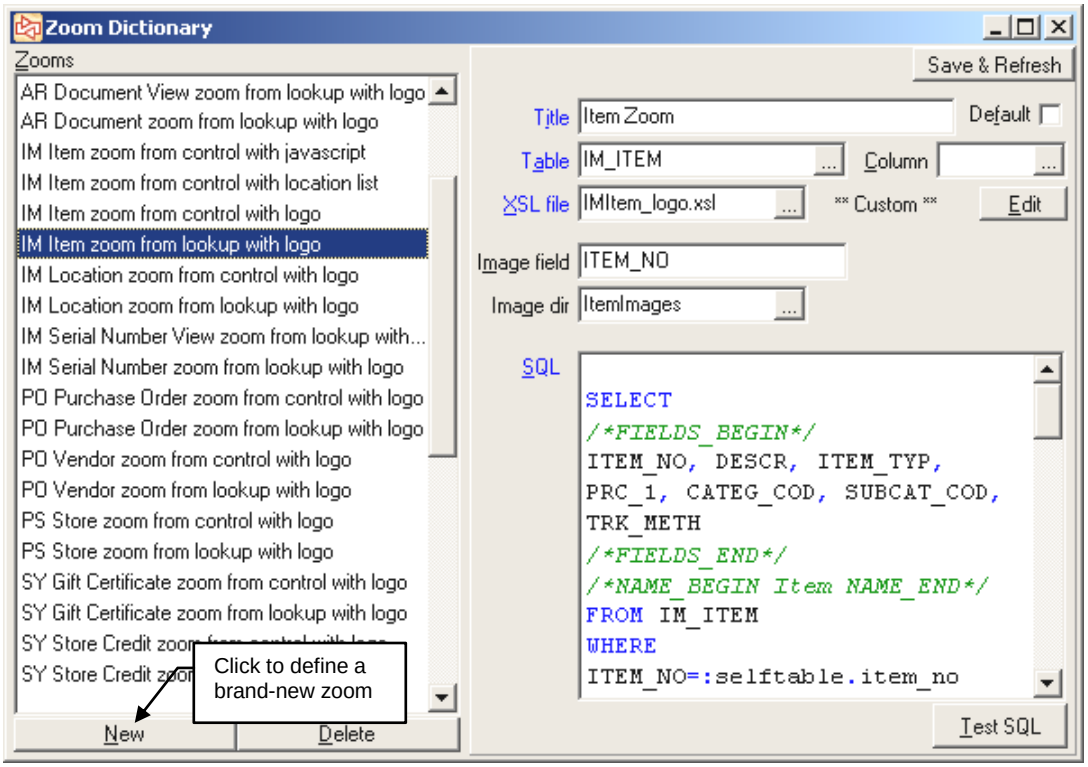
To edit an existing zoom:

Select the zoom you want to change from the System Zoom Dictionary, and click Customize zoom

Then make your changes here



Zooms already customized for DemoGolf database



Title *	Name of Zoom that appears in the title bar of the Zoom window
Default	Initial zoom that displays for the table and field
Table *	The physical table in the database to use for this zoom
Column	Physical field in the table to which this zoom is connected. Required for a zoom from a field. Leave blank for a zoom from a lookup.
XSL file *	Name of the .XSL file (“style sheet”) to use for formatting the zoom. Must reside in System\Zooms or [Company]\Zooms directory. Click **Edit** button to edit .XSL file or create a new one.
Image field	Field in the first SQL query to use as a base for identifying associated image files in the zoom
Image dir	Directory below Company’s directory where associated image files are located
SQL *	SQL query (or queries) that return the data for the zoom. Separate each query with the word “go” on a separate line. Click **Test SQL** button to test query for valid syntax.

* = entry required for this field

To save the modified zoom, select a different zoom, click Save & Refresh, or close the form and answer “Yes” to **Save current changes?**. The modified zoom is automatically saved in the Zooms directory for the company.

Customizing Zoom Windows

XSL file

Controls the appearance of the zoom window.

Use: - ***GenericZoom.xsl*** (up to 5 queries allowed, last four on buttons),
- ***GenericDocumentZoom.xsl*** (up to 6 queries, last four on buttons), or
- create your own .xsl file (if you know XSL/HTML).

Hyperlink to call a Counterpoint form looks like this:

CPV8:FormCaption=[Zoom Customer Notes] FormClass=[TfrmCustomerNotes] FormFilter=[(CUST_NO=@CUST_NO@)] DefaultField=[CUST_NO] DefaultValue=[@CUST_NO@] DisableControl=[edtCustNo] EditActions=[E]

To control whether deletes (D), inserts (I), and/or edits (E) are allowed, add EditActions=[DIE] to the hyperlink. Order and capitalization do not matter. Omitting EditActions results in read-only access.

SQL query

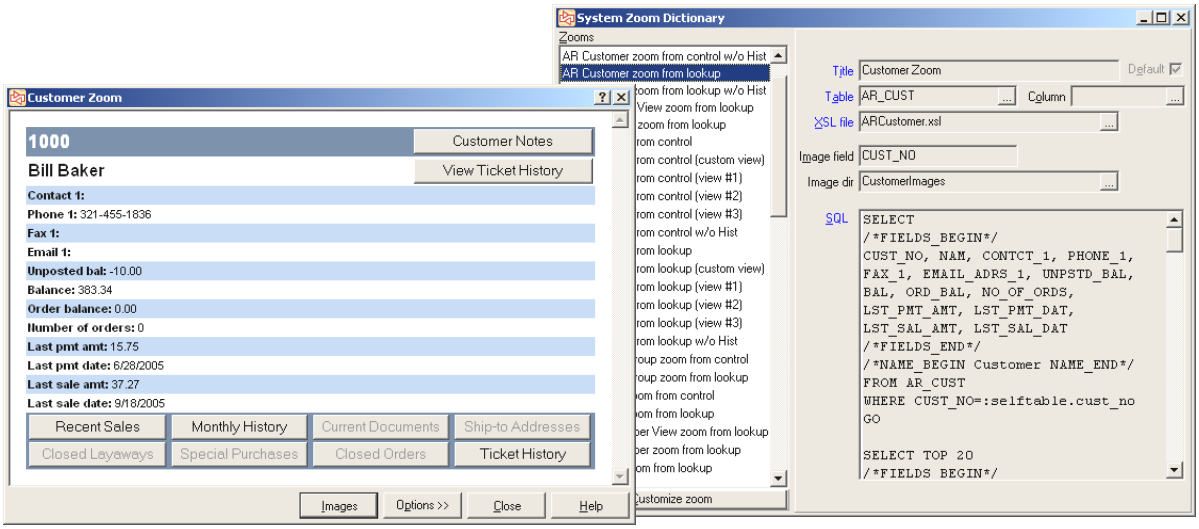
Controls the data in the zoom window.

Every SELECT statement should use a WHERE clause. For a zoom from a field, use ***:Self*** on the right side of the WHERE clause. For a zoom from a lookup, use ***:Selftable.field_ID*** instead.

List the fields that you want to display in the zoom between the comments ***/*FIELDS_BEGIN*/*** and ***/*FIELDS_END*/***. By using the comments, simple customizations of the zoom will also be permitted, and limited to the fields in the list. If you enter an asterisk between the comments (***/*FIELDS_BEGIN*/ * /*FIELDS_END*/***), all fields will initially display in the zoom window and you can then use simple customization to limit which fields to display.

To include an image button in the zoom, include the field identified for “Image field” in the first SQL query, either before or after the ***/*FIELDS_BEGIN*/*** and ***/*FIELDS_END*/*** comments (e.g., ***/*FIELDS_BEGIN*/ * /*FIELDS_END*/, CUST_NO***).

Labels on buttons at the bottom of the zoom window will automatically display the Data Dictionary value for “Description” for the first table named in the SQL query. To use a different label (such as “Recent Sales”), enter the desired label between ***/*NAME_BEGIN*** and ***NAME_END*/***.

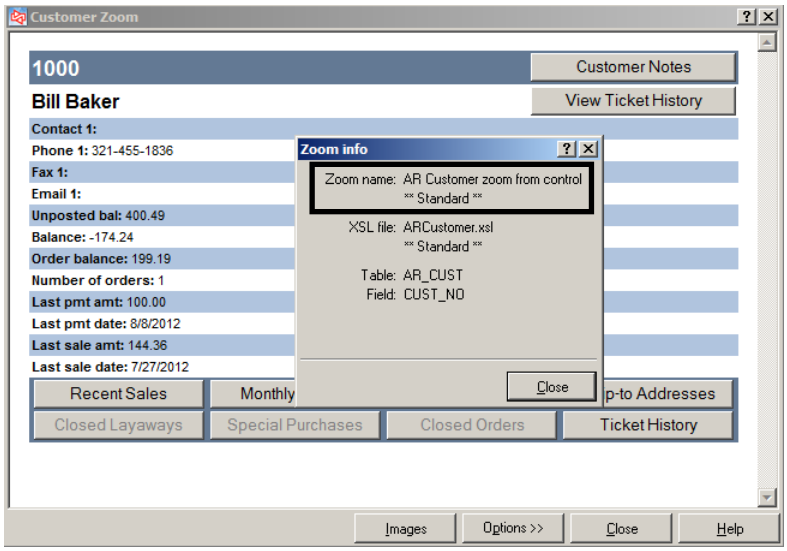


Try It Yourself!

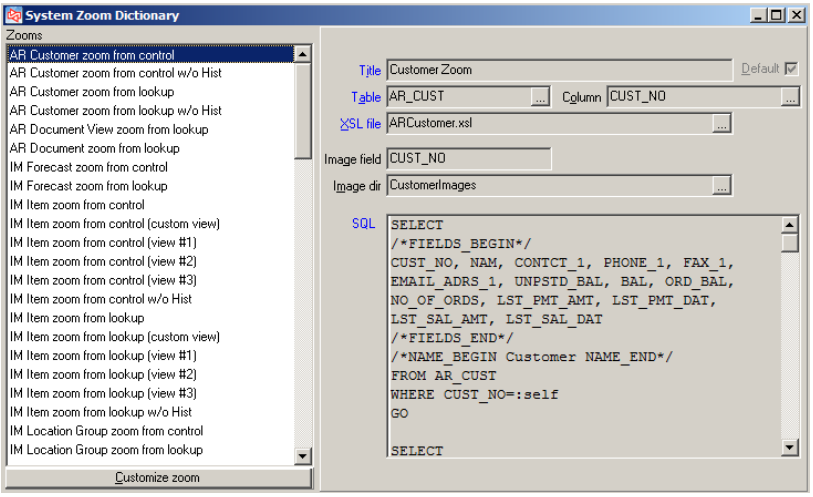
In this exercise, you will customize the Customer lookup window to add the tracking number to the list of orders that appear for a customer. You will also switch the "Closed Layaways " button to a "Tracking Numbers" button.

1. In Customers/Customers, display a customer and click to zoom on the customer.

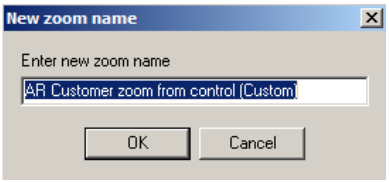
   Right-click the title bar of the zoom window, select "Zoom information" and note the name of the zoom.



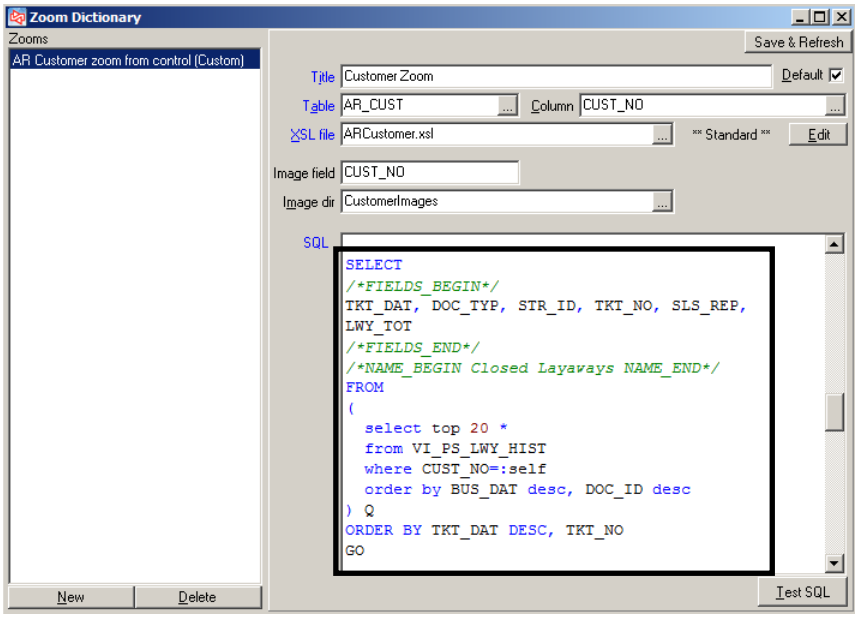
2. Go to **Setup>System>Configuration>Zoom Dictionary**. Find the customer zoom in the list and click Customize zoom to customize it.



Enter a name for the custom zoom.



3. In the Zoom Dictionary window, locate the Closed Layaways section in the SQL area.

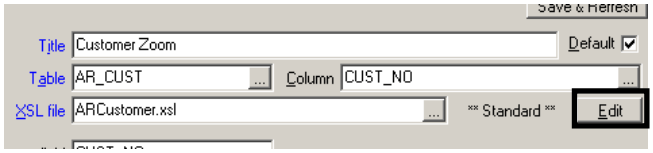


Replace the query with this statement:

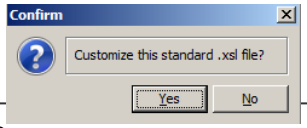
```
SELECT
/*FIELDS_BEGIN*/
H.BUS_DAT, H.DOC_TYP, H.STR_ID, H.TKT_NO, H.SLS_REP, T.PKG_TRK_NO
/*FIELDS_END*/
/*NAME_BEGIN Tracking Numbers NAME_END*/
FROM
   VI_PS_TKT_HIST  H
   join VI_PS_TKT_HIST_PKG_TRK_NO  T
   on T.DOC_ID=H.DOC_ID
WHERE CUST_NO=:self
ORDER BY H.BUS_DAT DESC, H.TKT_NO
GO
```

Click Save & Refresh.

4. Click Edit to edit the ARCustomer.xsl file. (You may need to change the file's properties so that it is not read-only.)

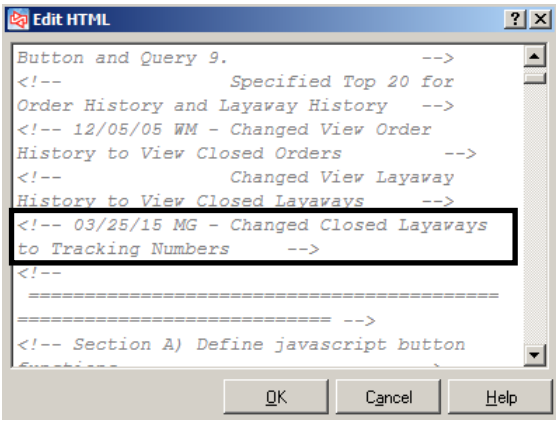


Answer Yes to customize the file.

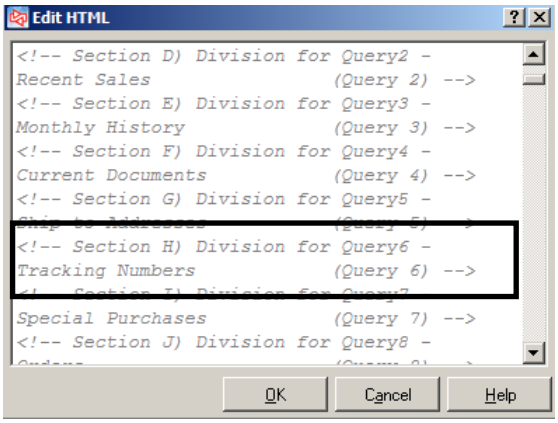


Begin by scrolling down to the bottom of the comments.

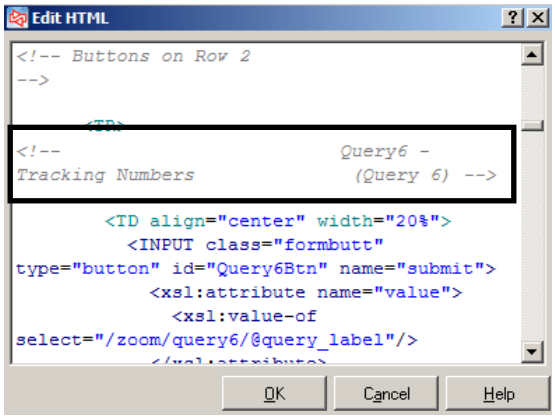
Add a comment about the change you are making.



Scroll further down to the comment for Section H and change the section name from Layaways to Tracking Numbers.

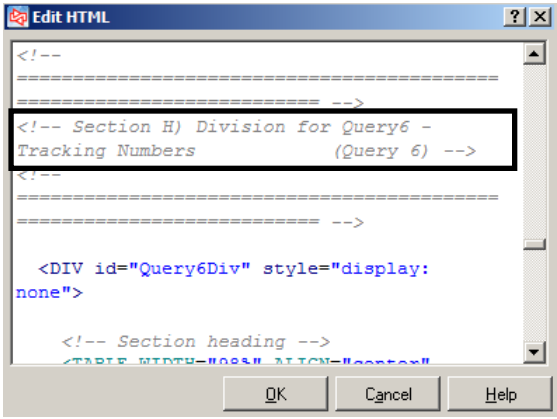


Scroll down again to the comment for Query6 and change the comment from Layaways to Tracking Numbers.



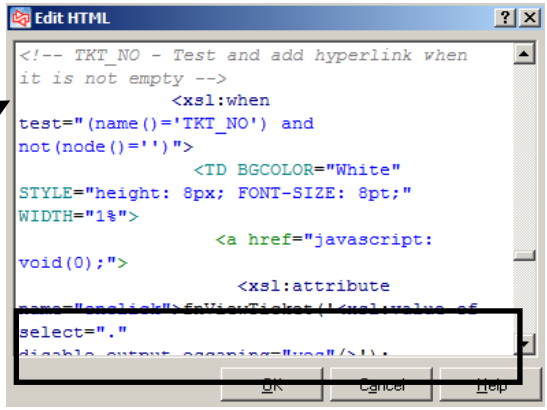
Scroll down to the comment for Section H) Division for Query 6.

Change the comment from Layaways to Tracking Numbers.



The above four changes are cosmetic and do not change the zoom function.

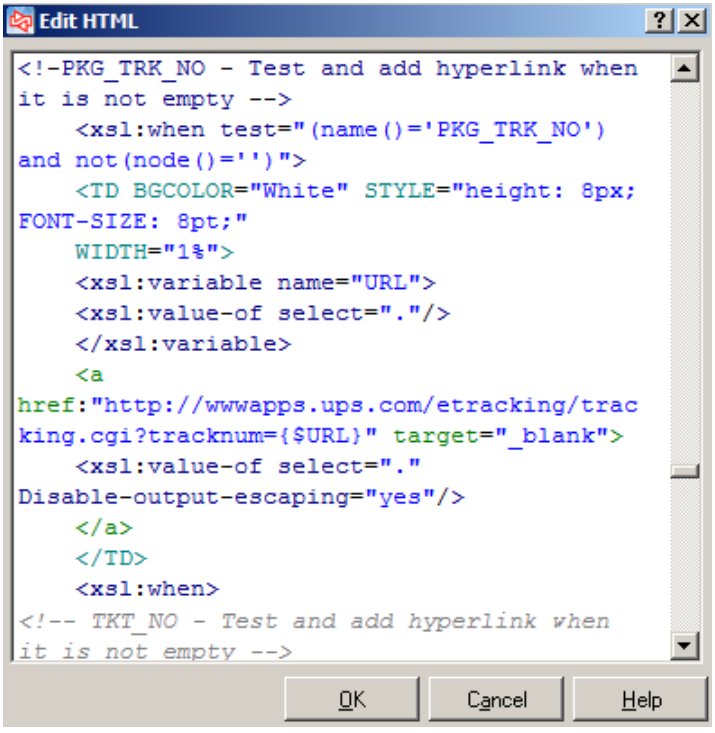
Under the Section H query, locate the "<!--TKT_NO – Test and add hyperlink…" comment.



Following "At <xsl:attribute name="onclick">", change **fnViewLayaway** to **fnViewTicket**.

Insert this query immediately above the "<!--TKT_NO – Test and add hyperlink…" comment.

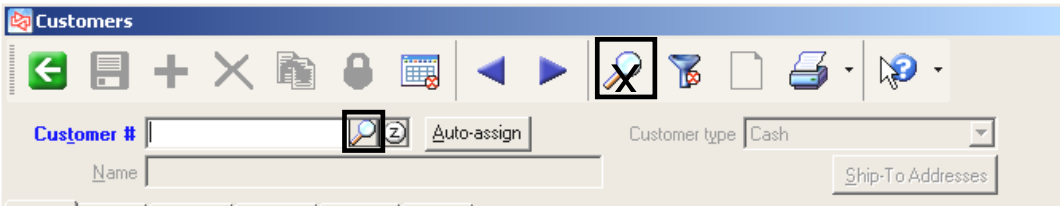
```
<!--PKG_TRK_NO – Test and add hyperlink when it is not empty -->
<xsl:when test="(name()='PKG_TRK_NO') and not(node()='')">
<TD BGCOLOR="White" STYLE="height: 8px; FONT-SIZE: 8pt;"
WIDTH="1%">
<xsl:variable name="URL">
<xsl:value-of select="."/>
</xsl:variable>
<a href:"http://wwwapps.ups.com/etracking/tracking.cgi?tracknum={$URL}"
target="_blank">
<xsl:value-of select="." disable-output-escaping="yes"/>
</a>
</TD>
</xsl:when>
```



5. Click OK to close the Edit HTML window and then click Save & Refresh.

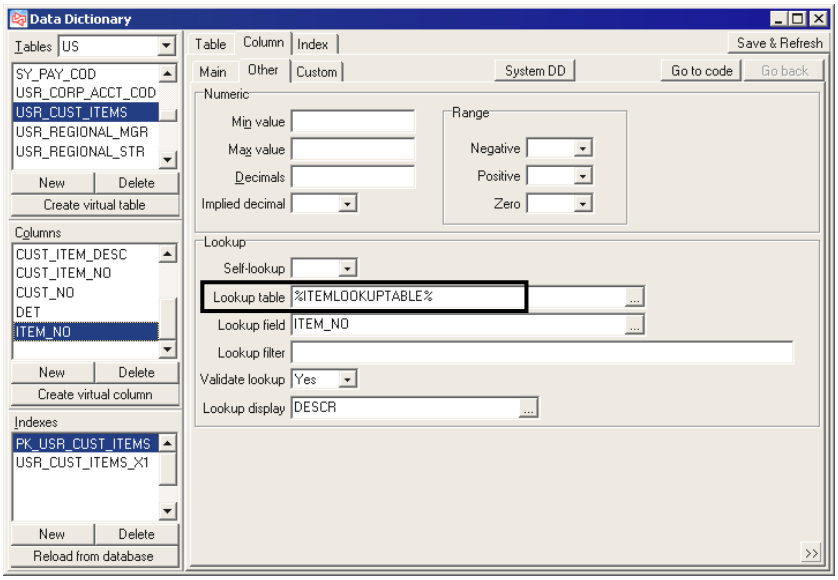
Data Dictionary Settings for Lookup table

Only for lookups from a control, not from toolbar lookup



Lookup table – Table or view from which lookup data is drawn

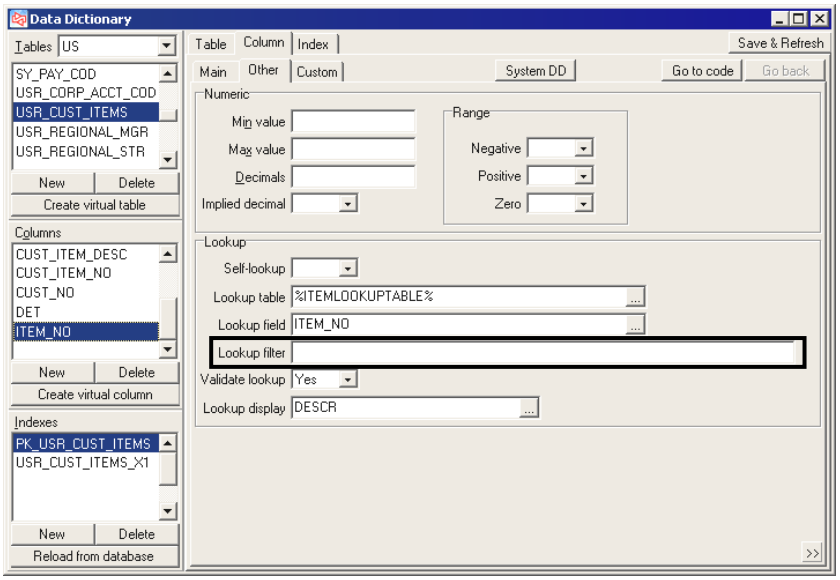
%ITEMLOOKUPTABLE% = table/view specified for "Item lookup shows" in **Setup / System / Workgroups**



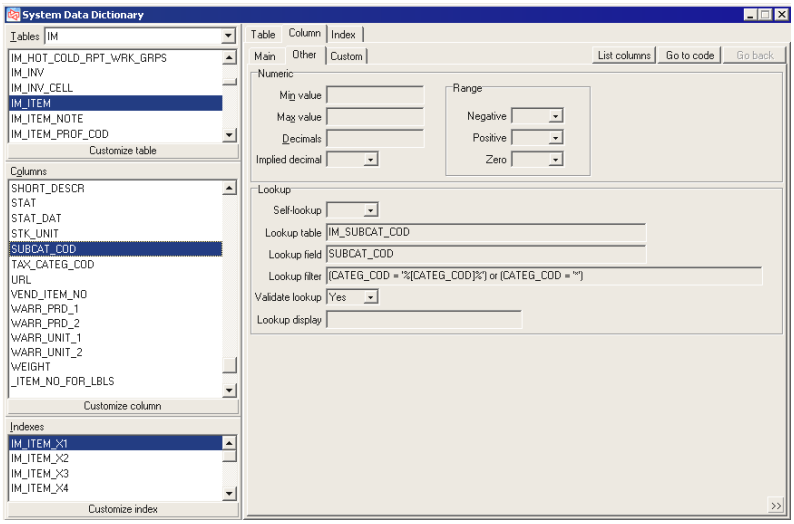
Using Macros in Lookups

Data Dictionary Settings for Lookup filter

Lookup filter – use to limit data returned in lookup



Dynamic filter: use %[FIELDNAME]% to limit records presented in lookup, based on value of a different field in same table



Example: (CATEG_COD = '%[CATEG_COD]%') or (CATEG_COD = '*')

When defined for sub-category column in IM_ITEM table, restricts sub-category lookup to show only sub-categories valid for the current category or for all categories.

Works in Data Dictionary lookup filters & filter saved with lookup (V8.3.8)

More examples of Dynamic Filters

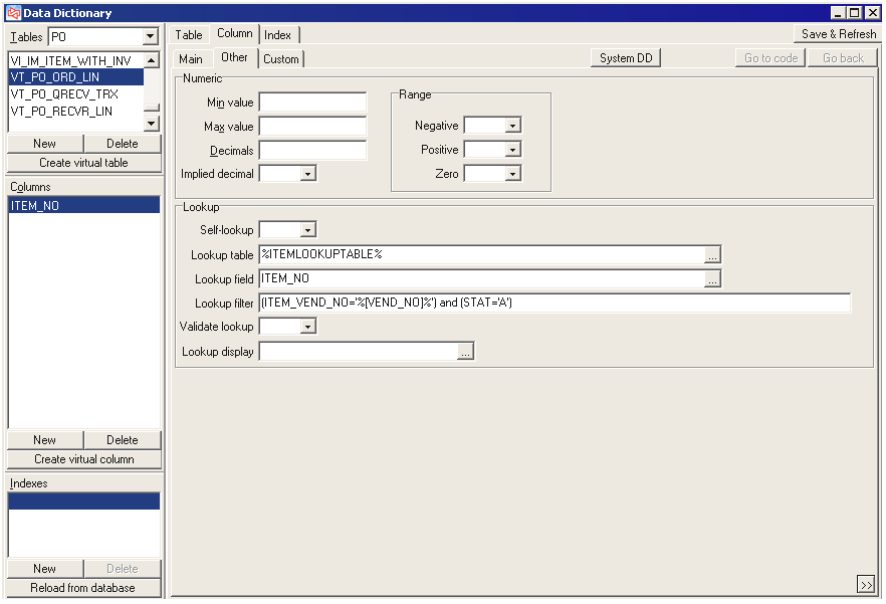
What does this filter do?

_____

_____

_____

_____



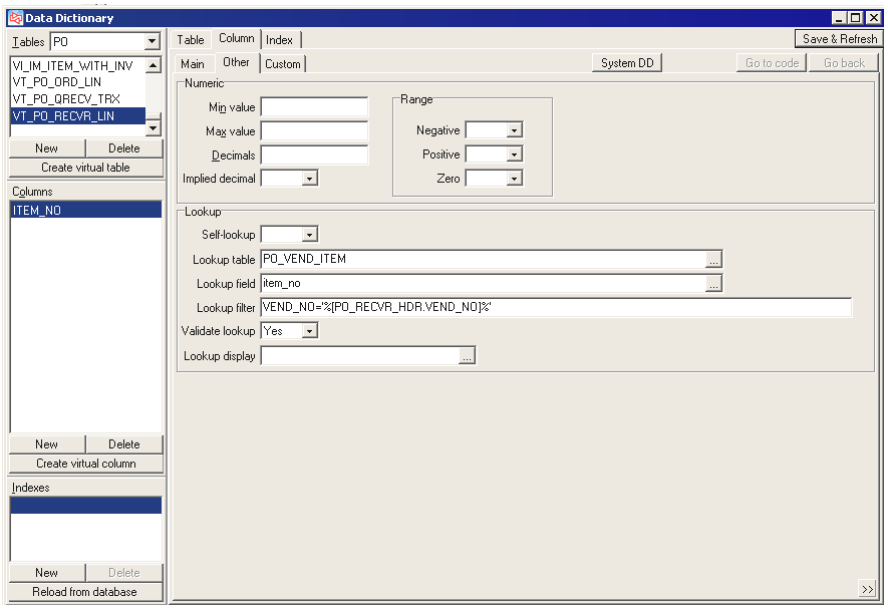
What does this filter do?

_____

_____

_____

_____



Hint: Use %[TABLENAME.FIELDNAME]% to limit records in lookup, based on value of a field in a different table

Using Macros in Lookups

Data Dictionary Settings for Lookup filter

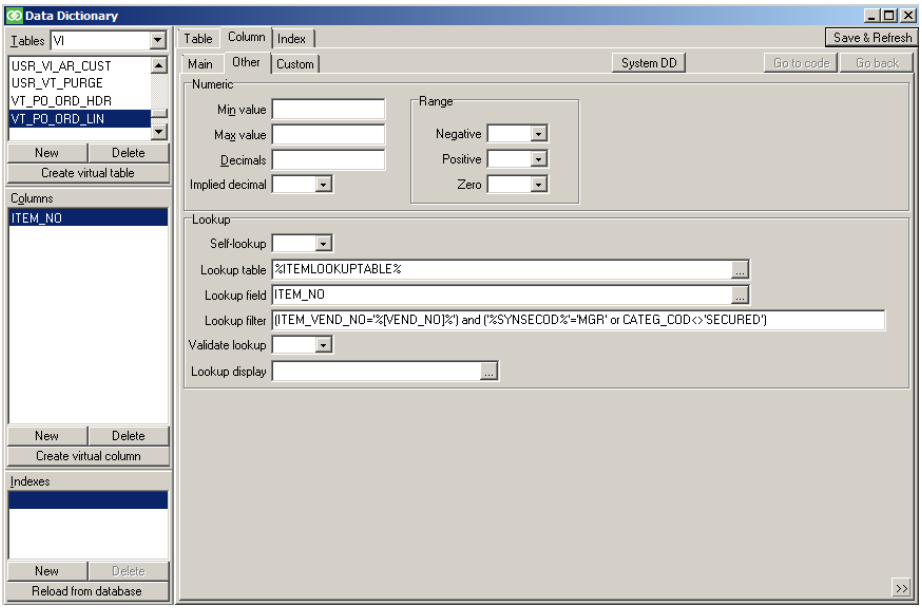
Filter macros:

- use these special macros to limit records presented in lookup

%SYNUSER%	User logged into Counterpoint (or Touchscreen)
%SYNSECOD%	System security code of active user
%SYNWKGRP%	Workgroup active user logged into
%SYNLOC%	"Current" location (varies by form)
%SYNSTORE%	"Current" store (varies by form)

- Allows a single lookup filter to dynamically change, based on location, workgroup, user, etc.
- Can be used in filters saved with lookups, as well as data dictionary filters
- Cannot be used in Point of Sale tables

Example: Lookup table = %ITEMLOOKUPTABLE%
Lookup field = ITEM_NO
Lookup filter = (ITEM_VEND_NO='%(VEND_NO)%') and
('%SYNSECOD%' = 'MGR' or
CATEG_COD <> 'SECURED')



When defined for VT_PO_ORD_LIN.ITEM_NO, only users with a system security code of 'MGR' are allowed to purchase items in the 'SECURED' category from the item's primary vendor.

More examples of Filter macros

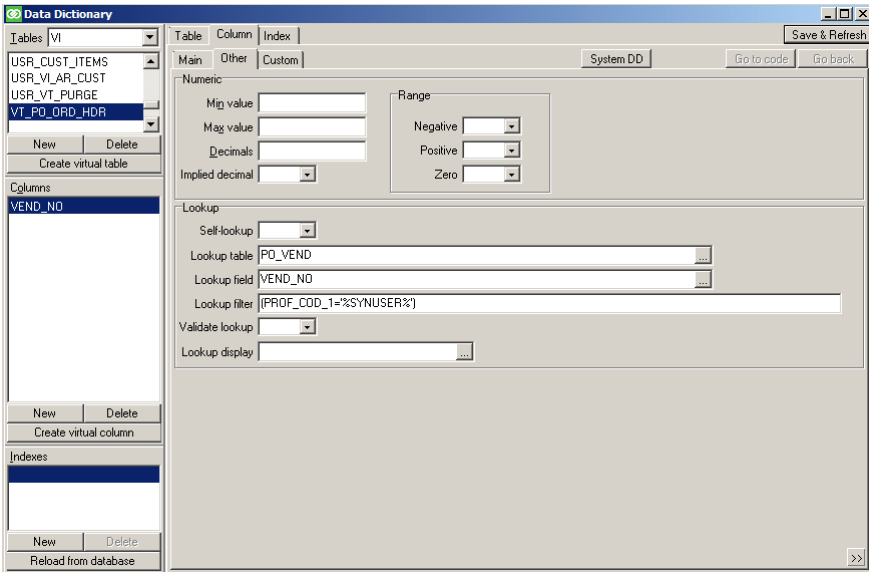
What does this filter do?

_____

_____

_____

_____



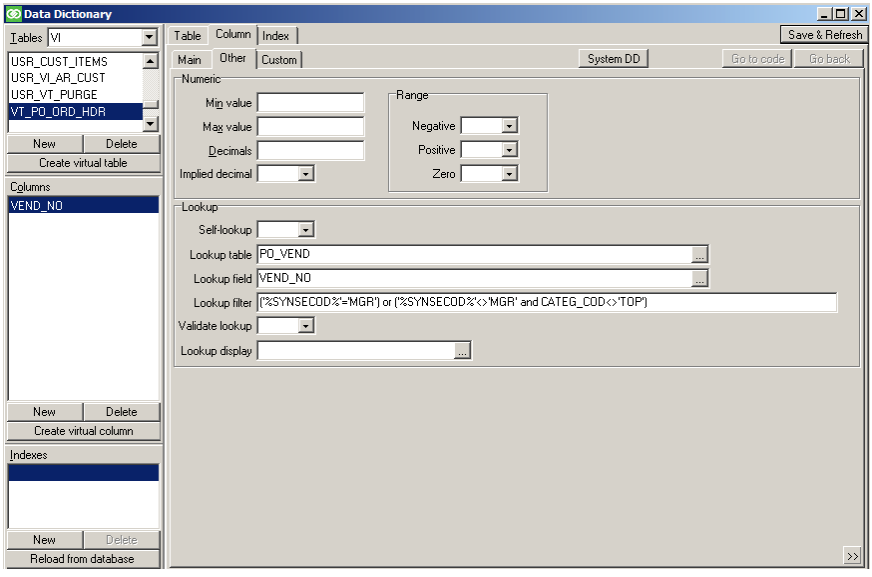
What does this filter do?

_____

_____

_____

_____

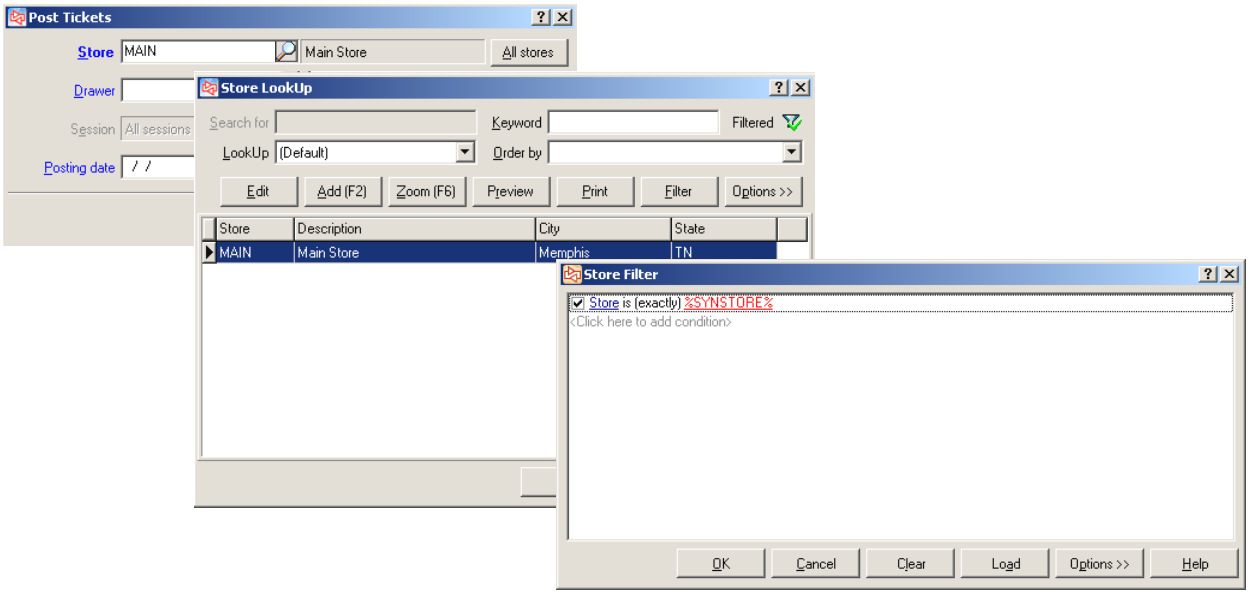


Using Lookup Filters

Lookup Filters using Macros and Dynamic Filters

Two places to define Lookup Filters: 1) Data Dictionary Lookup info
2) Saved within a lookup

	Data Dictionary Lookup Filter	Saved filter within Lookup
Determines which records display	Yes	Yes
User can edit or remove the filter on-the-fly to display other records	No	Yes
User can directly type a value to access a record that's not already displayed	No	Yes
Dynamic filter support (%FIELDNAME%)	Yes	Yes (V8.3.8)
Macro support (%SYNxxxxx%)	Yes	Yes



What is a Stored Procedure?

- Group of SQL statements that resides in the database and that executes with a single call to the server
- Executes before or after a standard Counterpoint stored procedure
- Use for complex and/or repetitive business rules

What is a Trigger?

- Special type of stored procedure that resides in the database and is attached to a table or view
- Executes when a record insert, update, or delete occurs in the table to which it is attached
- Use to enforce "referential integrity" (related data remains valid across multiple tables)

Custom Stored Procedures and Triggers

When to use Triggers

- To perform a certain action as a result of an insert, update, or delete

  Examples: Determine whether to allow an order to be added, based on the customer's account status.

  Send an e-mail message to a specified person when a new order is added.

- To cause updates to be done automatically to matching records in other tables

  Example: When a customer record is deleted, automatically delete that customer's customer-item number records.

- Instead of a column constraint, when restriction is based on column in a different table

  Example: When an employee job level is added or updated in the Job Levels table, check that the job level is within the salary range defined for the job. Salary ranges are stored in the Jobs table.

Browse to ***Adv Customizing/Example Projects*** in the CPSQLTrainCD folder on the thumb drive for several examples of applications that use triggers and stored procedures.

Vendor Item Catalog

Description: A vendor supplies you with a text file that contains a full catalog of his merchandise. You only want to add one of the vendor's item to your Items table in Counterpoint if a customer actually wants to buy or order the item.

Method:

1) Add the Vendor Catalog Items table to the database and create data dictionary entries for it. Add a virtual column to the table, an "Add item" checkbox.

2) Import all of the vendor's catalog items into the new table, including prices and descriptions. Do this before adding the trigger to the table (next step).

3) Add a trigger to the table such that when "Add item" is enabled in a vendor catalog item record, the trigger fires to add the item to the Items table, produce an inventory record for it, and return to the ticket when completed.

4) Add a custom maintenance menu selection to permit maintenance of the data in the new table.

5) Add a custom button action in Actions.xml that will execute the new menu selection.

6) Add a Custom Action button in Touchscreen, where the custom action is the name of the menu action from Actions.xml.

In Touchscreen ticket entry, when the Custom Action button is selected, it runs the Vendor Catalog Items maintenance function and allows look up into the Vendor Catalog Items table.

When you:
- select an item from the lookup,
- enable the "Add item" checkbox, and
- save the record,

it fires the trigger that:
- copies the item information into the Items table,
- adds an inventory record for the item, and
- returns to ticket entry with the item ready to sell or order.

Custom Stored Procedures vs Triggers

Stored Procedures	Triggers
Non-maintenance or non-ticket functions Name as USER_BEFORE_standardSPname or USER_AFTER_standardSPname Maintenance and ticket functions (see next page)	No special naming requirements
Runs from Counterpoint whenever a standard stored procedure is called	Runs from MSSQL when the triggering event occurs on the attached table
Can call other stored procedures ("nesting")	Can fire other triggers when other tables with their own triggers are updated Can call stored procedure (regardless of name)
Can accept parameters as input to the SQL statements or produce parameters as output	No parameter passing
Knows what caused it to run only through the parameters passed to it	Knows what specific event caused it to run, through the inserted/deleted table
Will not run if applications other than Counterpoint call the standard stored procedure	Will run if applications other than CP insert, update, or delete from the table
Disabled via "Enable customizations" in Setup / System / User Preferences	Not disabled via "Enabled customizations". Use SQL "Alter table" command to disable. Often disabled before batch transactions and enabled after batch transaction complete, to improve database performance
	Cannot create triggers on view tables

For maintenance forms when a record is saved or deleted

USER_<tablename>_BEGIN_TRANSACTION

- When record is about to be saved or deleted
- Old data is still present, except for inserts
- Custom updates are rolled back if transaction is rolled back

USER_<tablename>_BEFORE_COMMIT_TRANSACTION

- When SQL transaction is about to be committed
- New data is present, except for deletes
- Custom updates are rolled back if transaction is rolled back

USER_<tablename>_AFTER_COMMIT_TRANSACTION

- Doesn't execute if errors occurred that resulted in transaction rollback
- All updates have been committed
- Too late to rollback any updates
- Typically used to display message

For Ticket Entry/Touchscreen
USER_AFTER_PS_TE_SAVE_DOC
USER_AFTER_PS_TE_VOID_DOC
USER_PS_TE_BEFORE_PRINT_PS_DOC
USER_PS_TE_AFTER_PRINT_PS_DOC
USER_BEFORE_PS_TE_COMMIT_DOC
USER_AFTER_PS_TE_COMMIT_DOC
USER_BEFORE_PS_TE_UNCOMMIT_DOC
USER_AFTER_PS_TE_UNCOMMIT_DOC

For Ticket Posting
USER_BEFORE_PS_DOC_POST_RUN
USER_PS_DOC_POST_TRX
USER_AFTER_PS_DOC_POST_RUN

NOTE: Use DOCID in parameter passing for all ticket-related Stored Procedures

Custom Add-on-the-fly Forms

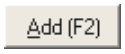
Build Custom Add-on-the-fly Forms to:

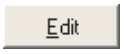
- Request different fields than those on the normal* add-on-the-fly forms

  For fields that don't appear, uses default values from

  1) template item or customer
  2) Data Dictionary ("Default" value for column)
  3) Database (Default value set with "constraint")

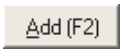
- Provide add-on-the-fly capability for a custom table

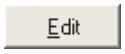
  To **enable** 

and 

in table lookup

- Prevent add-on-the-fly capability for a specific standard table

  To **disable** 

and 

in table lookup

- Use menu selections to define add-on-the-fly functions

  To restrict add-on-the-fly ability by menu code (user)
  To allow add-on-the-fly, but not edit-on-the-fly

User preference for "Enable customizations" must be enabled for Custom Add-on-the-fly settings to take effect.

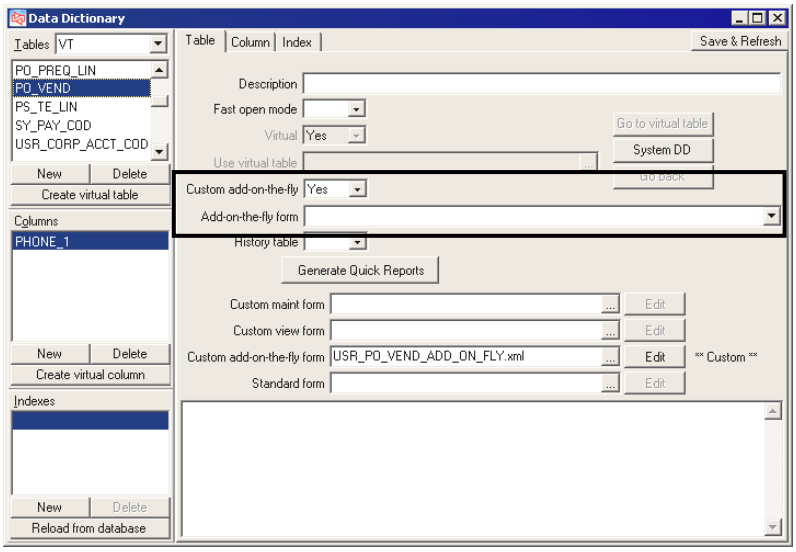
* "Normal" add-on-the-fly form is typically the full maintenance form.
If simplified add-on-the-fly is enabled for either customers or items, the "normal" form is instead the predefined add-on-the-fly form supplied with Counterpoint.

Select **Setup / System / Configuration / Data Dictionary**

From the System Data Dictionary, select the standard table to which the add-on-the-fly records will be added and click Customize table .

For a custom table, select the table from the Tables list in the Data Dictionary, instead of the System Data Dictionary. (You will not have to click anything.)

Answer Yes at **Custom add-on-the-fly**, to indicate that you want to use a custom add-on-the-fly form for this table, instead of the full maintenance form.

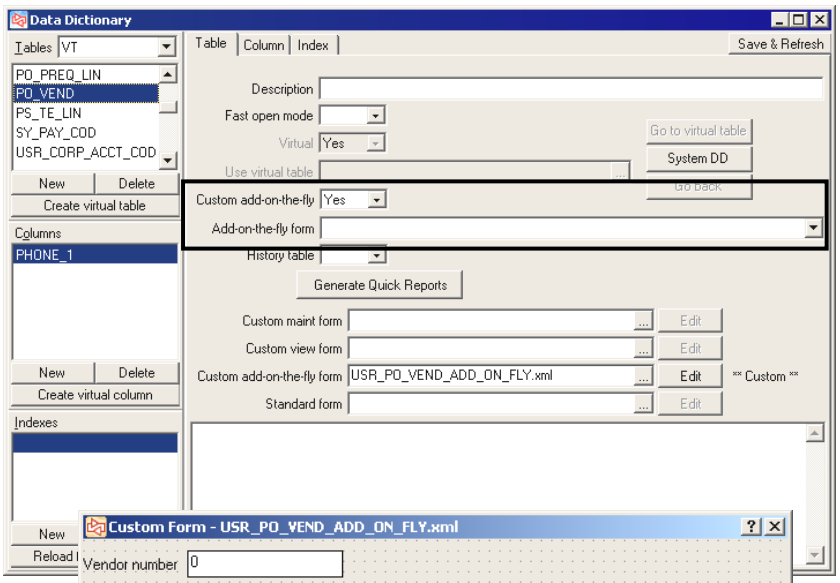


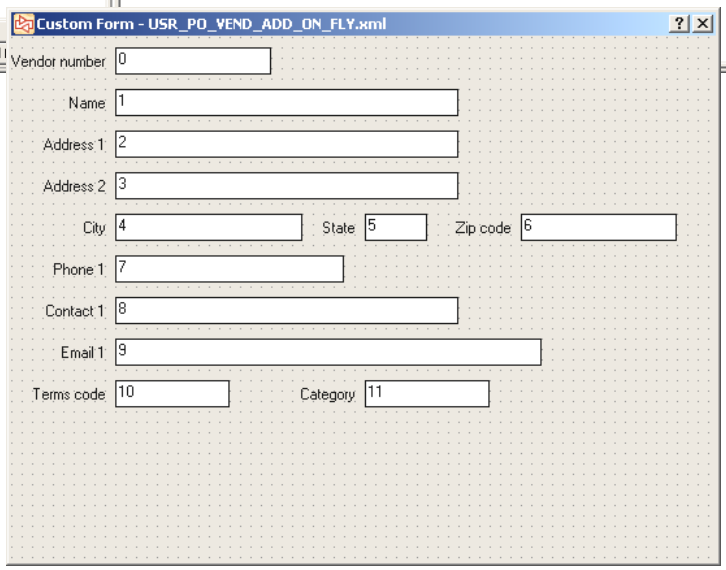
For a standard table, leave **Add-on-the-fly form** blank and define an XML layout at Custom Add-on-the-fly form.

For a custom table, build an “add-on-the-fly” menu selection in a menu code, and then select that menu selection at **Add-on-the-fly form** here in the data dictionary.

Custom add-on-the-fly	Add-on-the-fly form	Description
Y	<Blank>	Define an XML layout with desired fields under **Custom add-on-the-fly form**. Full maintenance function for table must be in user's menu code.
Y	<Menu-ID>	Specify a custom “add on the fly” menu selection. Menu selection must be in user's menu code. This is the only method that works for custom tables.
Y	(Do not allow add-on-the-fly)	Completely disable add-on-the-fly ability for this table. Applies to all users.
N or <Blank>	<Blank>	For standard CP tables, use either full maintenance form or simplified add-on-the-fly form if enabled in Customer Control or Inventory Control function. Full maintenance function for table must be in user's menu code.

On the Table tab, specify a layout file for **Custom add-on-the-fly form** and position the fields that you want displayed when records are added on-the-fly.





It is not necessary to also select or define the columns in the Data Dictionary, unless you wish to use a different setting (display label, for example) for any column.

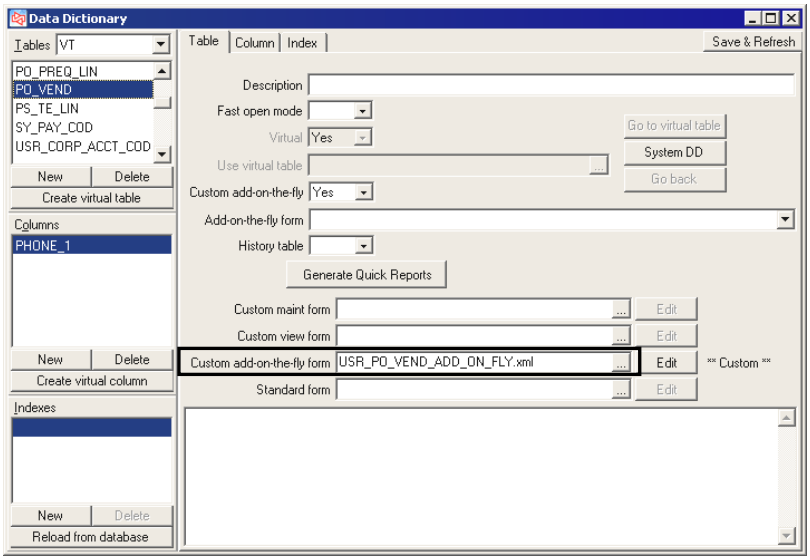
Using Menu Selections for Add-on-the-fly Forms

- Restrict access to add-on-the-fly forms with menu codes
- Provides ability to Allow edit, Allow insert, and Allow delete to records while in add-on-the-fly form
- Must use this method to add add-on-the-fly functionality for a custom table

Build the menu selection

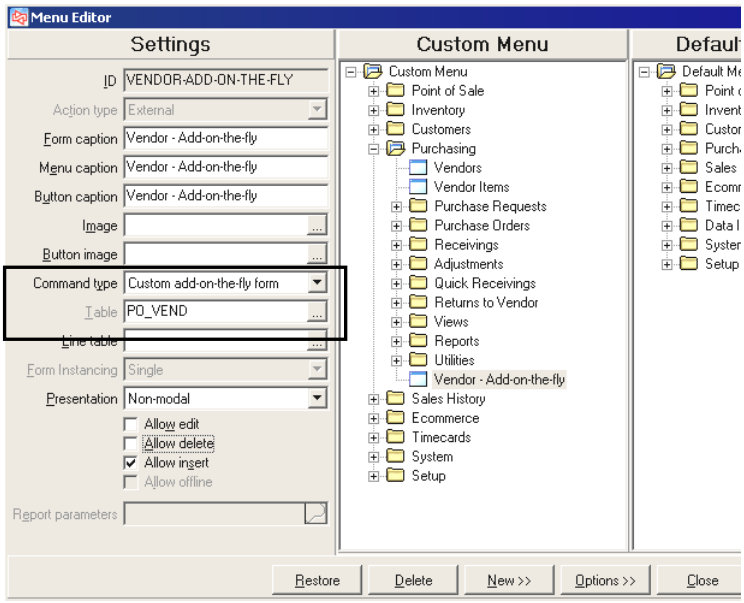
1. In the Data Dictionary, answer Yes to **Custom add-on-the-fly**.

   Create or update the **Custom add-on-the-fly form** for the table in the data dictionary, and position each column that you want to include in the add-on-the-fly form.

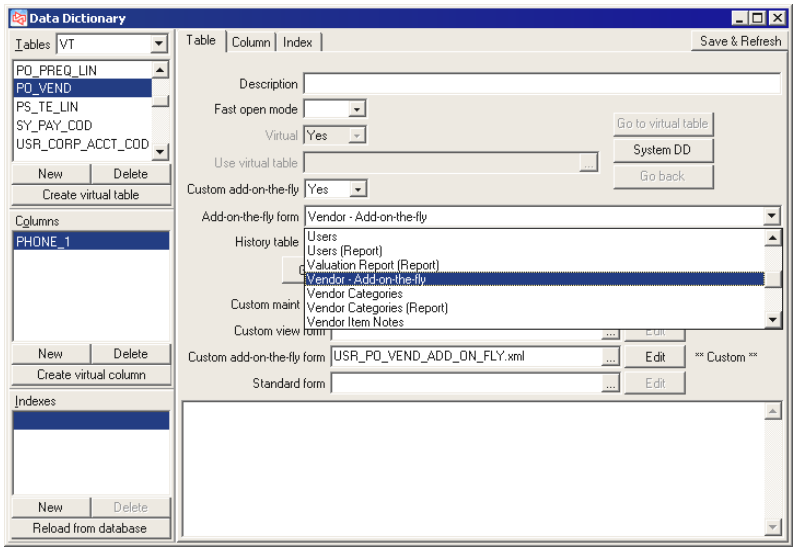


2. Use **Setup / System / Menu Codes** to add a menu item, where **Command type** is "Custom add-on-the-fly form" and **Table** is the table name.

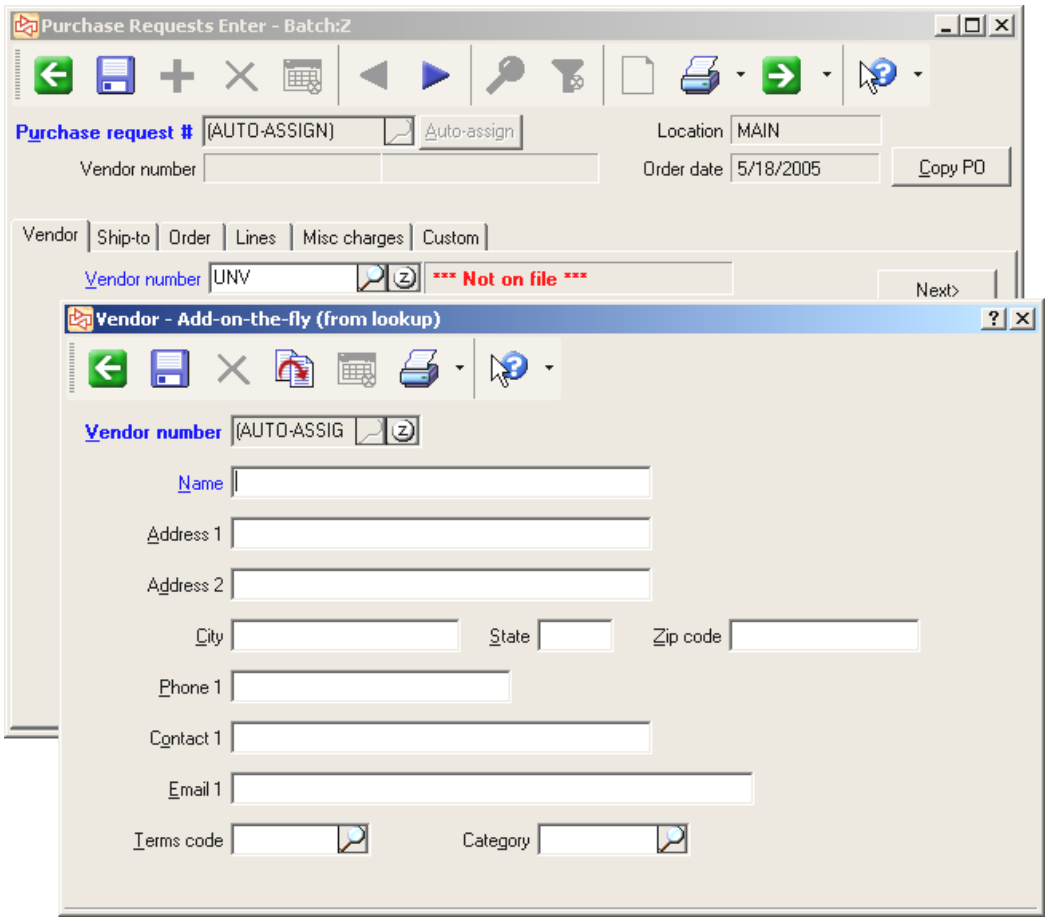
   You can also select whether to allow changes and/or deletions to existing records through this menu selection.



3. Return to the data dictionary, and on the Table tab, choose the new menu selection at **Add-on-the-fly form**.



In this example, when a new vendor is added "on the fly", the custom add-on-the-fly form automatically displays.



Try it Yourself!

Create an add-on-the-fly form for the Corporate Account Code table that allows users to add and edit records in that table "on the fly" from the lookup at the "Corp Account Code" column in **Customers / Customers**.

Bonus exercise 1:

Limit the add-on-the-fly form for the Corporate Account Code table so that users can only add new corporate account codes on the fly, without editing or deleting existing codes.

Bonus exercise 2:

Add the column "Store" to the predefined add-on-the-fly form for customers supplied with Counterpoint.

Solutions are provided in Appendix 1.

Running Custom Programs

What is a Custom Program?

An application, program, or stored procedure that you want to run from the Counterpoint menu

Custom programs can also be run from custom buttons added to toolbars in other menu selections.

Requirements to run a Custom Program

- Custom program must be able to:
  - accept arguments in its command line. Arguments will be passed to the program via a text file (in .ini format) that is created by Counterpoint when the custom menu selection is run.
  - establish its own connection to the company database in Counterpoint

  Custom stored procedure must be able to:
  - accept input parameters named "@OrderBy", "@<Table>_Filter", and "@PARAM_<Table>_<Field>"

- "Custom program" menu item that when run, optionally presents a parameter entry screen to the user to select an order-by and filter the records for the function. User's entries are passed to the custom program or stored procedure.

- Configuration file that defines the parameters, filters, and order-by choices to request from user

Creating the Configuration File

- Create using any text editor
- Save in **Actions** folder for the company, below the company's Configuration folder on the server (e.g., G:\Program Files\RadiantSystems\CounterPoint\CPSQL.1\TopLevel\DemoGolf\Configuration\Actions)

Format of Configuration File

Place each command on a separate line, in any order. Settings for each command can be designated with a colon (**:**) or an equal sign (**=**). Begin any comment lines with ***//***.

Program:ProgramName.exe or **StoredProcedureName** (required)

One Program command must be in the configuration file. Include a pathname if the program is not in the workstation's PATH.

RunMode: SP

If Program setting is a stored procedure, RunMode: SP is required so that the program name is interpreted as a stored procedure in the database.

WaitForCompletion: Yes

To minimize (and suspend operation of) Counterpoint until the custom program finishes running. Counterpoint always waits for a stored procedure.

AutoRun: Yes

Automatically executes custom program or stored procedure without clicking the Run button.

AutoClose: Yes

To automatically close the custom program parameter form after you program or stored procedure finishes.

ShowFilters:Table1, Table2, …

To allow the user to enter filters for the named tables which will be passed to the program. For each table that you name, a separate filter tab will appear for the user. For example, **ShowFilters:IM_ITEM, AR_CUST** will display filter tabs for the Item and Customer tables.

To control which fields appear on each filter tab, customize and save filters for each table in the usual way.

Multiple ShowFilters commands can exist in the configuration file.

ShowOrderBy:Table or
ShowOrderBy:Table.Field Direction, Table.Field Direction, etc.

To allow the user to select an "order-by" for the named tables and fields, including the sort direction, which will be passed to the program.

If only a single table is named, all fields defined in the indexes for the table (in the data dictionary) will appear as "order by" choices.

If specific fields in a table are named, follow the field name with either ASC or DESC to indicate whether to display the values in ascending or descending order. For example, to allow the user to only sort by the item category in either ascending or descending order, include these two commands:

```
ShowOrderBy:IM_ITEM.CATEG_COD ASC
ShowOrderBy:IM_ITEM.CATEG_COD DESC
```

Multiple ShowOrderBy commands can exist in the configuration file.

BeforeOrderBy:Table.Field Direction, Table.Field Direction

Identifies a fixed field and sort direction (ASC or DESC) that is inserted *before* the user's choice of order-by. For example, if a program always totals by item category first and the user can select to show the items within a category in order by item number, description, or Price-1, include these commands in the configuration file:

```
BeforeOrderBy:IM_ITEM.CATEG_COD ASC
ShowOrderBy:IM_ITEM.ITEM_NO ASC, IM_ITEM.DESCR ASC,
IM_ITEM.PRC_1 DESC
```

Only one BeforeOrderBy command can exist in the configuration file.

AfterOrderBy:Table.Field Direction, Table.Field Direction

Identifies a fixed field and sort direction (ASC or DESC) that is appended *after* the user's choice of order-by.

Only one AfterOrderBy command can exist in the configuration file.

Parameter:Table.Field

Displays a Parameters tab when the menu selection is run and requests entry of a value for the specified data dictionary field. The field must be defined in data dictionary (but doesn't have to exist in SQL database). "Domain" setting determines the type of entry field. If marked as "Required" in data dictionary, user must make an entry before program is run.

Multiple Parameter commands can exist in the configuration file, where each command results in another requested field on the Parameters tab.

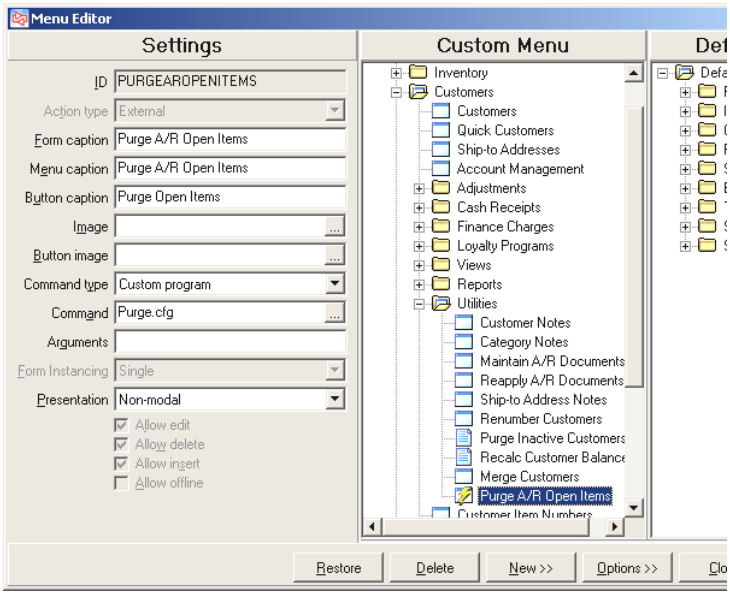
Running Custom Programs

Creating the Menu Item

Use **Setup / System / Menu Codes**, display an existing code or add a new code, and select to add a new menu item.

At **Command type**, select “Custom program”.

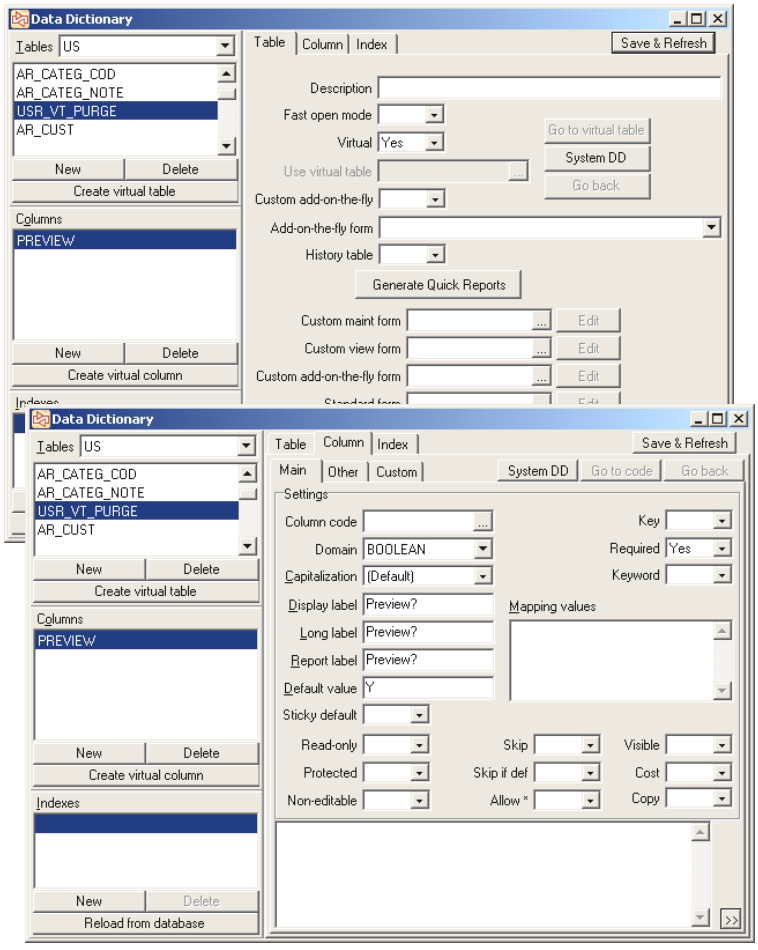
At **Command**, enter or browse to the configuration file name.



Creating the Parameter Field

If your configuration file specifies a Parameter setting, use **Setup / System / Configuration / Data Dictionary** to create the virtual table and virtual column named in the Parameter setting.

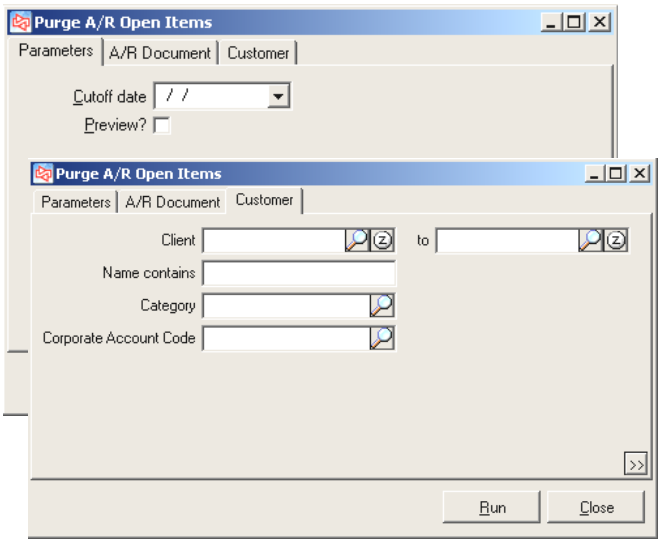
The Domain value of the column determines the type of entry field requested (checkbox, number, date, etc.) when the custom program menu selection is run.



Running the Custom Program

When the new menu selection is run, it displays a screen that contains:

- an order-by selection list that contains each ShowOrderBy command in the configuration file
- a parameter tab if the configuration file contains Parameter commands
- a filter tab for each table if the configuration file contains ShowFilter commands



When you click Run for a custom program, Counterpoint writes a .TMP text file (in .ini format) to the workstation's Local Settings\Temp folder that contains the filter, order-by, and parameters for the custom program. The custom program is then executed. The path and filename of the .TMP file are appended to any arguments that you specified in the menu item and are passed to the custom program.

When you click Run for a custom stored procedure, the parameters are passed directly to the stored procedure rather than passing them through a .TMP text file.

Try it Yourself!

This exercise shows what you will need to do to run a custom program that purges paid A/R documents. You will learn how to:

- build the configuration file,
- use the data dictionary editor to add a virtual table with two virtual columns which you can use to request a parameter for your program,
- add a menu item to run the custom program, and
- customize the filter for the new menu selection.

1. Build the configuration file for the custom program.

   Open Notepad (or some other text editor) and enter the following information:

```
ShowFilters:AR_OPN_ITEM,AR_CUST
Program:PurgeARPaidDoc.exe
Parameter:USR_VT_PURGE.CUTOFF
Parameter:USR_VT_PURGE.PREVIEW
WaitForCompletion:Yes
```

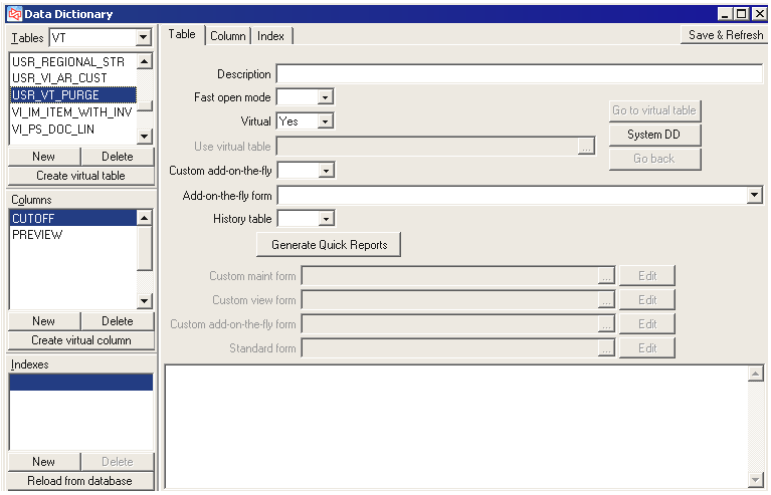
   For purposes of class, since the program doesn't really exist, substitute **Notepad** for **PurgeARPaidDoc.exe** in the Program command.

   Save the configuration file in the Actions folder for the company you are using (either TestGolf or CampgolfTraining), using the name **Purge.cfg**.

2. Use **Setup / System / Configuration / Data Dictionary** and click 

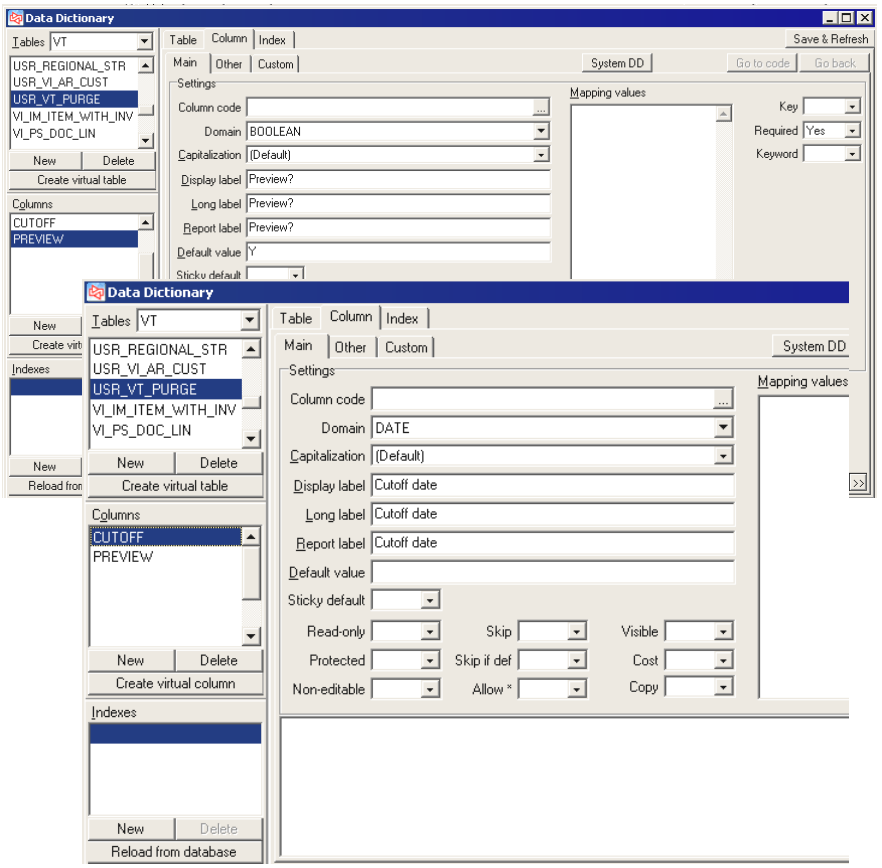
   Name the table **USR_VT_PURGE**.

   Use the same table name that you specified for the Parameter setting in the configuration file.



Click Create virtual column and add a column named **PREVIEW**.

In addition to entering a **Display label**, set **Domain** to “Boolean”, and set **Default value** to “Y”.



Add another virtual column named **CUTOFF**. Enter a display label ("Cutoff date"), and set **Domain** to "Date".

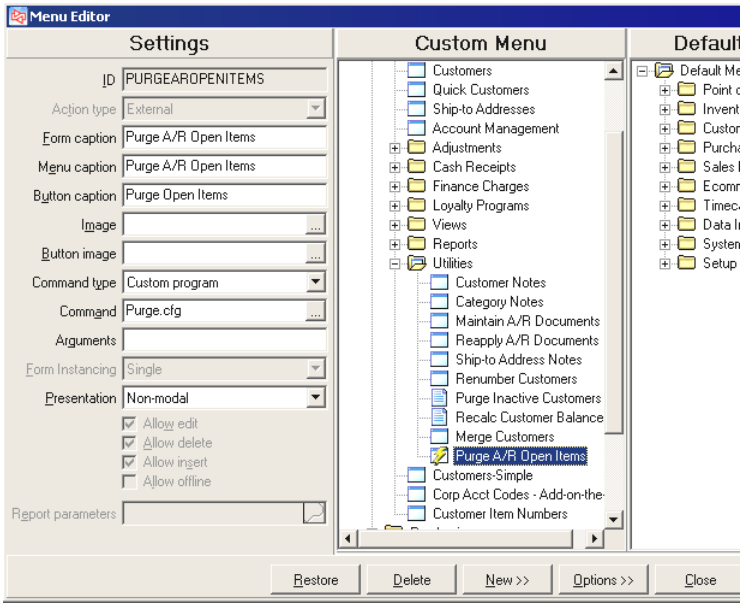
3. Use **Setup / System / Menu Codes** and display the TRAIN menu code.

Add a new menu item under Customers / Utilities.

At **Command type**, select “Custom program”.

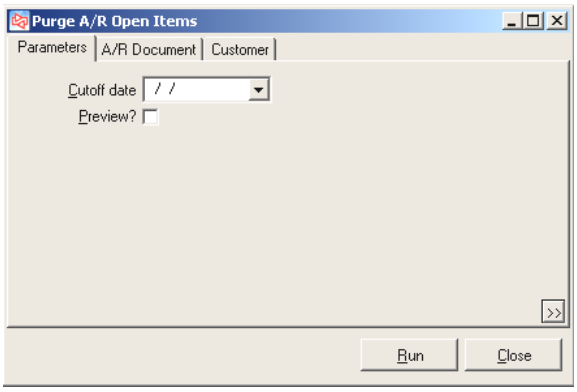
At **Command**, enter the configuration filename “Purge.cfg”.

Click Close and then save the menu code.



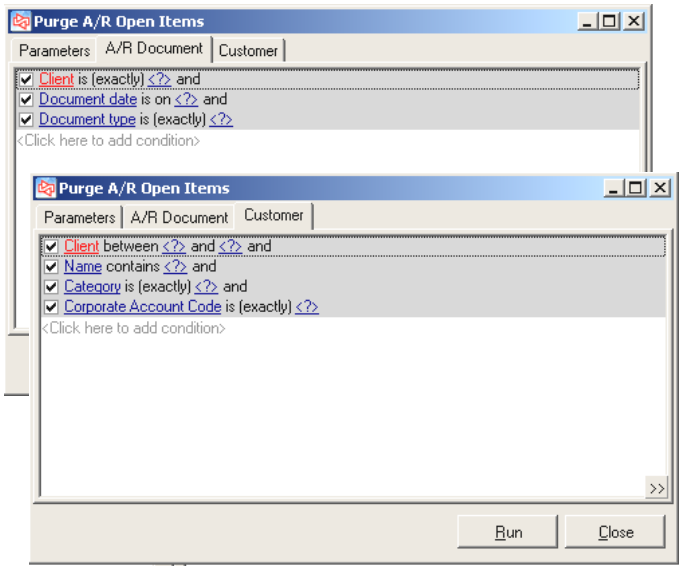
4. Run the new menu selection (**Customers / Utilities / Purge A/R Open Items**).

Notice that the Parameters tab contains the "Cutoff date" and “Preview?” columns.



The custom program needs to be written so that it (a) only purges fully paid items on or before the cutoff date, and (b) displays a list of what will be purged (without purging) if the “Preview?” column is selected.

The filters on the A/R Document and Customer tabs can be customized (and saved) to allow restriction of the documents and/or customers to consider.



Click Run to execute the custom program.

Note: For testing purposes, you can build the configuration file and substitute Notepad as the program. When you click Run , Notepad will open and show the contents of the .TMP file that will be passed to your “real” program.

End of Exercise

What is a Custom Crystal Report?

A user-defined report developed in Crystal Reports that will be run from the Counterpoint menu

What can be done in Counterpoint when running the report?

Select the sequence in which to run the report, provide filtering for the report, and request parameters that are defined in Crystal – all from Counterpoint.

Enter comments in the record selection formula of the Crystal report, using **Report / Selection Formulas / Record** in Crystal

	Comment in record selection formula
➲ Allow selection of an "Order by"	//ShowOrderBy
➲ Allow filtering of one or more tables	//ShowFilters
➲ Request parameters	//PromptForParameters
➲ Change the label on the Print button	//PrintCaption
➲ Change the label on the Preview button	//PreviewCaption
➲ Change the label on the Email button	//EmailCaption

➲ Prevent preview, print, email, save, printer setup, refresh or export ability	//AllowPreview
	//AllowPrint
	//AllowEmail
	//AllowSave
	//AllowPrinterSetup
	//AllowRefresh
	//AllowExport

➲ Print report on a 40-column receipt printer	//ReportType:Receipt

"//" in record selection formula = comments ignored by Crystal, but recognized by Counterpoint

Can also use record selection comment fields to customize standard Counterpoint reports

Allow Selection of "Order-by"

- Add **//ShowOrderBy:** comment to record selection formula in main report to identify order-by choices
- If **//ShowOrderBy:** is not included in record selection formula, user is given a choice of all indexes for first table in main report.
- If **//ShowOrderBy:** is added to record selection formula for an existing CPSQL report, also add comments for any existing order-by choices to retain them.
- Can instead add a new index for the table in the Data Dictionary for table-based reports (not for stored procedure-based reports)

//ShowOrderBy:Table

All fields defined as indexes for the table (in the data dictionary) appear as "order by" choices.

//ShowOrderBy:Table.Field Direction, Table.Field Direction, etc. or

Fields from multiple tables are supported as "Order-by" choices. Put each Table.Field entry on separate line, and follow field name with either ASC or DESC to indicate whether to display the values in ascending or descending order. (Default = ASC)

E.g., user can sort by item category in either ascending or descending order. Add both comments in record selection formula:

//ShowOrderBy:IM_ITEM.CATEG_COD ASC
//ShowOrderBy:IM_ITEM.CATEG_COD DESC

//ShowOrderBy: None

Do not allow user to select the "Order-by" for the report. (Sort order should be specified within the report.)

//BeforeOrderBy:Table.Field Direction, Table.Field Direction

Identifies a fixed field and sort direction (ASC or DESC) that is inserted before user's choice of order-by.

For example, if report always totals by item category first and user can select to print items within a category in order by item number, description, or Price-1, include these comments in record selection formula:

```
//BeforeOrderBy:IM_ITEM.CATEG_COD ASC
//ShowOrderBy:IM_ITEM.ITEM_NO ASC, IM_ITEM.DESCR ASC,
IM_ITEM.PRC_1 DESC
```

//AfterOrderBy:Table.Field Direction, Table.Field Direction

Identifies a fixed field and sort direction (ASC or DESC) that is appended after user's choice of order-by.

Example: Your ticket report always sorts by store and station, but user has the option of whether to sort tickets by ticket number or customer number within each store/station. On the same report, line items are shown in entry order. The comments in the record selection formula for this report would look like this:

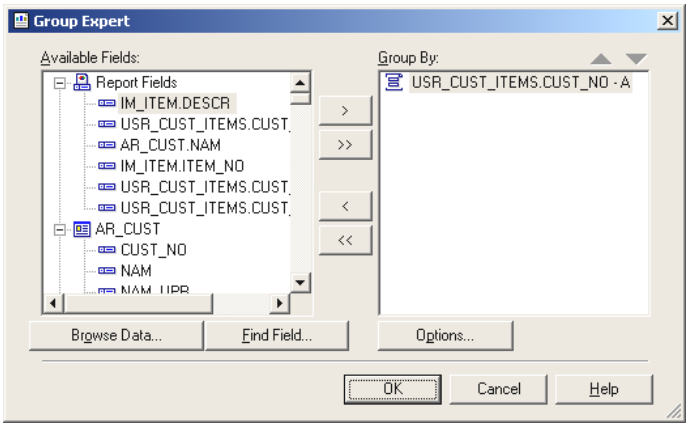
```
//BeforeOrderBy: PS_DOC_HDR.STR_ID, PS_DOC_HDR.STA_ID
//ShowOrderBy: PS_DOC_HDR.TKT_NO ASC,
//ShowOrderBy: PS_DOC_HDR.CUST_NO ASC
//AfterOrderBy: PS_DOC_LIN.SEQ_NO
```

Or even better:

```
//BeforeOrderBy: PS_DOC_HDR.STR_ID, PS_DOC_HDR.STA_ID
//ShowOrderBy: PS_DOC_HDR.TKT_NO ASC,
//ShowOrderBy: PS_DOC_HDR.CUST_NO ASC, PS_DOC_HDR.TKT_NO ASC
//AfterOrderBy: PS_DOC_LIN.SEQ_NO
```

How are “order-by” sorts applied to a report?

1) Sorts defined in Crystal Reports using **Record Sort Expert** or **Group Sort Expert**

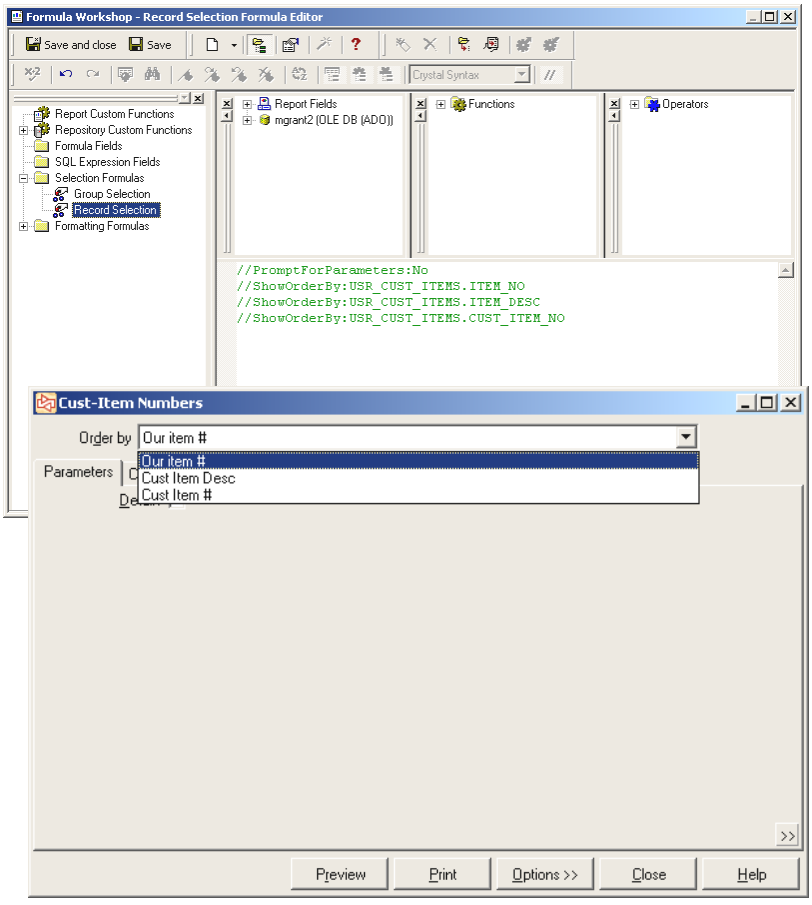


Record Selection Formula settings in Crystal:

2) //BeforeOrderBy:

3) User’s “order-by” choice (//ShowOrderBy: settings)

4) //AfterOrderBy:



Allow Filtering of One or More Tables

- Add **//ShowFilters:** comment to record selection formula in main report to identify table(s) for which filtering will be available.
- If **//ShowFilters:** is not included in record selection formula, filter tab will appear for every table in the main report.
- If **//ShowFilters:** is added to record selection formula for an existing CPSQL report, also add comments for any existing filter tabs to retain them.

//ShowFilters:Table1, Table2, etc.

Separate tab appears for each table on report parameter form. Customize filters to control filter fields for each table.

//ShowFilters:None

Do not show any filter tabs for report.

//ApplyFilters:No or **Yes** (default)

Should filter be applied to main report? Should filter be applied to sub-report?

If Yes, Counterpoint applies as many user-selected filters as it can to main report and each sub-report.

If No, user-selected filters are not applied to main report or sub-reports.

Examples of //ApplyFilters:No only in sub-report:

- Filter to show any ticket where a price override was performed on a line item, but show all line items on ticket.
- Print inventory information for all locations if item is out-of-stock at at least one location.
- Print all open items for any customer who has at least one unapplied credit or payment.

Request Parameters

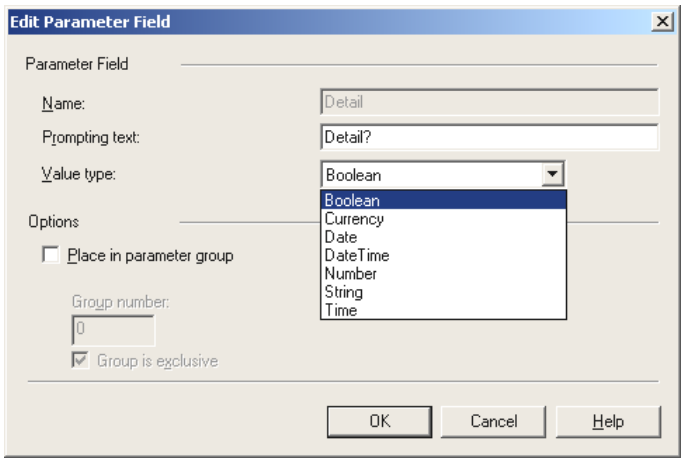
To request other parameters for report (e.g., date range, detail or summary, etc.)

Method 1: Request Crystal's parameters in Counterpoint, instead of in Crystal

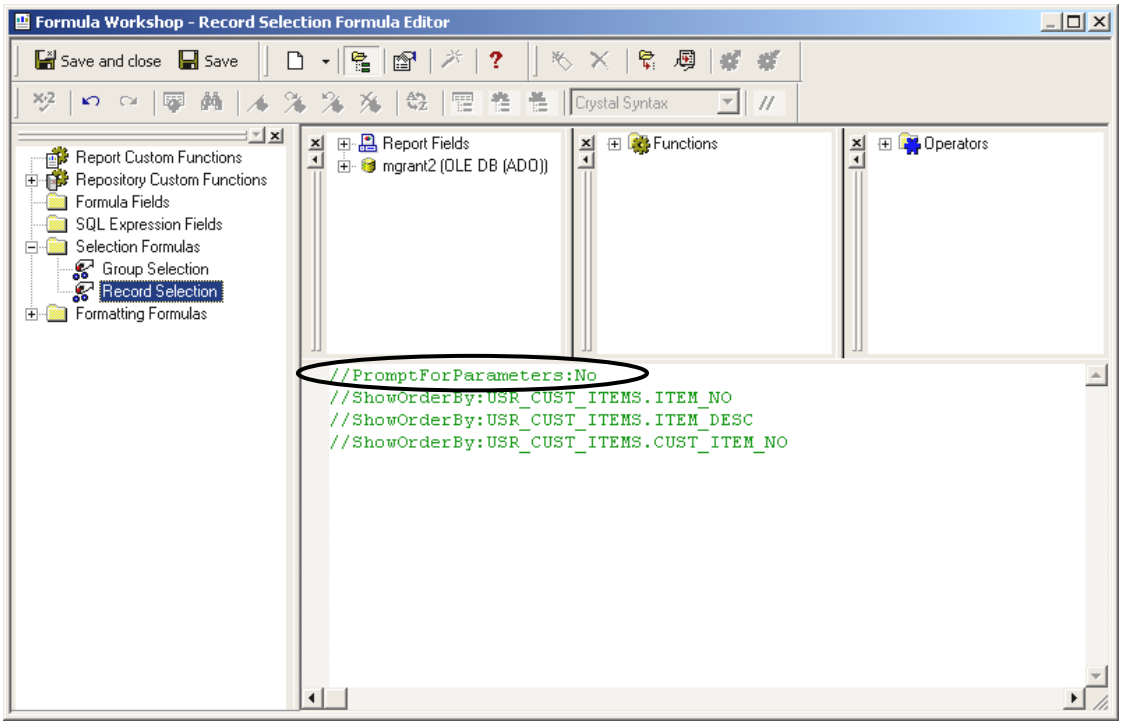
//PromptForParameters:No

In Crystal

Build parameters where **Value type** is Boolean, Currency, Date, Number, String, or Time

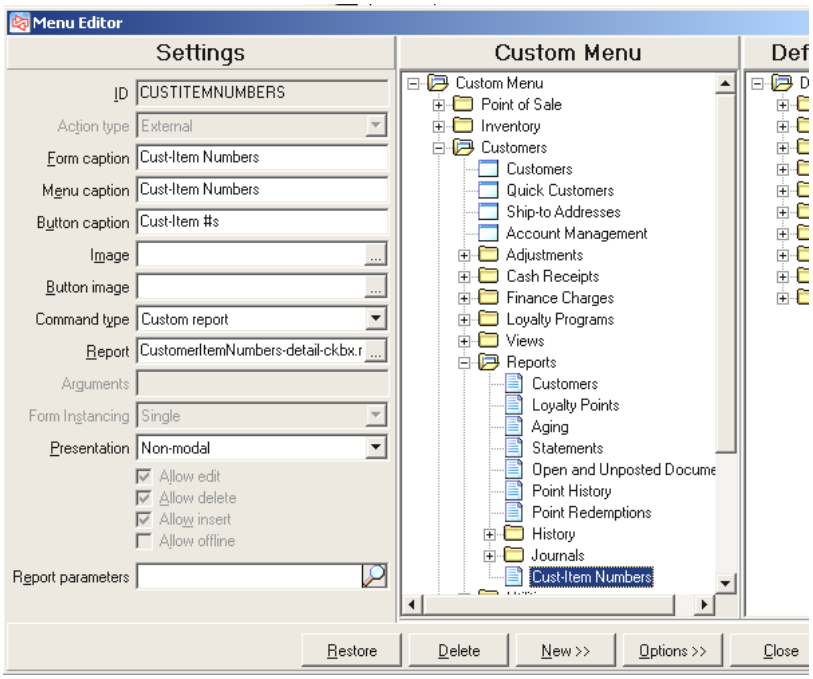


Add **//PromptForParameters:No** to record selection formula.

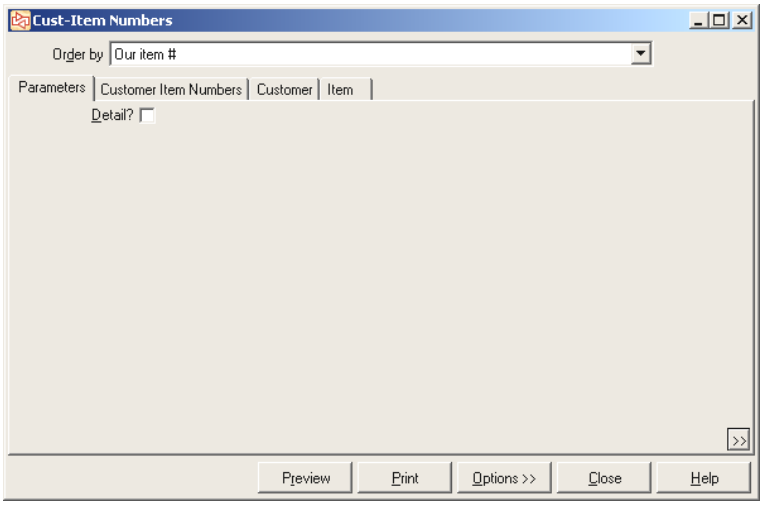


In Counterpoint

Add menu item.



When report is run in Counterpoint:



Does not support:

- parameters with multiple default values (“pick list”)
- multiple Boolean parameters that are mutually exclusive (choices display as a list)
- entry of date and time in a DateTime parameter (use separate Date and Time parameters instead)

Request Parameters

***Method 2**:* Build Crystal formula field and define corresponding data dictionary entries

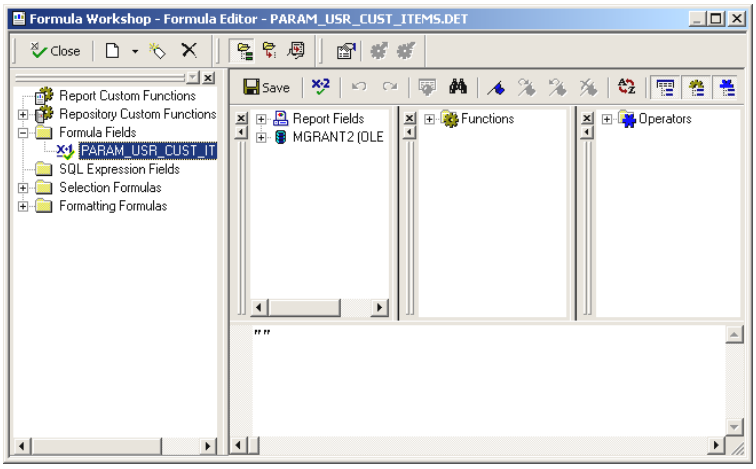
For string parameters, allows
- lookups (specify lookup table and field in data dictionary)
- pick lists (define mapping values in data dictionary)

<u>In Crystal</u>

Create formula named PARAM_table.field (e.g., PARAM_USR_CUST_ITEMS.DET).

Formula content only identifies data type of parameter:

"" = string
1 = numeric
#06/23/03# = date
False = Boolean

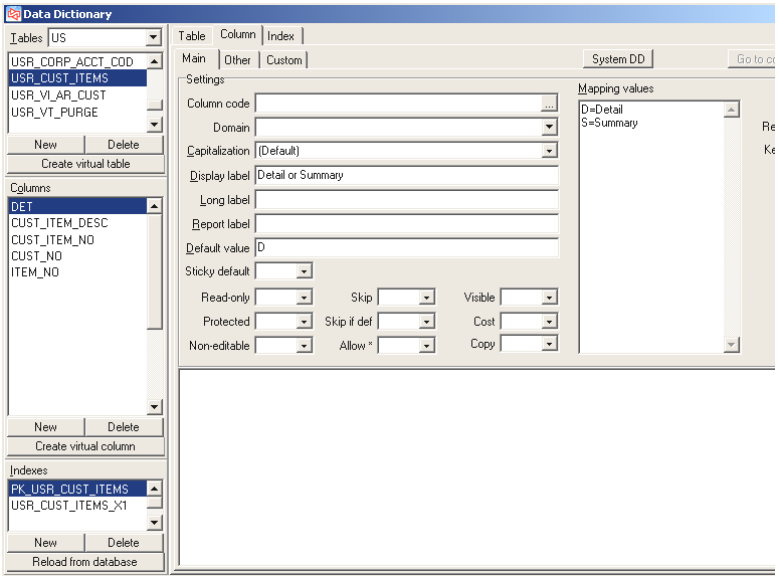


<u>In Counterpoint</u>

Define data dictionary entries for table and field used in formula name.

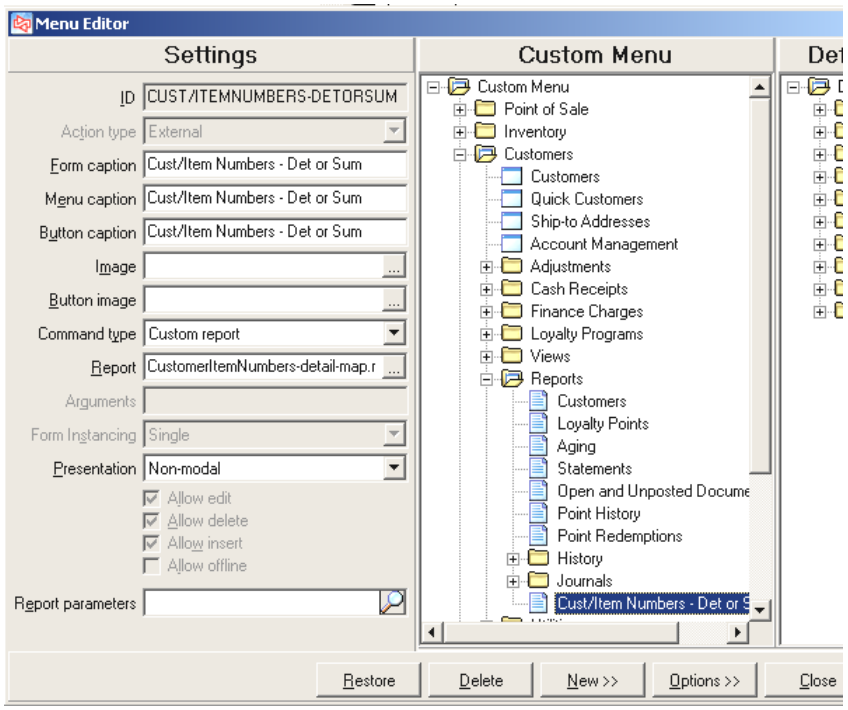
In this case, the column **DET** is created as a virtual column.

Column settings determine type of parameter (see below).

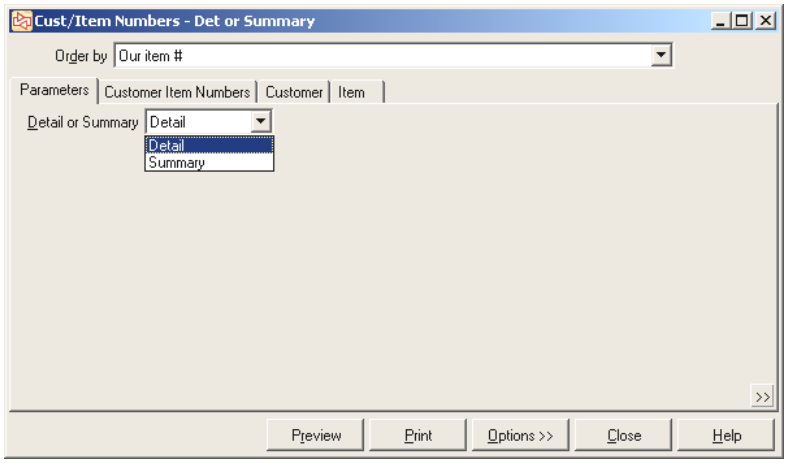


Type of Parameter	Data Dictionary setting for column
Checkbox	Set “Domain” to **Boolean**
Edit with lookup	Set “Lookup table” and “Lookup field”
Numeric	Set “Domain” to **Numeric**
Edit with selection list	Enter “Mapping values”
Date	Set “Domain” to **Date**
Time	Set “Domain” to **Time**

Add the menu item.



When the report is run in Counterpoint:



Change label on Print Button, Preview Button or Email Button

Add **//PrintCaption:YourText** comment to record selection formula in main report to change label of the “Print” button.

Example: //PrintCaption:To Paper

Add **//PreviewCaption:YourText** comment to record selection formula in main report to change label of the “Preview” button.

Example: //PreviewCaption:To Screen

Add **//EmailCaption:YourText** comment to record selection formula in main report to change label of the “Email” button.

Example: //EmailCaption:To Web

YourText value must be short enough to fit on existing button size; text will not wrap to next line or result in enlarged button

Prevent preview, print, email, save, printer setup, refresh, or export

Add to record selection formula in main report

//AllowPreview:No or Yes	disable Preview option
//AllowPrint:No or Yes	disable Print option
//AllowEmail:No or Yes	disable Email option
//AllowSave:No or Yes	disable Save to Disk option
//AllowPrinterSetup:No or Yes	disable Printer Setup access
//AllowRefresh:No or Yes	disable Refresh option
//AllowExport:No or Yes	disable Export option

If set to No, associated button and menu selection will be disabled on main parameter form and on Preview form (Refresh can only occur from a Preview form)

If not set, defaults to Yes

@ Parameters in Custom Reports

- Use in report to condition printing based on value of parameter
- Define as parameters in Crystal
- @ parameters do not appear in Counterpoint if //PromptForParameters:N

Parameter to define in Crystal		What it tells you
@RunModeDetail	(*Number* value type)	How report was run 1 = Preview 2 = Print 3 = Export 4 = Save 5 = Refresh
@RptFileName	(*String* value type)	File name of report being run
@SynUser	(*String* value type)	Counterpoint user who is running the report
@SynStore	(*String* value type)	Store from current workgroup (non-PS form) or from Ticket Entry/Touchscreen login
@SynLoc	(*String* value type)	Location from current workgroup (non-PS form) or from Ticket Entry/Touchscreen login
@SynWkgrp	(*String* value type)	Workgroup that report user is logged into
@SynSecod	(*String* value type)	System security code of report user

Examples of use:

- Print ID of user running the report in the report header
- Use security codes to show or hide certain data (e.g., only certain people get to see customer addresses)
- Using a record selection formula, modify data returned based on user (e.g., Filter a customer list to show only customers where SLS_REP = <current CP user>)
- Modify the format of the report so that it prints in a larger font for a particular user

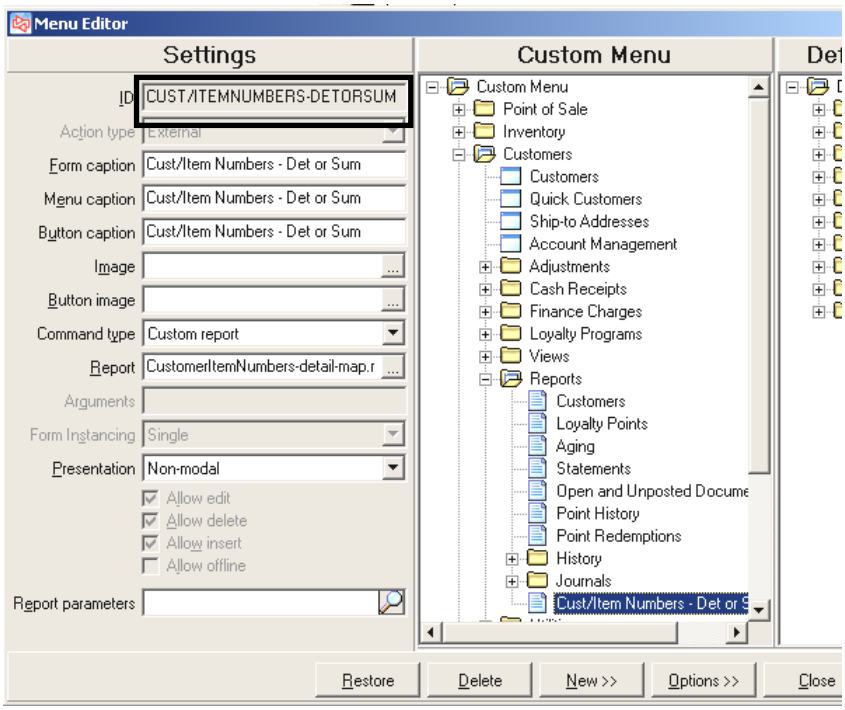
Saved Parameters for Custom Reports

- Save parameter values, type of report, filters, and order by

- For custom reports, parameters saved under Menu ID name (not Form Class)

Use same Menu ID when adding report to different menu codes to use same saved parameters

Optionally identify specific parameters to automatically load at start-up



Running Custom Crystal Reports

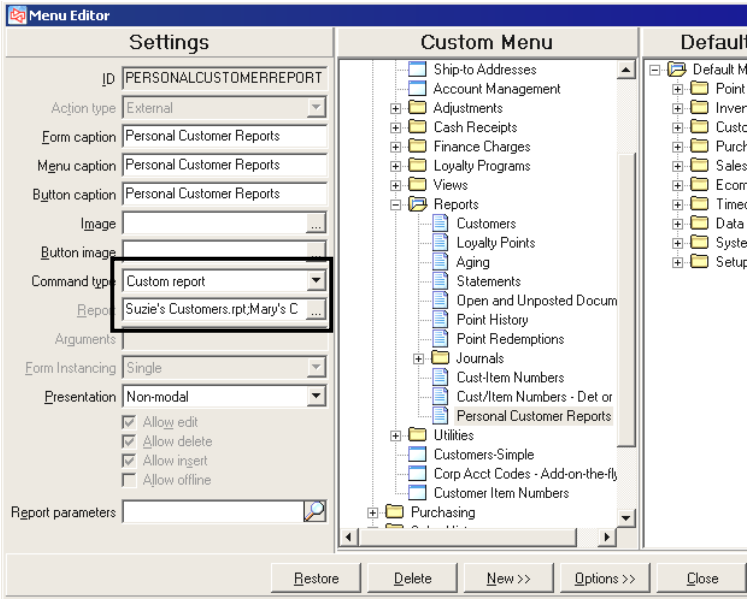
Multiple Reports Run from a Single Menu Selection

Use **Setup / System / Menu Codes** to add the new menu item.

Use a Custom Report **Command type**.

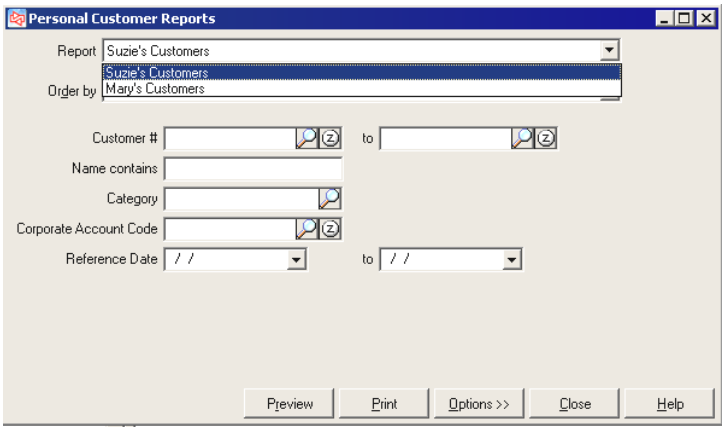
For **Report**, enter the name of each .rpt file to include, separating them with a semi-colon (;).

If you use the Browse button to find them, the reports must be in the same folder (e.g., your company's Reports folder) so you can Ctrl-click them.



When you run the menu selection, the report names appear in the **Report** drop-down list.

Any parameter fields, order-by options, and filters defined for a report will automatically appear when the report is selected.



Stored Procedure Reports

Stored Procedure = compiled SQL program consisting of one or more SQL statements that runs on server and returns data set

Stored Procedure report = report designed around returned data set

Filters:

- //ShowFilters:Table1, Table2 in record selection formula

  Include *TableName_Filter* varchar parameter in stored procedure to accept filters passed to report.

- Do not use //ApplyFilters:No since stored procedure must apply filters to report

Order by

- //ShowOrderBy:Table.Field, //BeforeOrderBy:, //AfterOrderBy: in record selection formula

  Include one *OrderBy* varchar parameter in stored procedure to accept order-by choice passed to report

- Stored procedure must apply order-by selection to report

Parameters

- //PromptForParameters:No in record selection formula

  Declare *@RunMode* parameter in stored procedure and include

```
if @RunMode = 0
begin
    set @SQL = @SQL + ' where 1 = 0' + @CRLF
    set @Filter = 'Y'
end
```

  in stored procedure to provide speedy method for Crystal's verification of dataset structure

- Do not use Crystal Formula/Data Dictionary method to request additional parameters.

- All @ parameters can be used in stored procedure reports

Macros for use in Crystal Formulas and Text Objects

Text object macro	Formula macro	Meaning
%<Table>.<Field>%	REPLCP_%<Table>.<Field>%	Data Dictionary column label
%TABLE.<Table>%	REPLCP_%TABLE><Table>%	Data Dictionary table description
%[<Table>.<Field>]%	REPLCP_%[<Table>.<Field>]%	Value of a global object
%<Control>%	REPLCP_<Control>	Value of parameter form control
----	REPLCP_<Table>.<Field>_MV	Convert storage value of a mapping value to display value
----	REPLCP_<Table>.<Field>_Legend	Legend of mapping values (e.g., "I=Inventory, D=Discount, …")
----	REPLCP_<Control>_ItemIndex	Numeric item index of a parameter form combobox
----	REPLCP_<Control>_MappedValue	Internal storage value of a parameter form combobox (not as fragile as _ItemIndex)
----	REPLCP_ViewCost *	Cost security setting (Y/N)
----	REPLCP_UsingLocPrices	Location prices active (Y/N)
----	REPLCP_SiteType	Hub, Remote, Offline, or Single-site
%SYNTOPLEVEL%	REPLCP_%SYNTOPLEVEL%	Top-level directory name (UNC)
%ALIAS%	REPLCP_%ALIAS%	Company alias
%ORDERBY%	----	English order-by
----	REPLCP_OrderBy	SQL order-by from Counterpoint, including any prefix or suffix; omits any order-by from Crystal
%USERFILTER%	REPLCP_UserFilter	English filter text; formula will include line breaks, which text object omits
%SQLFILTER%	----	SQL-syntax filter text
----	REPLCP_ReportName	Internal name of standard report being run (useful when .rpt report is used for multiple purposes, such as Journal/History report)
----	PARAM_<Table>.<Field>	Add control to parameter tab of report, based on data dictionary definition of table and field. Value of formula is whatever user types into the control.

* Using the REPLCP_ViewCost Formula Macro
1) Create a formula named **REPLCP_ViewCost** that contains the value **True**
2) Right-click column to be conditioned on View Cost authorization, and select **Format field**
3) Select to **Suppress**, and build a suppression formula: **{@REPLCP_ViewCost}=False**

What can be changed?

- Add or remove toolbars and menus
- Dock toolbars and menus to any border, or allow to "float"
- Change options that appear, in what order they appear, name of option, and icon
- Add new options and buttons
- Shortcut key assignments
- Flyover hints
- Toolbar icons

Where is the toolbar customized?

Select **Setup / System / Configuration / Customize Toolbars**

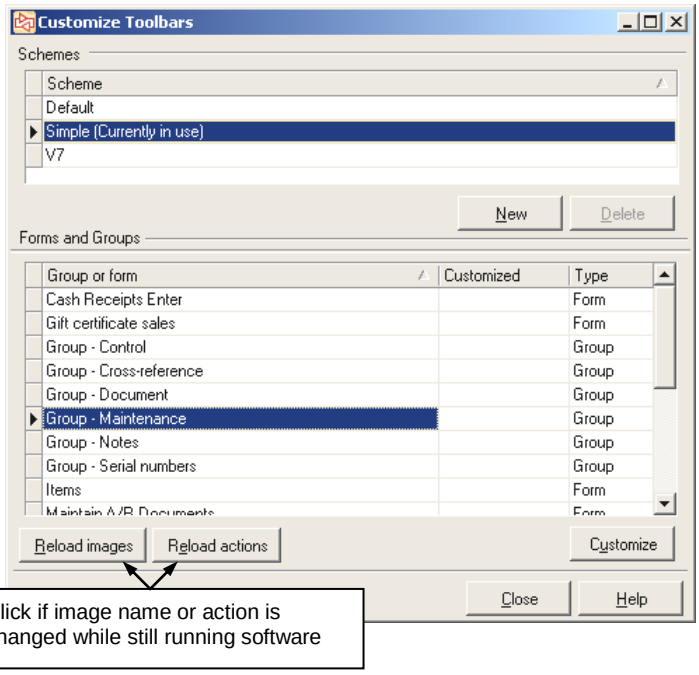
or

Right-click title bar of specific menu selection and select **Customize toolbar**

Select the Scheme whose toolbar or menu will be customized.

Select the individual form or group of forms (Type = **Form** or **Group**) to be customized.

Click Customize



Customizing Toolbars

Try it Yourself!

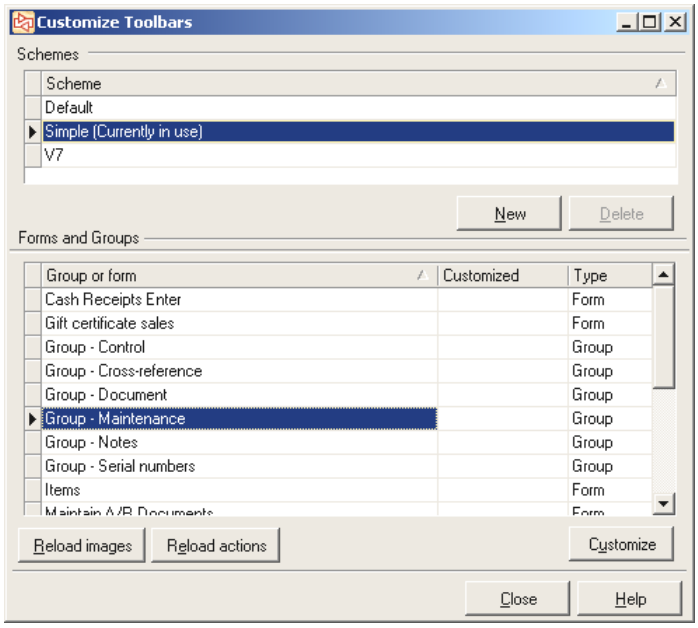
This exercise shows how to add a calculator to the toolbar for maintenance forms, when using the simple scheme toolbar.

1. Select **Setup / System / Configuration / Customize Toolbars.**

   Select the **Simple** scheme.

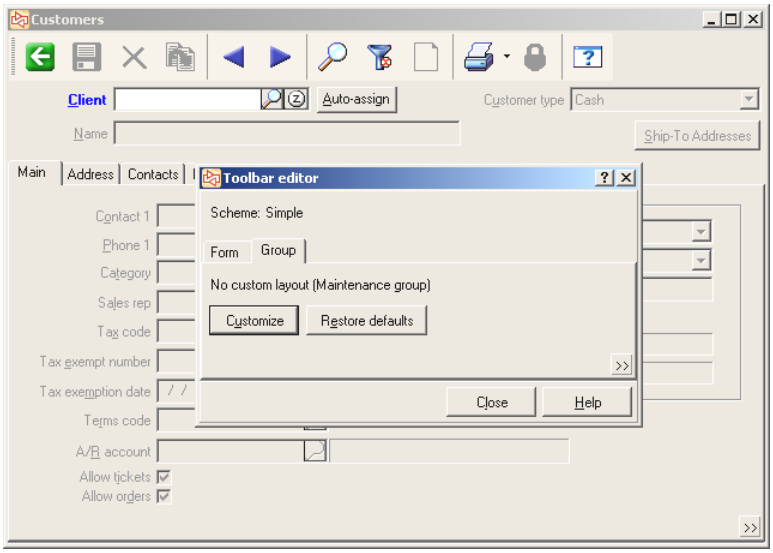
   Select the **Group – Maintenance** form group.

   Click Customize .



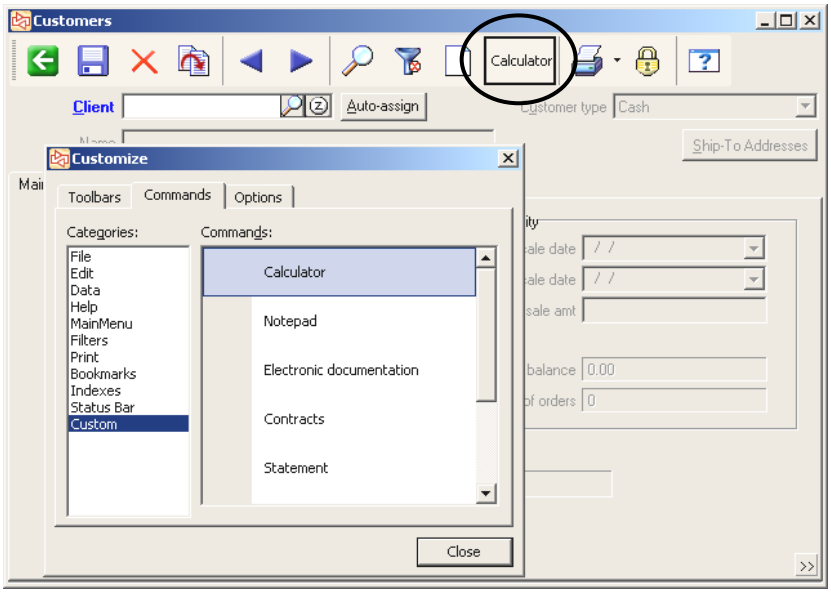
2. The Toolbar Editor displays, along with the **Customers** maintenance form (the 'representative' maintenance form).

   Click Customize .



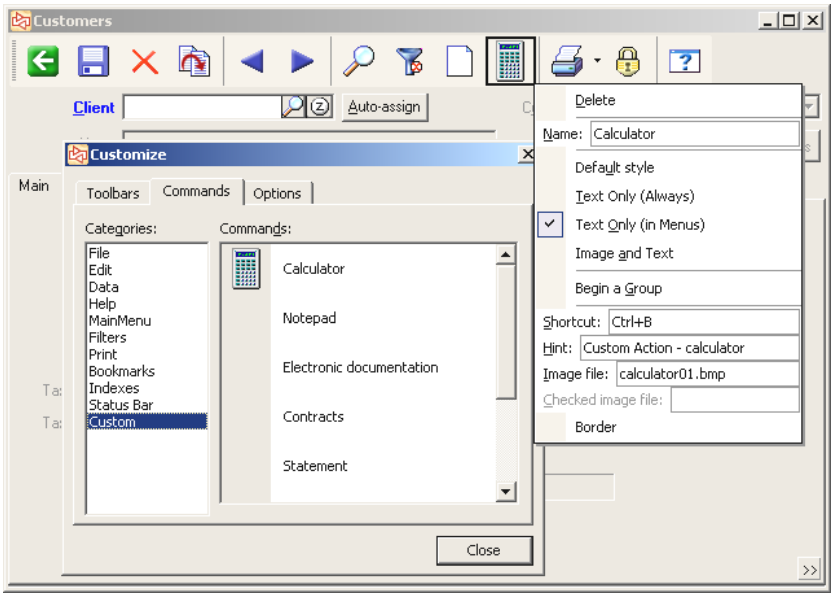
3. Switch to the Commands tab and select the **Custom** category. Several commands appear in this category.

Click **Calculator** and drag it to the desired position on the toolbar.



4. Right-click the new Calculator button and change it settings as shown here.

If you do not have an image file for the icon, leave "Image file" blank and select "Text Only (Always)".



When finished, click Close. Click Close when the Toolbar Editor window displays. Answer Yes when asked if you want to save the changes to the toolbar layout. Then click Close when the Customize Toolbars window appears.

5. Select **Purchasing / Vendors** from the menu. Since this function falls into the Maintenance group for toolbars, you should see the calculator button on the toolbar.

End of Exercise

Customizing Toolbars

What can new Custom buttons do?

➡ **Call an external file**

Batch file
Visual Basic script
.Exe file
Document (e.g., .DOC or .PDF)

➡ **Call a Counterpoint form or menu selection**

If report selection, pre-fills with data from the calling form

➡ **Call a Crystal Report**

Pre-fills with data from the calling form, if possible

Role of Actions.xml

- Located in TopLevel / System / Actions
- Definitions of additional Custom functions that can be added to toolbars

```
<actions>

<!-- Call external programs (must be in path) -->
<action id="Calculator">
<filetorun>Calc.exe</filetorun>
<caption>Calculator</caption>
<image></image>
<shortcut>Ctrl+B</shortcut>
<hint>Custom Action - calculator</hint>
<parameters></parameters>
</action>

<action id="Notepad">
<filetorun>Notepad.exe</filetorun>
<caption>Notepad</caption>
<image></image>
<shortcut></shortcut>
<hint>Custom Action - Notepad</hint>
<parameters></parameters>
</action>

<action id="InTouchManual">
<filetorun>"%SYNBIN%..\pdf\CounterPoint.pdf"</filetorun>
<caption>Electronic documentation</caption>
</action>

<!-- Call a form -->
<action id="Contracts" actiontype="menu">
<filetorun>TfrmContractGrpMaint</filetorun>
<parameters></parameters>
<caption>Contracts</caption>
<image>BlueCircle.bmp</image>
<hint>Constract maintenance</hint>
<shortcut></shortcut>
</action>

<!-- Call an existing report form - do not auto-preview -->
<action id="Statements" actiontype="menu" preview="No">
<filetorun>TfrmARStatements</filetorun>
<parameters></parameters>
<caption>Statement</caption>
<image></image>
<hint>Statement</hint>
<shortcut></shortcut>
</action>
```

Customizing Toolbars

Adding a custom action to Actions.xml

- Create new Actions.xml with text editor (supplements System Actions.xml)
- Save as Actions.xml in company's Actions directory
- Open in Internet Explorer to see syntax errors

For each action:

```
<action id="wwwwwww"  actiontype="xxxxx" preview="Yes/No"
 default-report-params="yyyyy">
<filetorun>name</filetorun>
<parameters>zzzzzz</parameters>
<caption>text</caption>
<image>filename</image>
<hint>text</hint>
<shortcut>keystrokes</shortcut>
</action>
```

Note: <**switchname**> and </**switchname**> must be lowercase

action id *	Identifier for action; no spaces or special characters
actiontype	**menu** if action calls a Counterpoint menu item **report** if action calls a Crystal report
preview	**Yes** if action calls a Crystal report, to allow report to be previewed
default-report-params	Name of saved parameters to load if action calls a Crystal report
filetorun *	Name of program if action calls an external file (if no path specified, must be in workstation's path or file extension association exists) Menu item ID if action calls a Counterpoint menu item Name of .rpt file if action calls a Crystal report
parameters	Parameters to pass on command line if action calls an external file
caption	Default text label for action's button
image	Default image file for action's button. If no path specified, looks in company's Images directory and then System/Images
hint	Flyover text for action's button
shortcut	Keystrokes that select the action

* = required entry

Try it Yourself!

This exercise shows how to add a button to the toolbar in Ticket Entry that runs the Customers List.

1. Use Notepad to edit the Actions.xml file in the Actions folder for your company.

   Add an entry for a new action between "<actions>" and "</actions>" so that it looks like this:

```
<actions>

<!-- Call the Customers / Reports / Customers menu item -->
<action id="CustReport" actiontype="menu">
<filetorun>TFRMARCUSTOMERSRPT</filetorun>
<parameters></parameters>
<caption>Cust List</caption>
<image></image>
<hint></hint>
<shortcut></shortcut>
</action>

</actions>
```

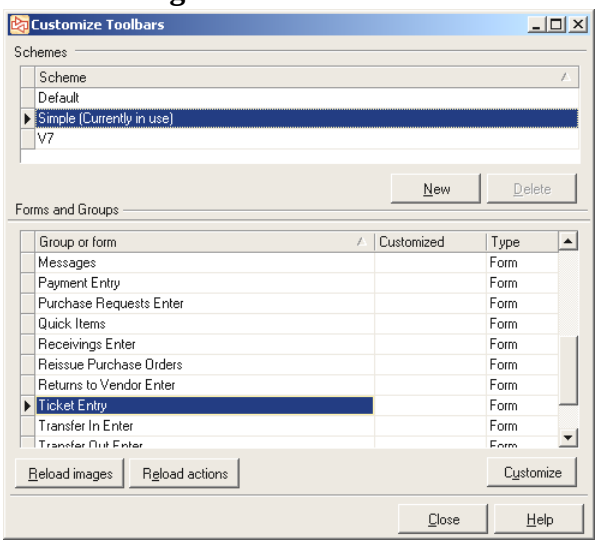
   Save the file.

2. In Internet Explorer, select to Open the file to verify the syntax of your entries.

3. Start Counterpoint and select **Setup / System / Configuration / Customize Toolbars.**

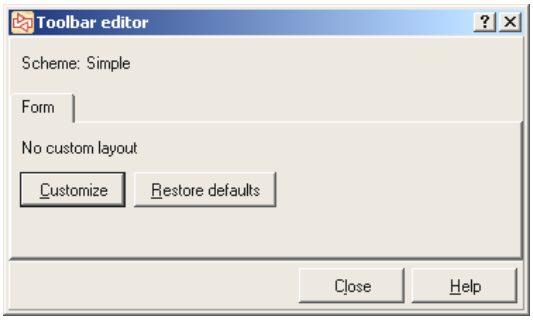
   Select the Simple scheme and the Ticket Entry form.

   Click Customize .



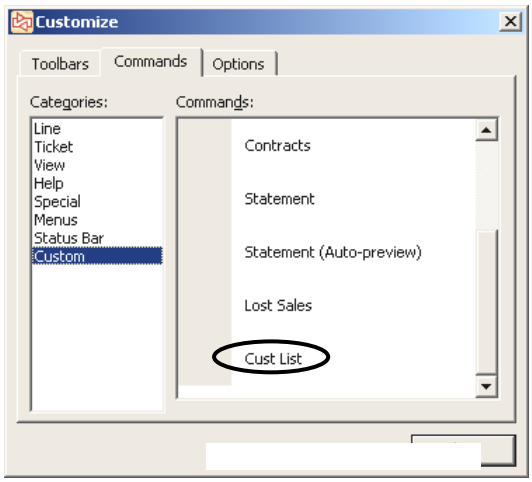
4. When the ticket entry login appears, click OK to log in with the default values.

The toolbar editor appears, with the ticket entry form in the background.



Click Customize .

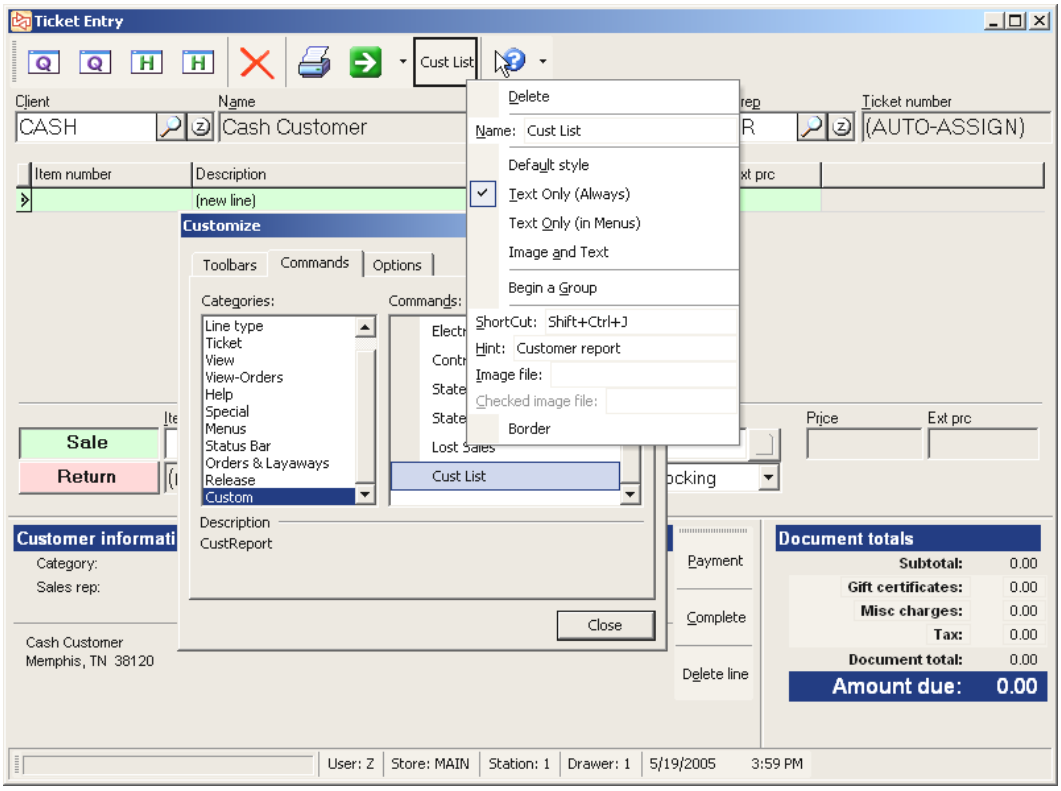
When the Customize window appears, switch to the Commands tab and select the **Custom** category.



Your new action "Cust List" should appear at the bottom of the commands.

Drag **Cust List** to the desired position on the ticket entry toolbar.

5. Right-click the new Cust List button and change its settings as shown here.



6. When finished, click Close in the Customize window.

   Click Close when the Toolbar Editor window displays. Answer Yes when asked if you want to save the changes to the toolbar layout.

   Then click Close when the Customize Toolbars window appears.

7. Click the "Cust List" button on the ticket entry toolbar. The filter for the Customers report should appear. Preview the report.

   By running a menu selection from the toolbar, the function behaves the same way as if you had selected it from the menu. In this case, it was necessary to at least click Preview before the report displayed.

8. Exit from Counterpoint. Navigate to the TopLevel \ Campgolf Training \ Configuration \ Actions directory and reload Actions.xml into a text editor.

   Change your new action entry so that it runs the AR Customers.rpt file directly, as a Crystal report (changes are highlighted):

```
<actions>

<!-- Call the AR Customers.rpt report -->
<action id="CustReport" actiontype="report" preview="Yes">
<filetorun>C:\Program Files(x86)\Radiant Systems\CounterPoint\CPSQL.1\
TopLevel\System\Reports\AR Customers.rpt</filetorun>
<parameters></parameters>
<caption>Cust List</caption>
<image></image>
<hint></hint>
<shortcut></shortcut>
</action>

</actions>
```

   Restart Counterpoint and select **Point of Sale / Tickets / Ticket Entry**. Click the "Cust List" button on the ticket entry toolbar. A preview of the report automatically appears.

End of Exercise

Bonus Exercise:

Add a button to the toolbar in the Vendor add-on-the-fly form so that the full maintenance function for Vendors can be accessed in situations where additional information needs to be specified than what is allowed on the add-on-the-fly form.

Running a Custom Program from a Button

Running a custom program from a toolbar or Touchscreen button

1. Define the .CFG configuration file for the custom program, as usual.

2. Add a menu item that runs the custom program, as usual.

3. Add a "menu" action to custom Actions.xml, similar to this.

```
<!-- Call a Custom Program -->
<action id="CustomProgram" actiontype="menu">
<filetorun>CustomProgram</filetorun>
<parameters>"Argument1 %SYNUSER%"</parameters>
<caption>CustomProgram</caption>
<image></image>
<hint>CustomProgram</hint>
<shortcut></shortcut>
</action>
```

4. Customize a maintenance toolbar to add the new "Custom" button.

   - If maintained table is a "ShowFilters:" table for custom program, it automatically sets the filter to the current maintenance record

     ShowFilters:AR_CUST

   - If custom program uses a "Parameter: Table.Field" type of parameter and the current table is the parameter's "Table", it automatically sets the "Field" in the parameter to the current maintenance record.

     Parameters:AR_CUST.USR_CORP_ACCT_COD

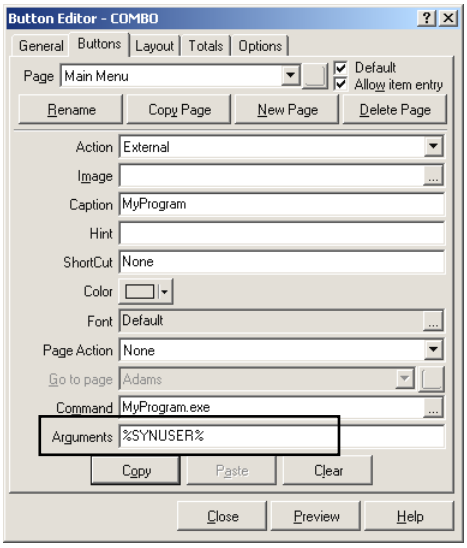
   To disable these automatic pass-forward settings, add this to the custom program's configuration file:

   PassForwardParams: No

5. In Touchscreen Code, define a "Custom action" button and select the new action as the custom action.

Passing variables to an External action

- Use to pass variables as Argument for an External button action



Variable/Argument	Description
%SYNBIN%	Application directory on workstation (e.g., C:\Program Files\Radiant Systems\ CounterPoint\CPSQL.1\Bin)
%SYNTOPLEVEL%	Current top-level directory on server
%SYNDD%	Path to company's data dictionary files
%SYNHELP%	Path to help files (e.g., C:\Program Files\Radiant Systems\CounterPoint\CPSQL.1\Help)
%SYNLOG%	Path to CounterPoint.log (root directory on server)
%WINDOWS%	Path to Windows directory (e.g., C:\Windows)
%SYSTEM%	Path to Windows system directory (e.g., C:\Windows\System32)
%TEMP%	Path to Windows temp directory (e.g., C:\Windows\Temp)
%ALIAS%	Database alias used during login (e.g., Camptown Golf)
%SYNUSER%	User currently logged in to Counterpoint or Ticket Entry/Touchscreen

Course Evaluation

Course Name _____ **Training Date** _____

Trainer _____ **Your Name (optional)** _____

Ranking

Please use the following rankings to complete this evaluation:
1 = Unacceptable, 2 = Did Not Meet My Needs, 3 = Meets My Needs, 4 = Exceeded My Expectations

Registration

1. It was easy to enroll in this course.	1	2	3	4

\Facilities

2. The workspace available was appropriate for the subject matter being taught.	1	2	3	4
3. The room was set up effectively for the curriculum – if PC's were supplied, they were loaded with the relevant software.	1	2	3	4

Materials

4. Manual content was appropriate to the subject being taught.	1	2	3	4
5. The format of the manual was easy to understand.	1	2	3	4

Experience

6. The course prerequisites were clearly stated & appropriate to the topic.	1	2	3	4
7. The practice activities complemented the course.	1	2	3	4
8. The time allotted for the course was adequate.	1	2	3	4
9. I was given an opportunity to contribute during discussions.	1	2	3	4

10. What were your expectations for attending this course?

_____

11. Were your expectations met? ☐ Yes ☐ No
If yes, how were they met?

_____

If no, why not?

_____

Instructor

12. The instructor was prepared.	1	2	3	4
13. The instructor was knowledgeable about the course content.	1	2	3	4
14. The instructor encouraged questions from participants.	1	2	3	4
15. The instructor covered the course objectives.	1	2	3	4
16. What is your overall rating of this Instructor?	1	2	3	4

Comments

17. The best thing about this course was:

_____

18. This course could be improved by:

_____

19. Another course that I would like offered is:

_____

20. I would attend another course offered by Retail Training? ☐ Yes ☐ No
If yes, why?

_____

If no, why not?

_____

Thank you for completing this evaluation!

APPENDIX 1: EXERCISE SOLUTIONS

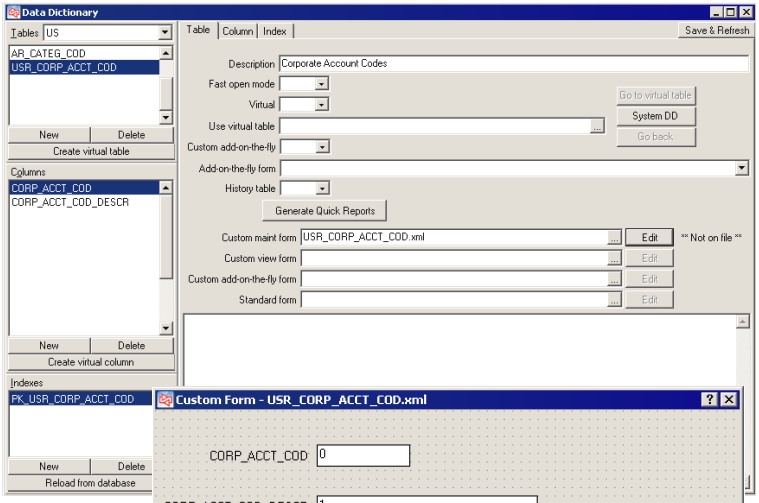
Adding Custom Tables (first exercise)

1) Commands in Query Analyzer:

```
create table  USR_CORP_ACCT_COD
(CORP_ACCT_COD                  T_COD       not null,
 CORP_ACCT_COD_DESCR            T_DESCR     null,
 LST_MAINT_DT                   T_DT        null,
 LST_MAINT_USR_ID               T_USR_ID    null,
 ROW_TS                         timestamp   null,
constraint PK_USR_CORP_ACCT_COD primary key (CORP_ACCT_COD)
)
go

alter table AR_CUST
    add USR_CORP_ACCT_COD           T_COD
    constraint FK_AR_CUST_USR_CORP_ACCT_COD foreign key
        (USR_CORP_ACCT_COD)
    references dbo.USR_CORP_ACCT_COD (CORP_ACCT_COD)
go
```

2) Data Dictionary entries:

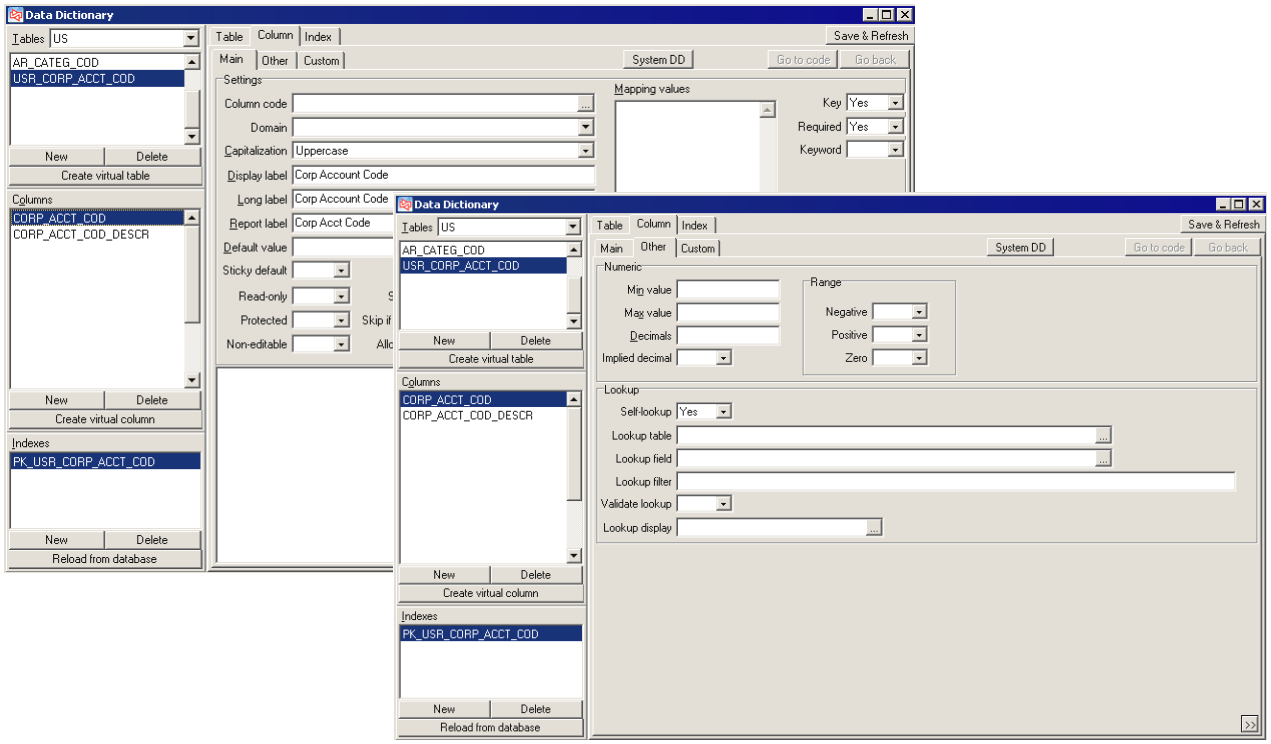
Custom table
USR_CORP_ACCT_COD



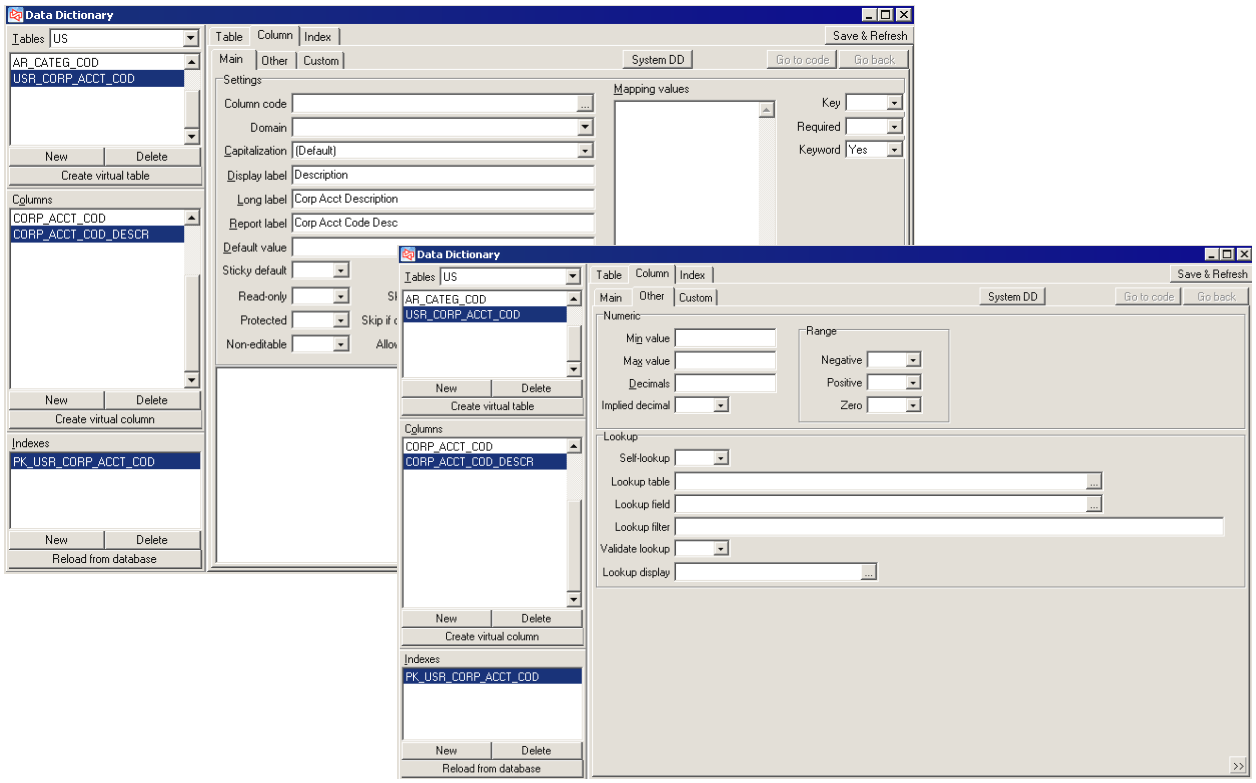
Layout form
USR_CORP_ACCT_COD.XML

Custom fields in USR_CORP_ACCT_COD table

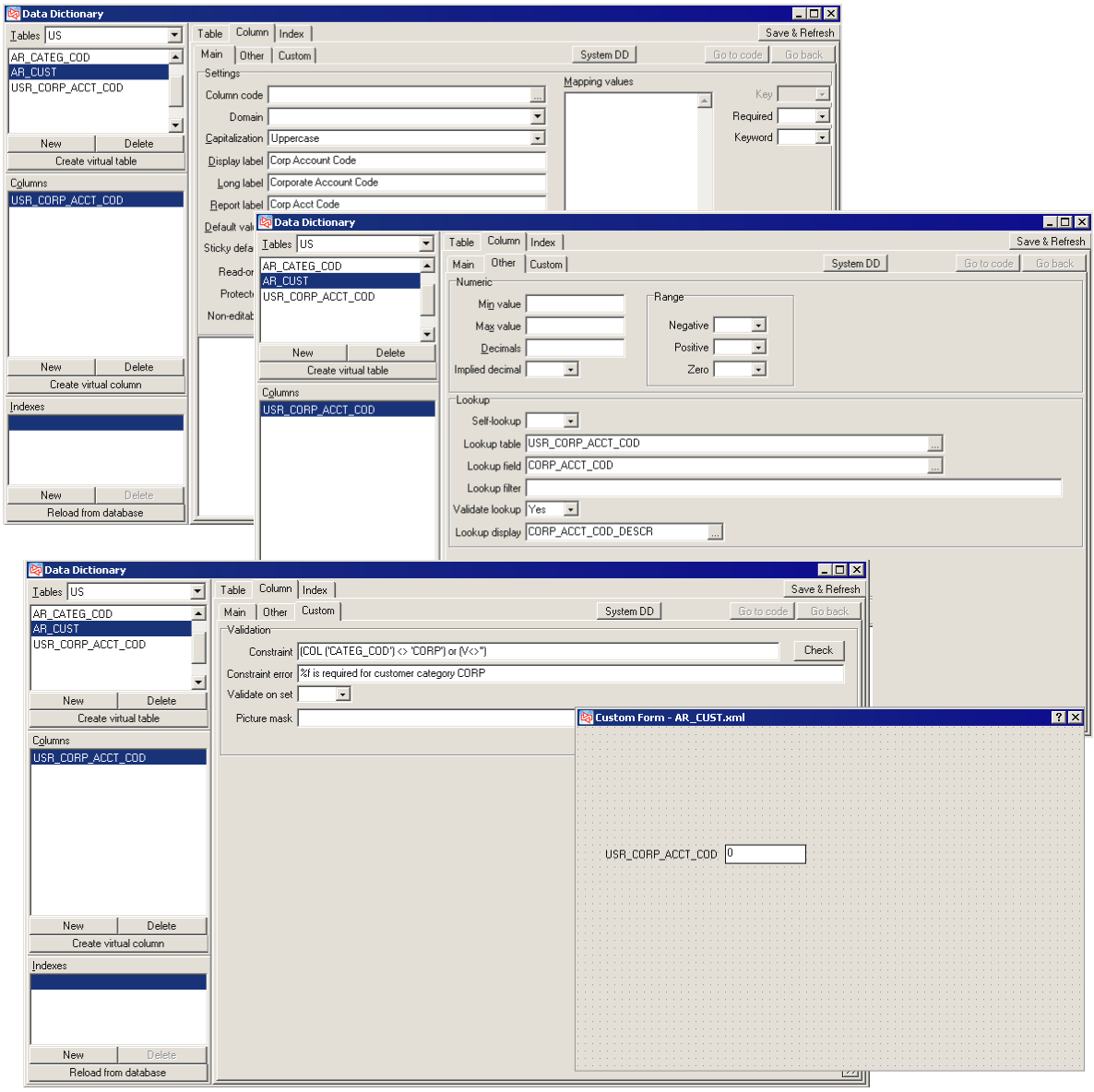
CORP_ACCT_COD



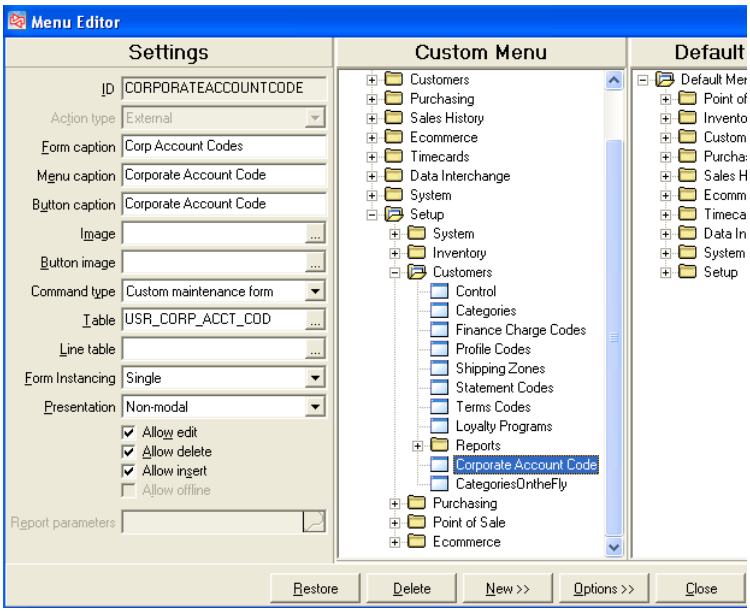
CORP_ACCT_COD_DESCR



Custom field in AR_CUST table
USR_CORP_ACCT_COD



3) Menu Code



Adding Custom Tables (second exercise)

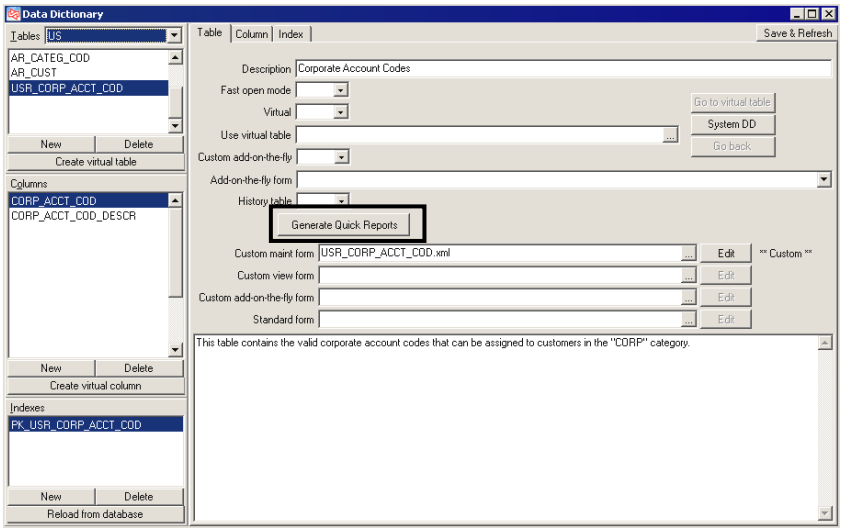
1) Generate a Quick Report template for the new table.

If you have the Crystal Creation API License, select **Setup / System / Configuration / Data Dictionary**.

Select the USR_CORP_ ACCT_COD table.

On the Table tab, click Generate Quick Reports .

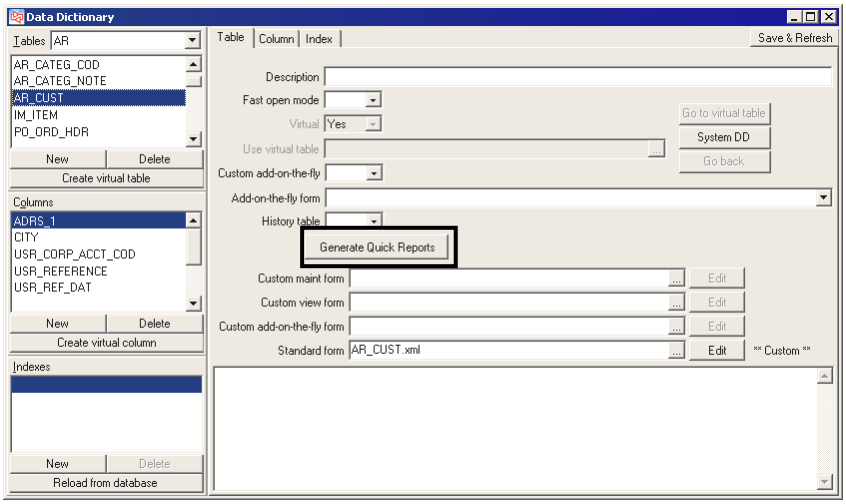
A new file named **USR_CORP_ACCT_COD.rpt** is produced in the company's QuickReports folder.



Next select the AR_CUST table.

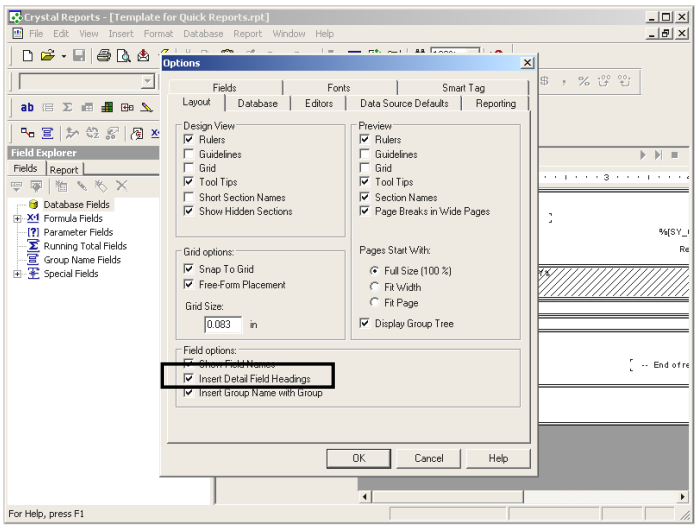
On the Table tab, click Generate Quick Reports .

The old Quick Report template for this table is replaced with the new one that includes the new field.

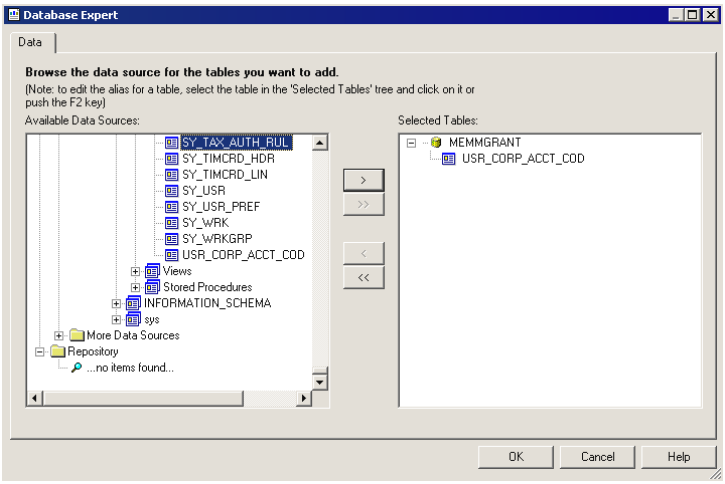


If you do not have the Crystal Creation API License, start Crystal and load **Template for Quick Reports.rpt** (in System / Quick Reports).

Select **File / Options** and ensure that "Insert Detail Field Headings" is enabled.

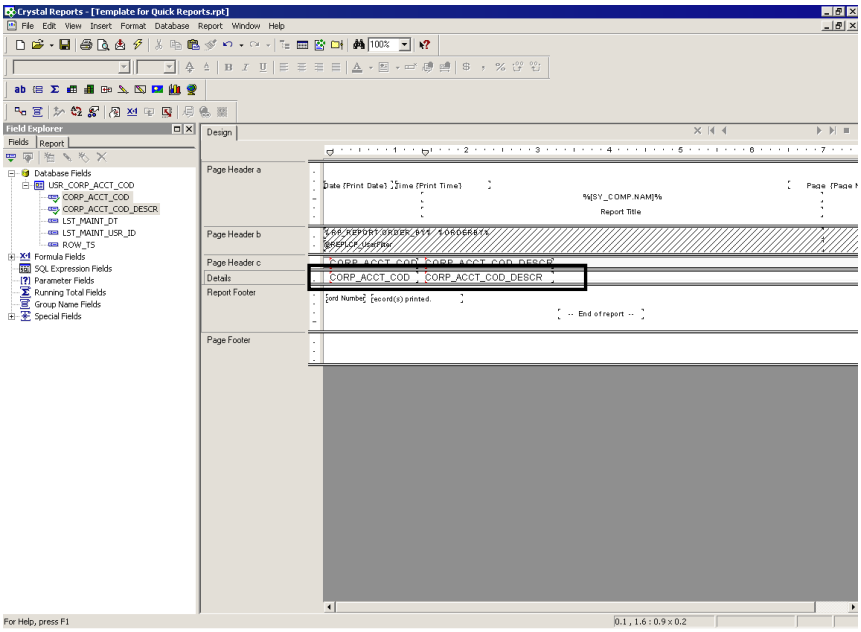


Use **Database / Database Expert** to add your custom table to the report.



Drag the fields in the custom table to the Details band.

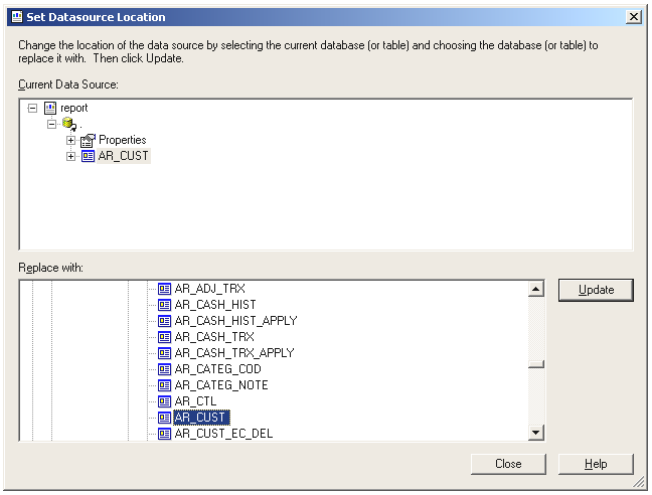
Column headings are automatically added in the Page Header band above.



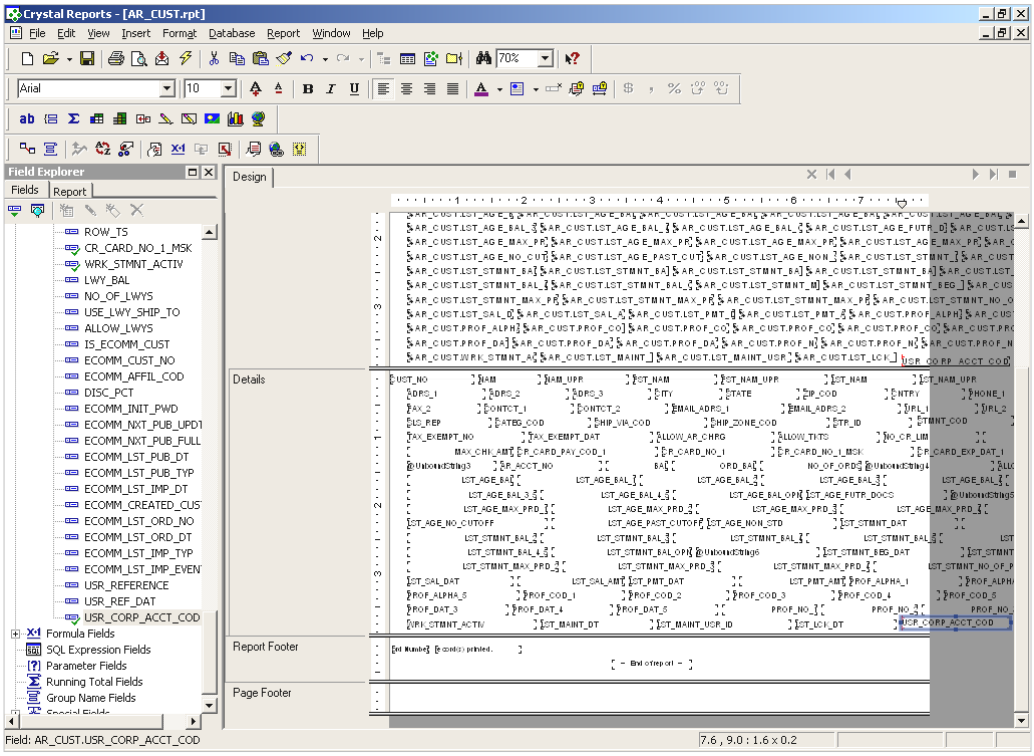
Save the new Quick Report in the company's QuickReports folder, using the table name as the file name (e.g., USR_CORP_ACCT_COD.rpt).

To add the Corporate Account Code field to the Quick Report for Customers, load the existing Quick Report (AR_CUST.rpt in System / QuickReports).

Use **Database / Set Datasource Location** to map the report to your datasource.

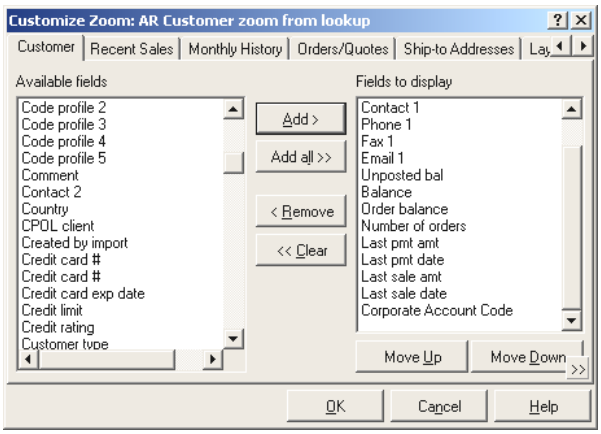


Drag the new field to the Details band. (It doesn't matter where in that band).



Save the revised Quick Report in the company's QuickReports folder, using the table name as the file name (e.g., AR_CUST.rpt).

2) Customize the Customers Zoom.



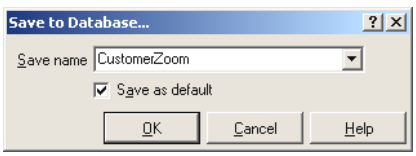
Select **Customers / Customers** and click  to lookup customers. Highlight any customer and click Zoom (F6) (or press F6).

In the Zoom window, click Options >> and select **Customize**.

Add the new field to the list of fields to display. Click OK when completed.

The new field appears at the bottom of the zoom window.

To save the changed zoom window, click Options >> and select **Save as…**.



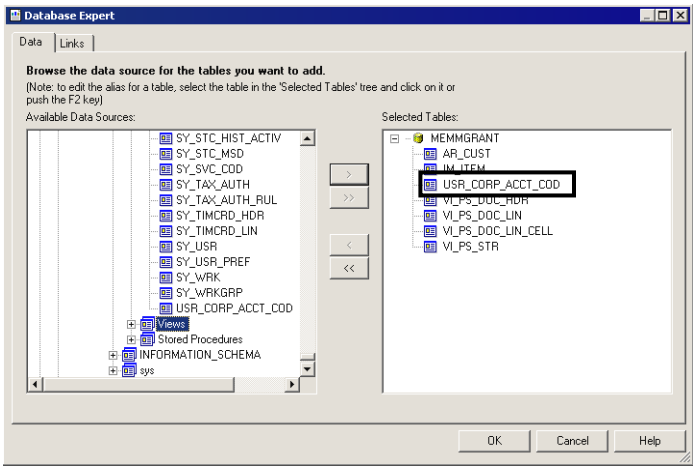
Name the zoom and indicate if you wish it to be the default zoom.

3) Customize the receipt.

To customize the RECEIPT1 form to print the customer's Corporate Account Code next to the customer's phone number, use **Setup / Point of Sale / Form Groups** and display a form group that uses the receipt form. Switch to the Forms tab, select the Receipt form (it must be "Receipt1.rpt") and click Form Editor . Alternatively, you can open Crystal Reports and load RECEIPT1.rpt from the TopLevel\System\PSForms directory.

In Crystal Reports, first use **Database / Set Datasource Location** to "remap" the existing tables in the form to the same tables in your database.

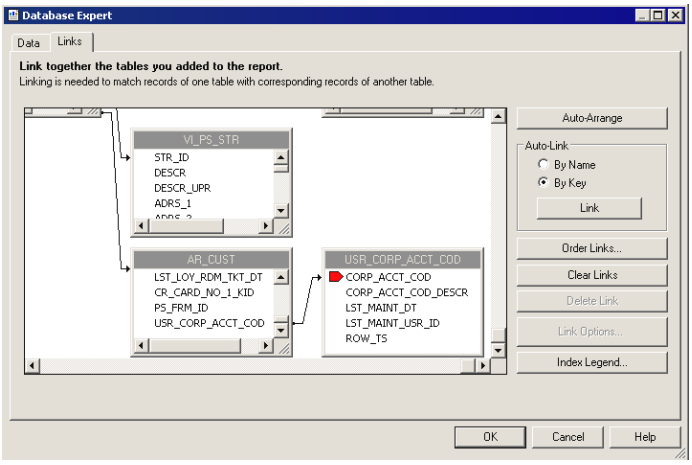
Then use **Database / Database Expert** and select the USR_CORP_ACCT_COD table.



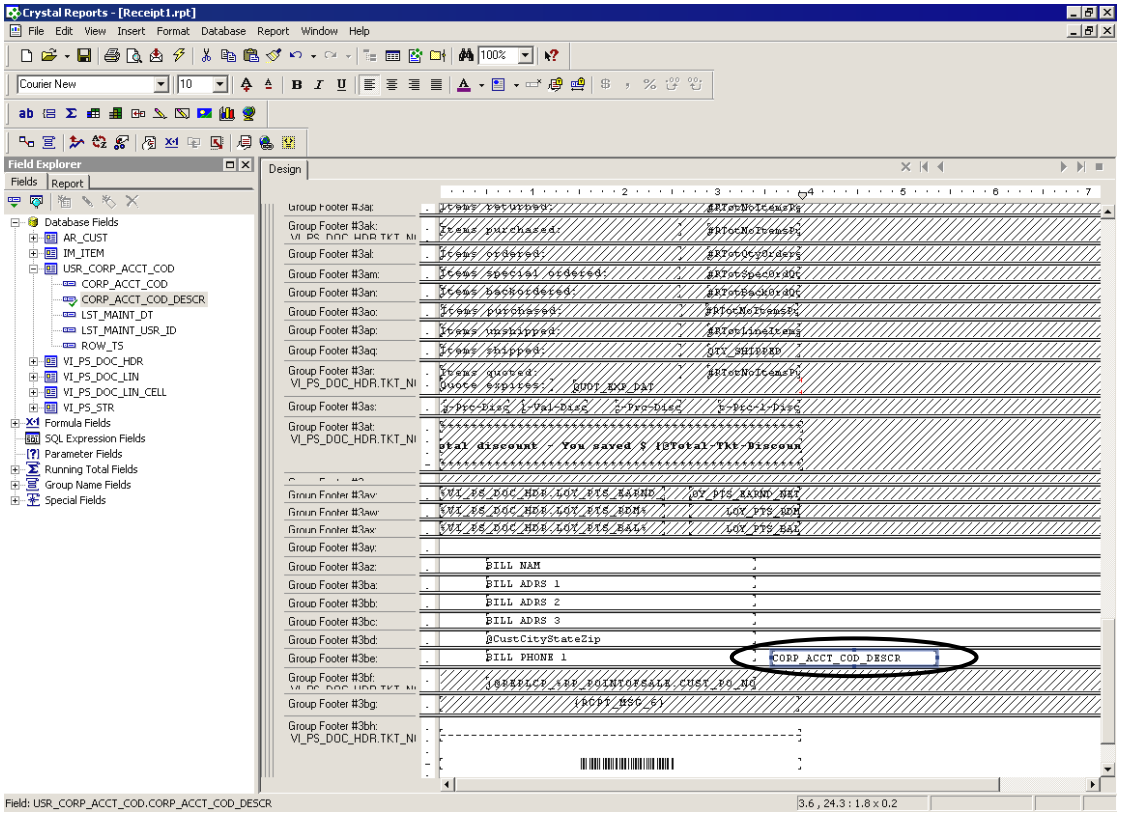
On the Links tab, link from **USR_CORP_ACCT_COD** in the AR_CUST table to **CORP_ACCT_COD** in the USR_CORP_ACCT_COD table. Also use Link Options to change the link to a Left Outer Join.

Do not change any existing links between other tables.

Click OK when completed.



In the Field Explorer, under Database Fields, double-click the USR_CORP_ACCT_COD table and select the field CORP_ACCT_COD_DESCR. Position the field in the Group Footer that contains the PHONE 1 field.



Save the revised form as RECEIPT1.RPT in the PSForms folder for your company (e.g., TopLevel \ Configuration \ DemoGolf \ PSForms).

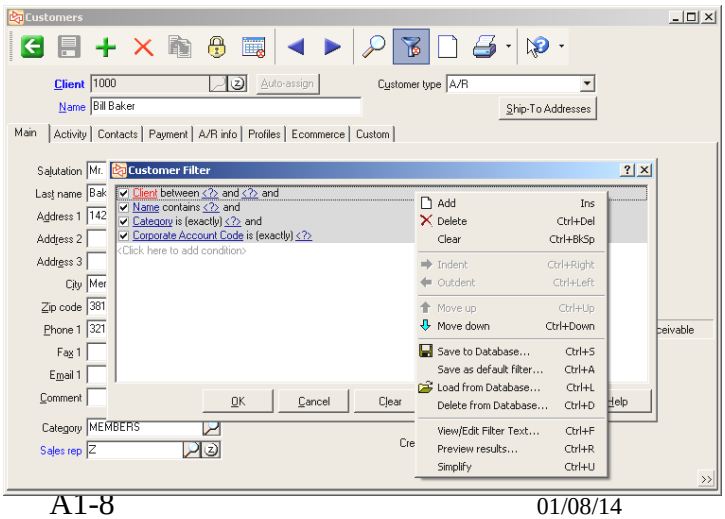
If you want the new field to also print on receipts printed from ticket history, make the same change to the RECEIPT1HISTORY.RPT form.

4) Customize the default filter for the Customers table.

To customize the default filter for the Customers table to add the new Corporate Account Code field, select **Customers / Customers**. Click  on the toolbar.

When the filter window opens, right-click and select to **Customize**. Add a new condition to the filter for the field **Customer Account Code**.

When finished, click  and select **Save as default filter**.



5) Build a default lookup for the new table.

To build a default lookup for the new Corporate Account Code table, select **Setup / Customers / Corporate Account Codes**.

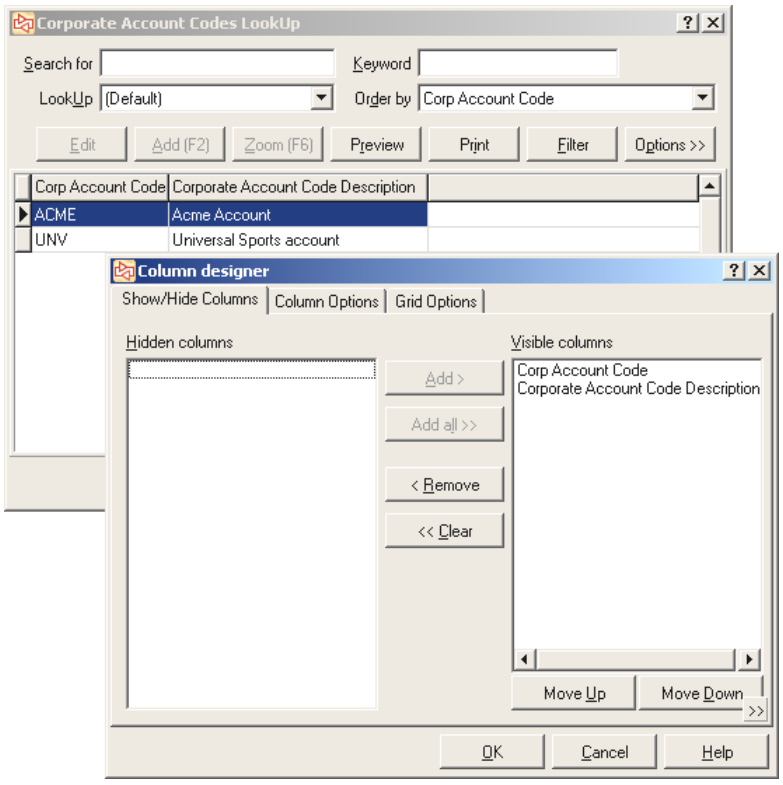
Click  on the toolbar to perform a lookup.

Click Options >> and use the Column Designer to add or remove fields in the list of Visible columns.

(For the Corporate Account Codes table, the only two fields are already in the default lookup so it's not necessary to make any changes.)

Click OK when done.

Then click Options >> and select **Save**.



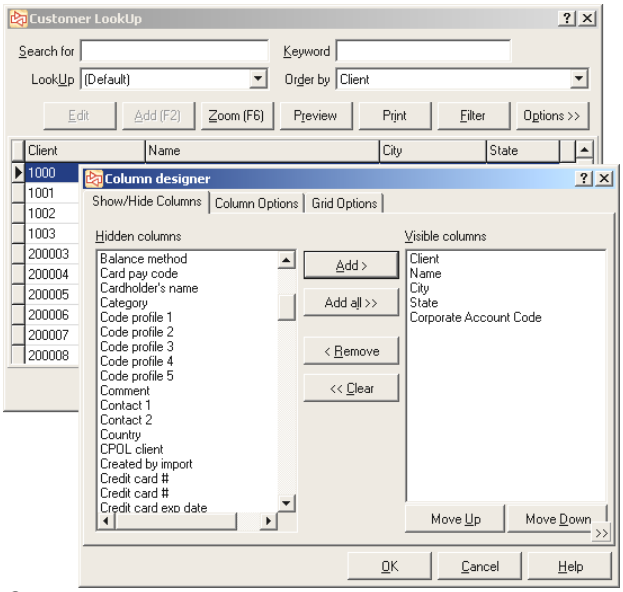
To add the new Corporate Account Code field in the Customers table to the default lookup for customers, select **Customers / Customers.**

Click  on the toolbar to perform a lookup. If necessary, switch to the (Default) lookup for the table.

Click Options >> and use the Column Designer to add the field **Corporate Account Code** to the list of Visible columns.

Click OK when done.

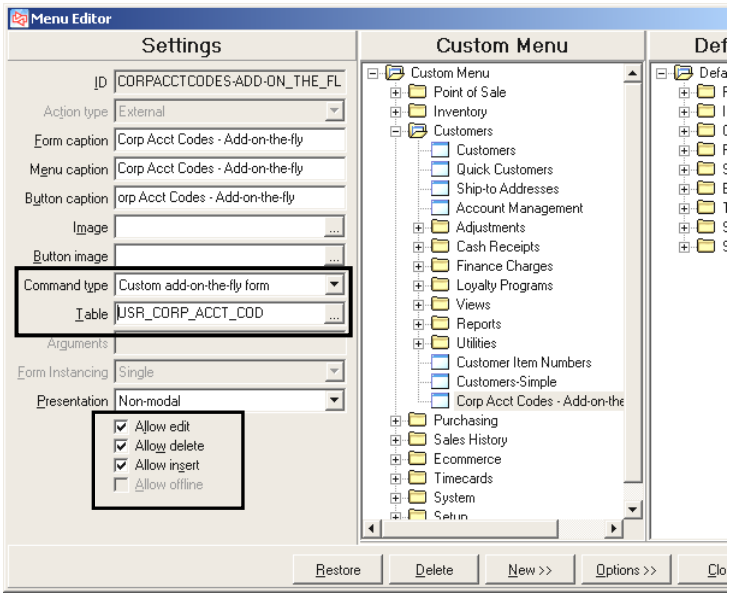
Click Options >> and select **Save** when completed to resave the lookup.



Building a Custom Add-on-the-fly Form for the Corporate Accounts Code Table (first exercise)

1) To add records on the fly to a custom table, your definition in the Data Dictionary will need to be based on a menu selection.

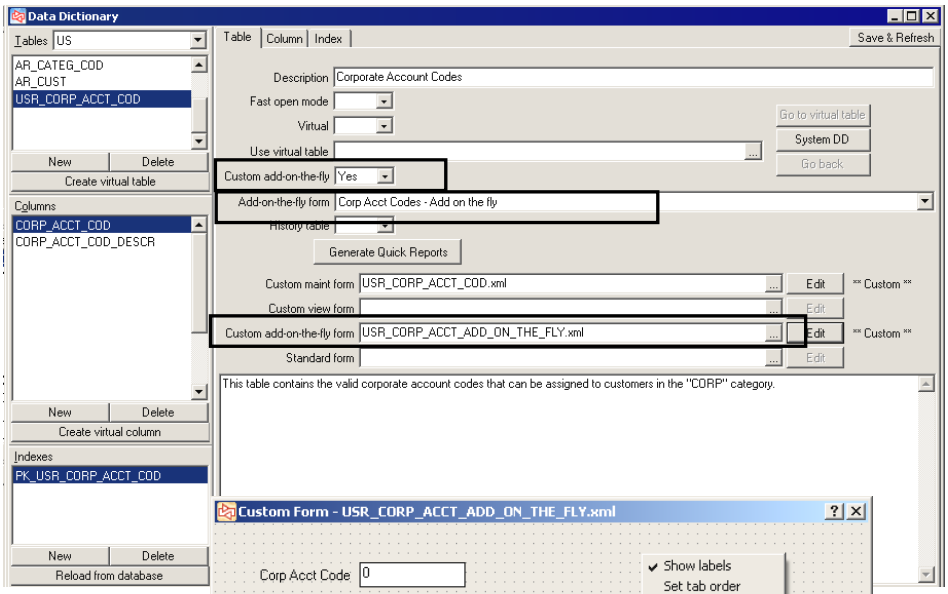
Use **Setup / System / Menu Codes** to add a new menu item to a menu code. Select "Custom add-on-the-fly form" for **Command type** and identify USR_CORP_ACCT_COD for **Table**.



2) On the Table tab in the Data Dictionary for USR_CORP_ACCT_COD:

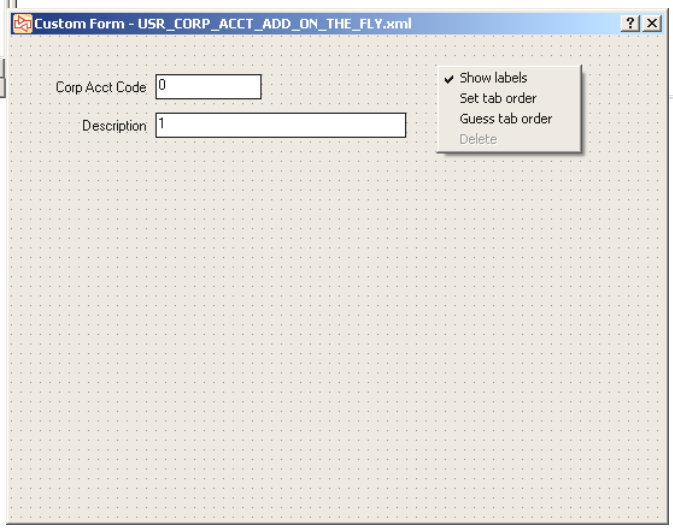
Answer Yes to **Custom add-on-the-fly**.

Pick your Add on the fly menu selection at **Add-on-the-fly-form**.



Specify a layout form for **Custom add-on-the-fly form** and position the fields that will display when adding a new corporate account code on the fly.

(In this case, you could also use the same layout form specified for **Custom maint form**.)

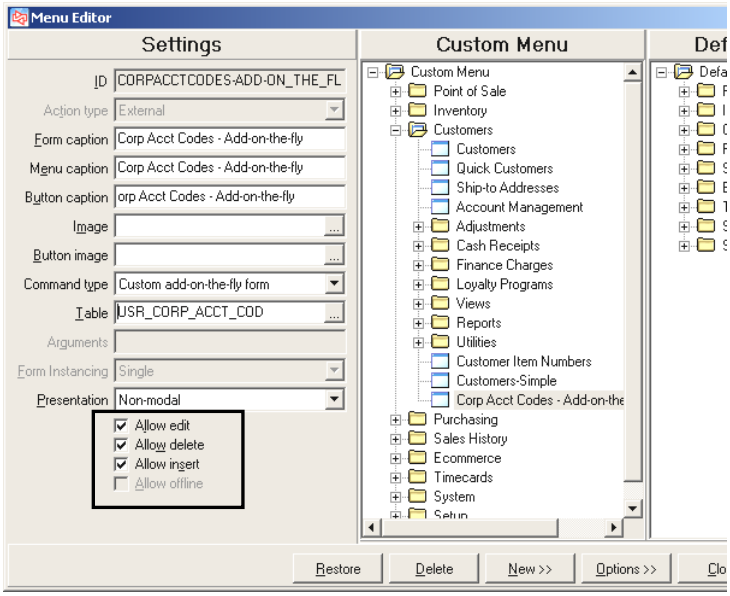


Building a Custom Add-on-the-fly Form (Bonus exercise 1)

Limit the add-on-the-fly form for the Corporate Account Code table so that users can only add new corporate account codes on the fly, without editing or deleting existing codes.

To restrict editing or deleting during add-on-the-fly, use **Setup / System / Menu Codes** to open the menu code that includes the add-on-the-fly menu selection..

Disable the **Allow edit** and **Allow delete** checkboxes.

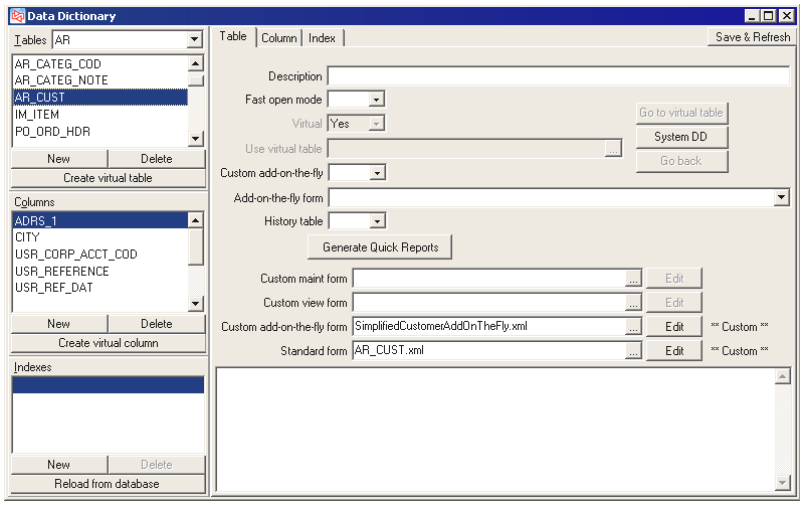


Building a Custom Add-on-the-fly Form (Bonus exercise 2)

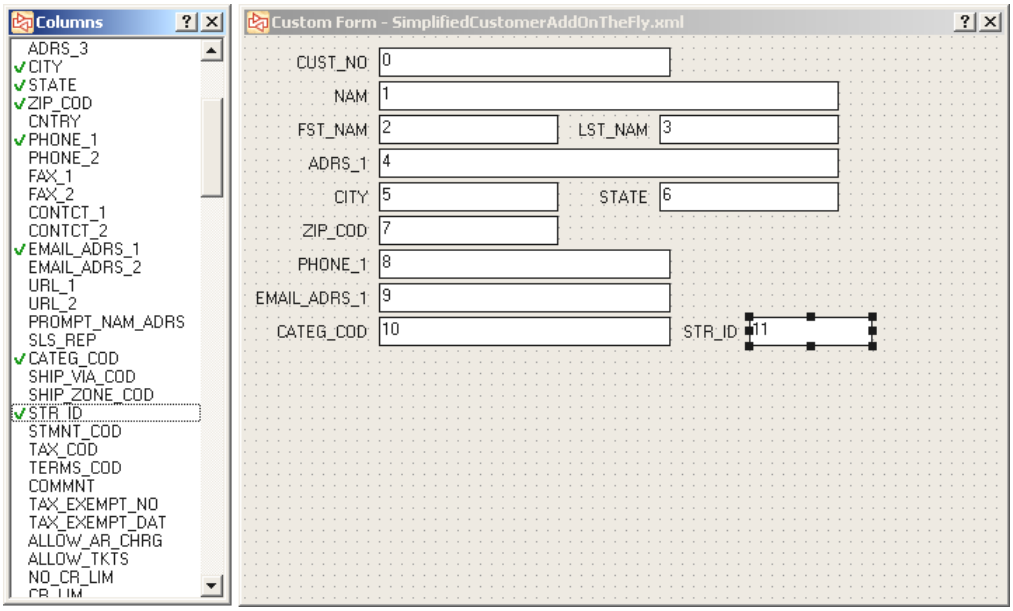
Add the column "Store" to the predefined add-on-the-fly layout form for customers supplied with Counterpoint.

In **Setup / System / Configuration / Data Dictionary**, select the AR_CUST table in the System Data Dictionary and click Customize table .

On the Table tab, at **Custom add-on-the-fly form**, browse to the System / CustomForms folder (under the TopLevel directory on the server) and select **SimplifiedCustomerAdd-OnTheFly.xml**.



Edit the layout form and add the STR_ID column.



Customizing Toolbars (Bonus exercise)

Add a button to the toolbar in the Vendor add-on-the-fly form so that the full maintenance function for Vendors can be accessed in situations where additional information needs to be specified than what is allowed on the add-on-the-fly form.

Add to Actions.xml for company

```
<action id = "Vendors" actiontype="menu">
    <filetorun>TFRMVENDORS</filetorun>
    <parameters></parameters>
    <caption>Vendors</caption>
    <image></image>
    <hint>Vendors</hint>
    <shortcut>Alt+V</shortcut>
</action>
```

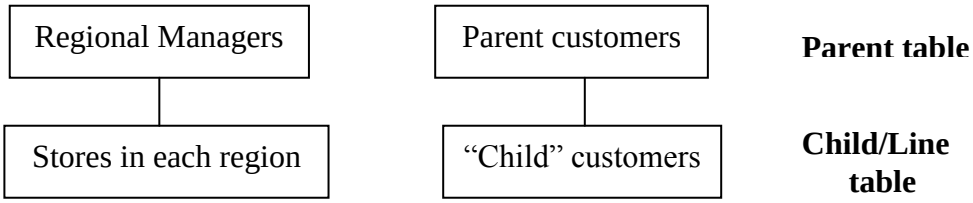
Run a menu selection where the Vendor add-on-the-fly form can be displayed (such as **Purchase Requests / Enter**). Right-click the title bar of the Vendor add-on-the-fly form, select to **Customize toolbars**, and add the new **Vendors** button to the toolbar for that form.

APPENDIX 2: Adding Custom Document Maintenance Forms

Custom Document Maintenance Form

Menu selection that allows maintenance of two related tables on a single form

Examples:



Requirements

SQL statement

- Parent and/or Child table can be a physical table or a view table
- Parent and Child tables must have the same primary key (e.g., REG_MGR_ID)
- Child table must also have SEQ_NO integer column as component of primary key
- Parent table must include LST_LCK_DT datetime column

Data Dictionary

- Define separate XML layouts for each table in same "slot" in Data Dictionary
  - Can be Custom maint form, Custom view form, or Custom add-on-the-fly form
  - Don't include primary key in XML layout for Child table
- Define separate XML layouts for each table in same "slot" in Data Dictionary
- If Child table column uses a Lookup Display column in Data Dictionary, create a virtual column named _**<ColumnName>_LU**
  - e.g., to use DESCR as Lookup display for a column named ITEM_NO in the Child table, create _ITEM_NO_LU

Adding Custom Document Maintenance Forms

1. Add the new tables

Parent table

```
Create table dbo.USR_REGIONAL_MGR
(REG_MGR_ID          T_USR_ID      not null,
 REGION_ID           T_COD         not null,
 REGION_NAM          T_NAM         null,
 REGION_INFO         T_NOTE        null,
 LST_MAINT_DT        T_DT          null,
 LST_MAINT_USR_ID    T_USR_ID      null,
 ROW_TS              timestamp     null,
constraint PK_USR_REGIONAL_MGR primary key (REG_MGR_ID, REGION_ID),
constraint FK_USR_REGIONAL_MGR_SY_USR foreign key (REG_MGR_ID) references
dbo.SY_USR (USR_ID)
)
```

Child table

```
Create table dbo.USR_REGIONAL_STR
(REG_MGR_ID          T_USR_ID      not null,
 REGION_ID           T_COD         not null,
 REG_STR_SEQ_NO      T_SEQ_NO      not null,
   constraint CK_REG_STR_SEQ_NO check (REG_STR_SEQ_NO >= 0),
 REG_STR_ID          T_COD         not null,
 LST_MAINT_DT        T_DT          null,
 LST_MAINT_USR_ID    T_USR_ID      null,
 ROW_TS              timestamp     null,
constraint PK_USR_REGIONAL_STR primary key (REG_MGR_ID, REGION_ID,
REG_STR_SEQ_NO),
constraint FK_USR_REGIONAL_STR_REGIONAL_MGR  foreign key (REG_MGR_ID, REGION_ID)
references dbo.USR_REGIONAL_MGR (REG_MGR_ID, REGION_ID)
)
```

- Primary key of child table must include all components of parent table's primary key, as well as SEQ_NO column.
- Foreign key in child table must reference both components of parent's primary key, if it's going to reference any part of the parent's primary key.

2. Create any additional indexes for the parent table

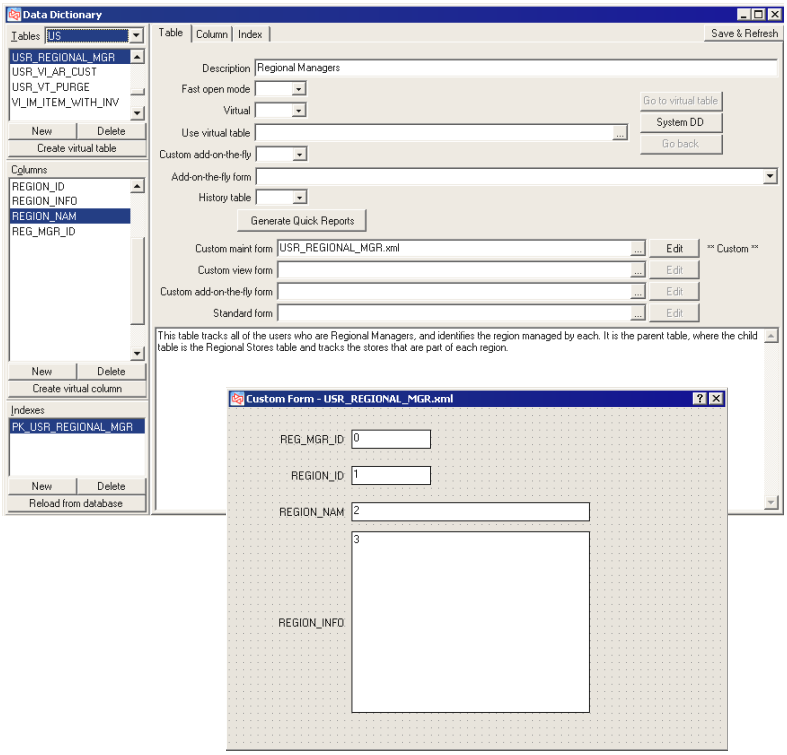
If you want to sort and look up values in the parent table using anything besides the primary key, you can create additional indexes.

3. Add Data Dictionary entries for both tables.

Parent table

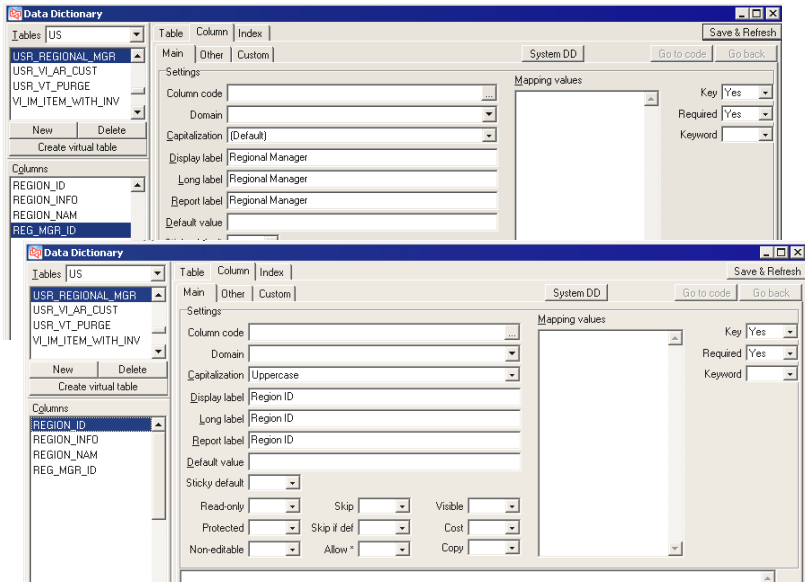
Enter a description of the parent table and identify its purpose.

Build the XML layout as a Custom maint form.

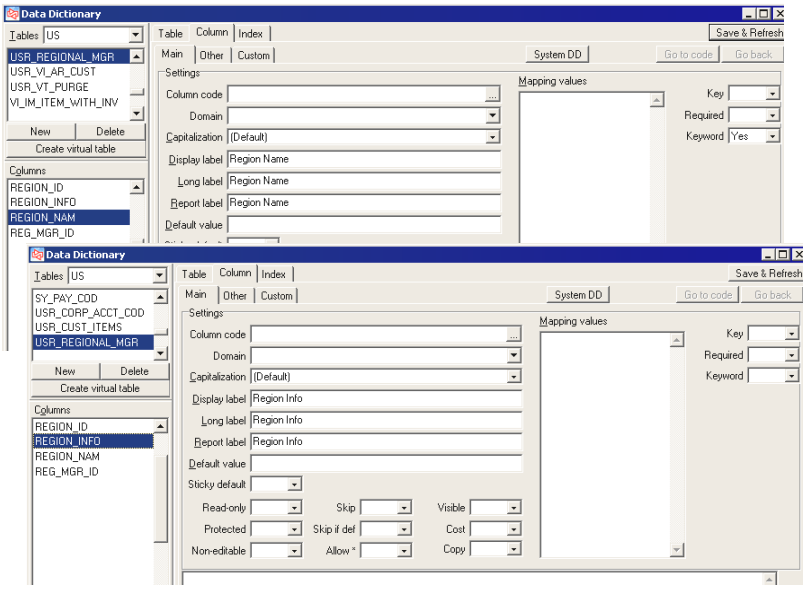


Enter display labels for all of the fields.

The two fields that make up the primary key must be set to Yes for **Key** and **Required**.

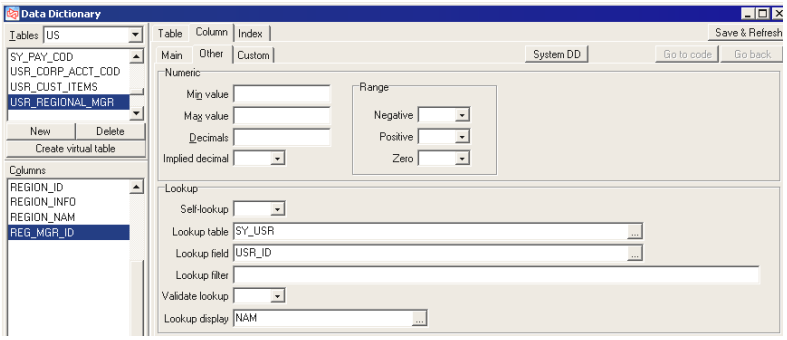


Set **Keyword** to Yes for the Region Name field so it's treated as a keyword in lookups.



The lookup for the column REG_MGR_ID is set to SY_USR and the field USR_ID in that table.

The user's name will appear next to the selected user ID.



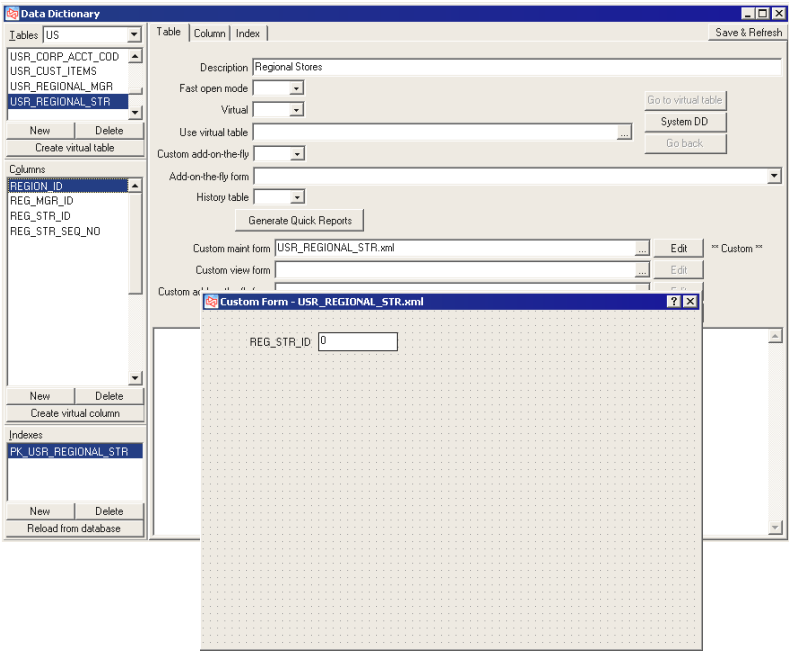
None of the other columns in the USR_REGIONAL_MGR table use lookups.

Child table

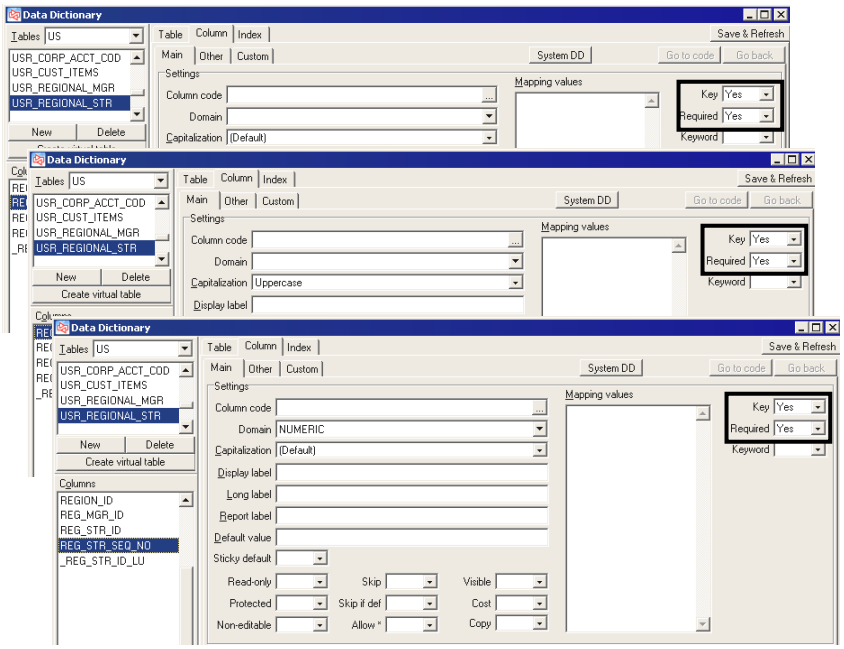
Add a description and XML layout for this table too.

The XML layout must be in the same "slot" as the parent table's format.

Notice that the XML layout does not include the primary key fields, or the SEQ_NO field.



Add all three components of the primary key for the child table (REG_MGR_ID, REGION_ID, and REG_STR_SEQ_NO) to the Data Dictionary, along with REG_STR_ID.

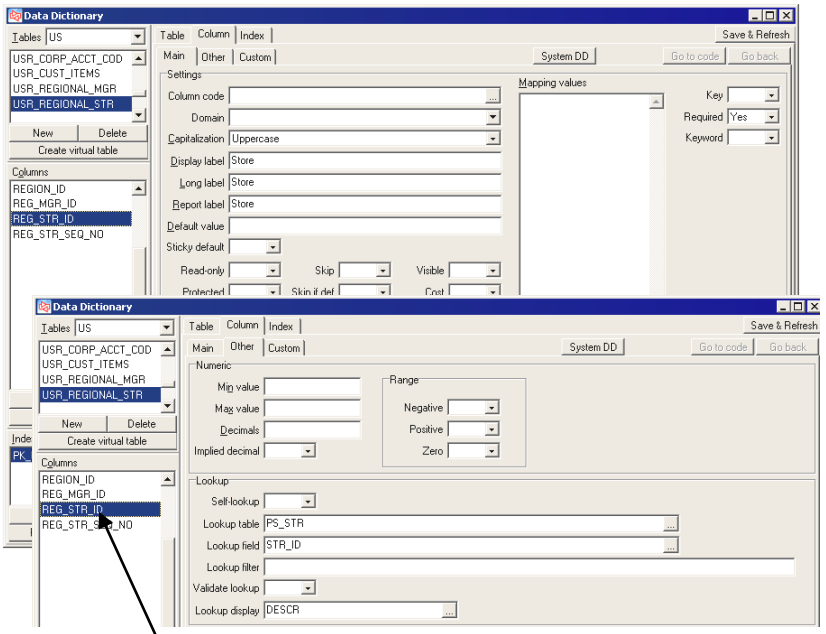


Set the three key fields to Yes for **Key** and **Required**. You can optionally set them to No for **Visible**, to avoid having them appear in the line item grid of the maintenance form. None of these fields will need a display label.

Enter a display label for REG_STR_ID and set **Required** to Yes, since each region needs at least one store.

Set the **Lookup table** for REG_STR_ID to PS_STR and the **Lookup field** STR_ID.

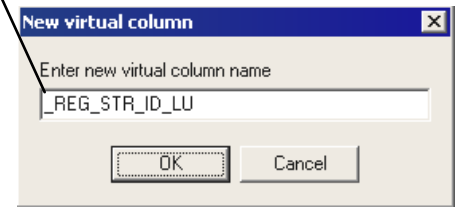
Set the **Lookup display** to DESCR so that the store's description appears next to each selected store, but for this to work, you also need to create a virtual column.



The name of the virtual column must be _**<ColumnName>_LU**, (_**REG_STR_ID_LU** for this example).

Enter a display label for this virtual column as well ("Store Name" for this example).

Name is associated with Column name, not Lookup field



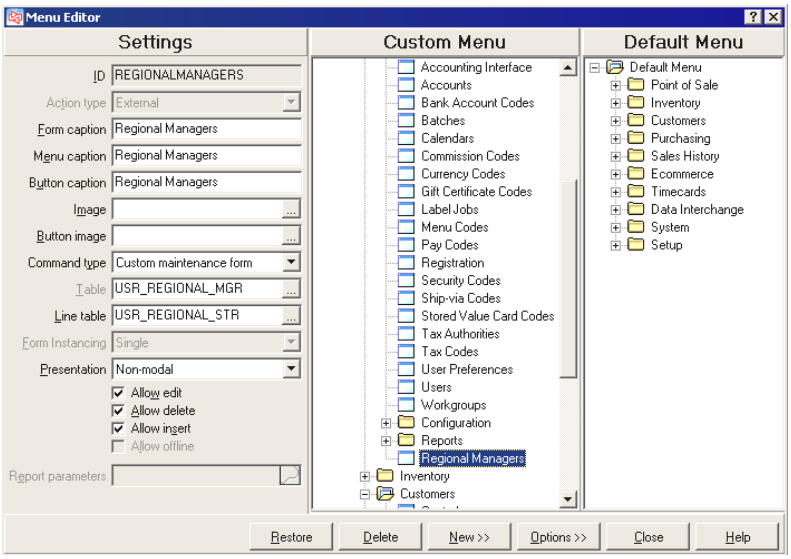
4. Add menu selection to Counterpoint menu.

Use **Setup / System / Menu Codes** to add the new menu selection.

Be sure that your selection for **Command type** matches the "slot" where you specified the XML layout in the Data Dictionary.

For **Table**, specify the parent table.

For **Line table**, specify the child table.

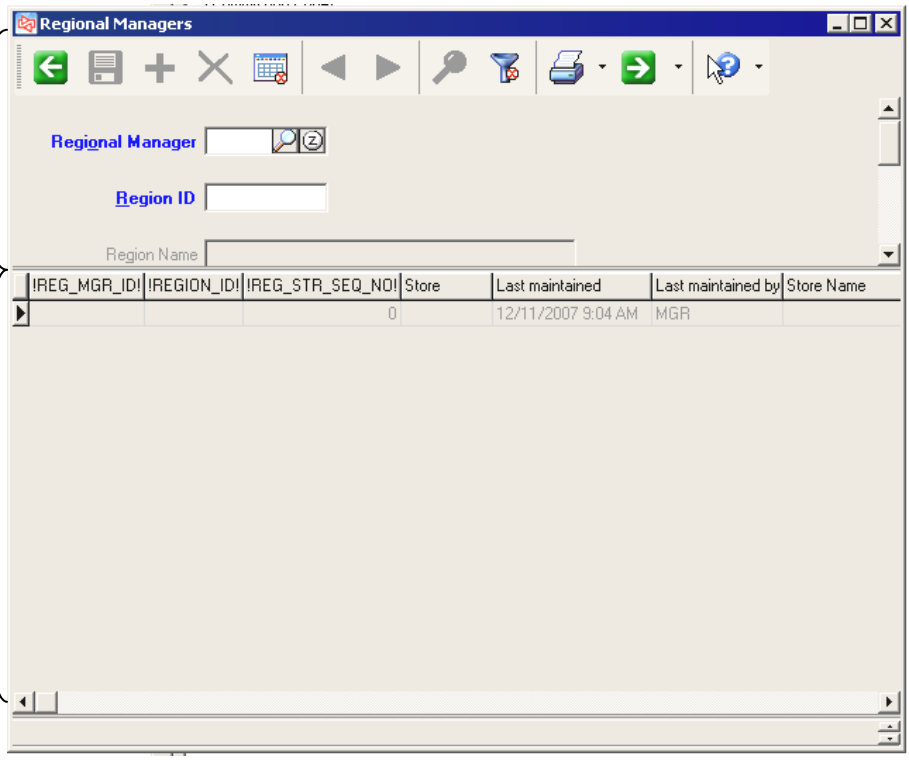


Run the menu selection and edit the line item grid.

You will need to access the menu selection and edit the line item grid before releasing it to your users.

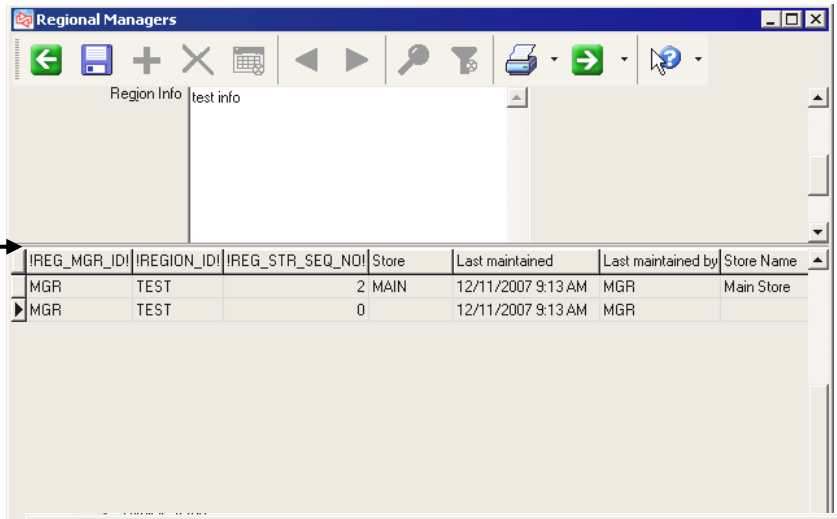
Parent area

Child area



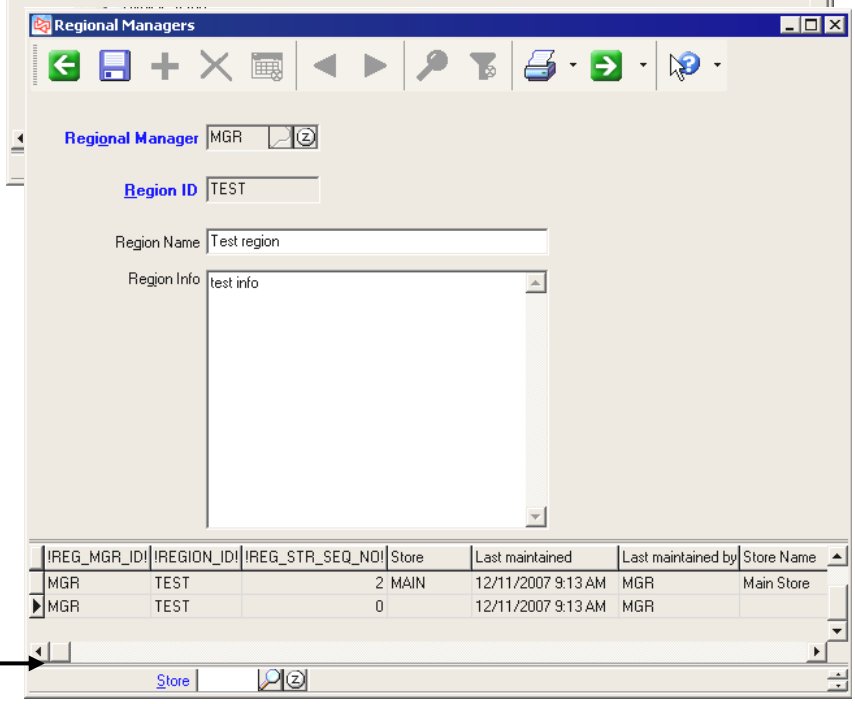
To enable all of the editing controls, enter a Regional Manager with at least one store.

Drag the splitter down between the parent and child areas until you see all of the parent fields and the scroll bar along the right disappears.

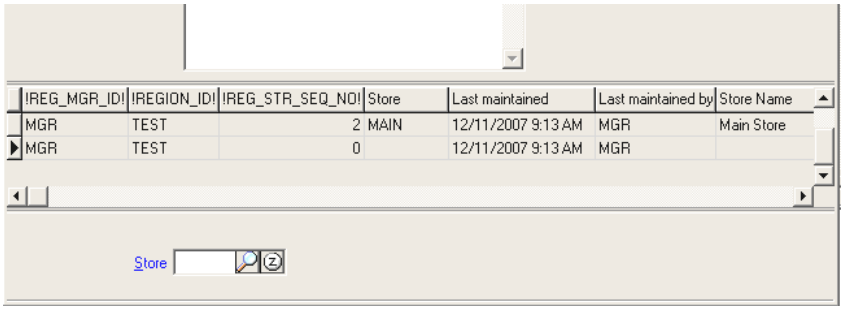


Drag the lower right corner down to resize the form until you are satisfied with the position of the **Store** entry field.

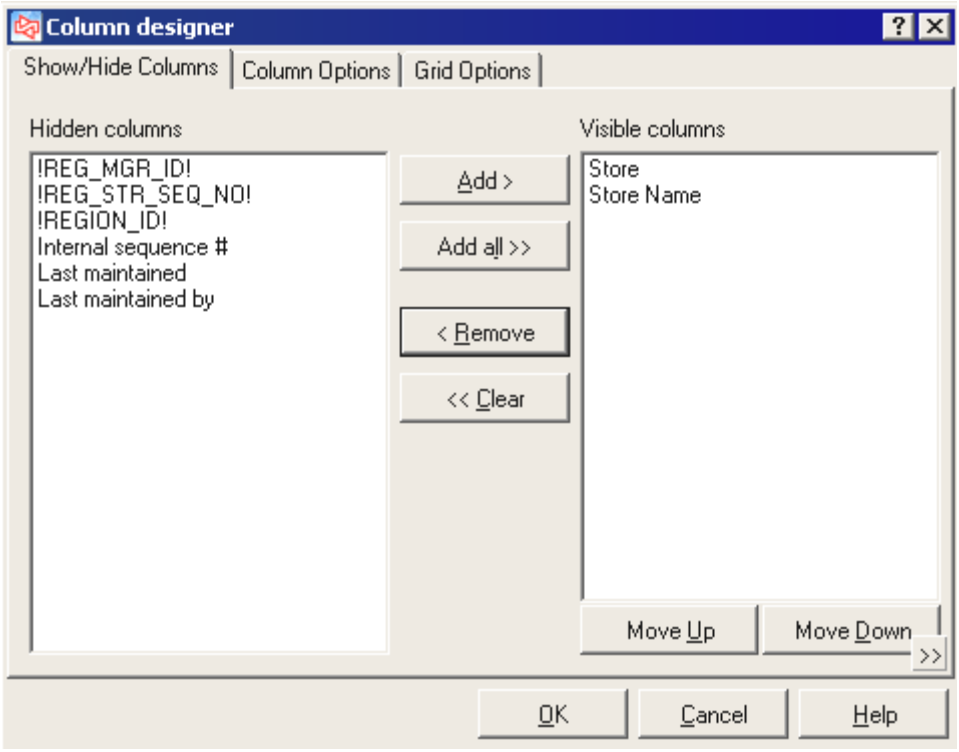
Then drag the splitter up between the line item grid and the **Store** entry field.



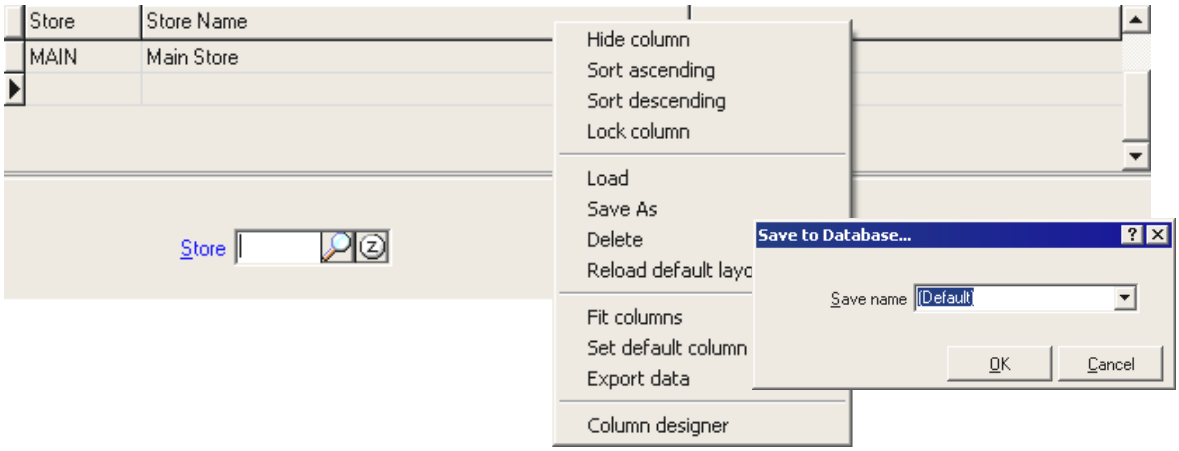
You want the child area to look similar to this when you are finished.



Now use the Column Designer to eliminate the child fields that don't need to appear in the line item grid.



When you're finished, save the modified line item grid as the default.



Your custom maintenance document form is now ready to be used!

Adding Custom Document Maintenance Forms

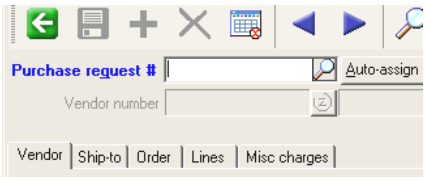
Try It Yourself!

In this exercise, you will build the Regional Managers and Regional Stores tables and create the menu selection that brings the two tables together to allow the maintenance of the list of stores that are managed by each regional manager.

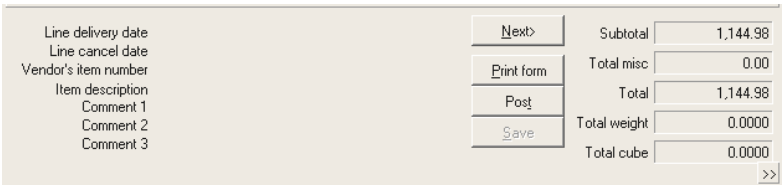
Use the preceding pages for assistance.

Limitations of Custom Document Maintenance Forms

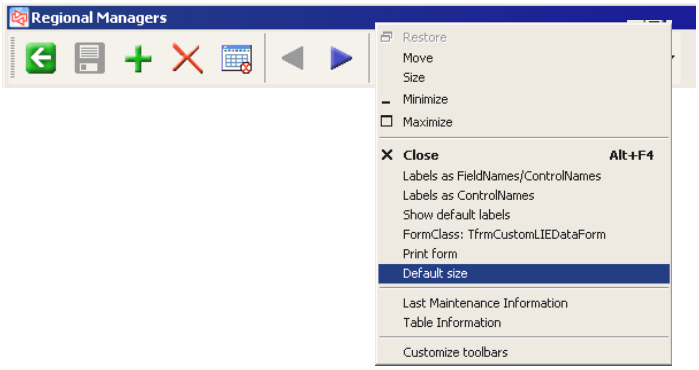
- No additional tabs for parent fields



- No totals display area



- Cannot require certain parent fields to be entered prior to entry of lines
- Cannot require that parent have at least one line item when saved
- "Default size" option on form control menu doesn't set form to any reasonable size

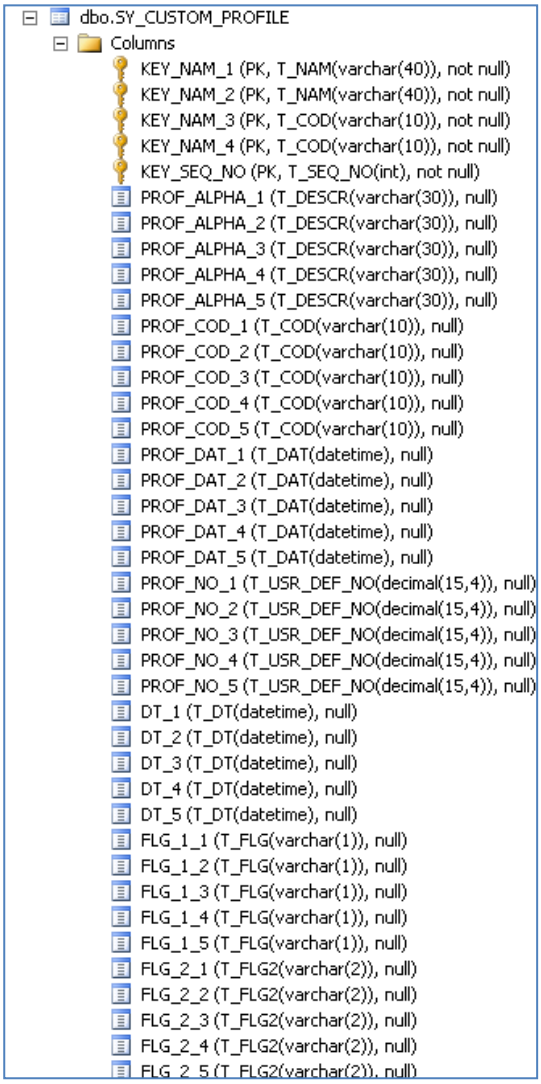


- Two users can edit the same document at the same time. Changes to different parent fields will be saved successfully, but changes to the same or different line items will not. Line item changes saved last will "win" and no warning or error messages will be issued.
- Honors user preference to start maintenance forms (not document forms) in Table View.

APPENDIX 3: Using a View of SY_CUSTOM_PROFILE

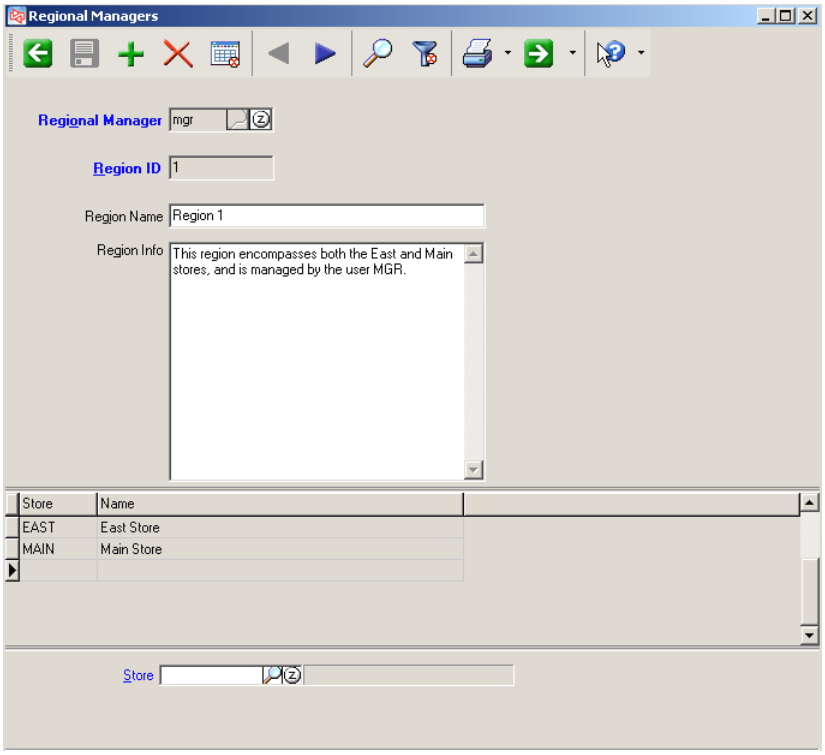
SY_CUSTOM_PROFILE

- Standard table in CPSQL database
- Contains 5 key columns and many other columns
- Use for customizations that would normally require a new table be defined
- Not appropriate for extensive storage (e.g., transaction history)
- Create views against table to build data dictionary entries, maintenance forms, and quick reports
- Already included in replication rules and Offline Ticket Entry configuration data



Building a Document Maintenance Form Using Views of SY_CUSTOM_PROFILE

In a previous exercise, you built a custom document maintenance form by creating parent and child tables to track regional managers and the stores in each region.



An alternate way to produce the custom document maintenance form could be done by creating a header view and line view against the SY_CUSTOM_PROFILE table.

Building a Document Maintenance Form Using Views of SY_CUSTOM_PROFILE

1) Build the views of SY_CUSTOM_PROFILE

Parent view (Region header)

```
create view dbo.USER_VI_SLS_REGION_HDR
as
select
  KEY_NAM_1,                                -- Always 'SLS-REGION'
  KEY_NAM_2,                                -- Always 'SLS-REGION'
  KEY_NAM_3,                                -- Always 'HEADER'
  KEY_NAM_4      as SLS_REGION_ID,          -- Sales region ID
  KEY_SEQ_NO,                               -- Always 0 for header
  ADDL_DESCR_1   as DESCR,                  -- Description
  USR_ID as REG_MGR_ID,                     -- Regional manager ID
  LST_MAINT_DT,
  LST_MAINT_USR_ID,
  LST_LCK_DT,
  ROW_TS
from SY_CUSTOM_PROFILE
where KEY_NAM_1  = 'SLS-REGION'
  and KEY_NAM_2  = 'SLS-REGION'
  and KEY_NAM_3  = 'HEADER'
  and KEY_SEQ_NO = 0
```

Child view (Region line)

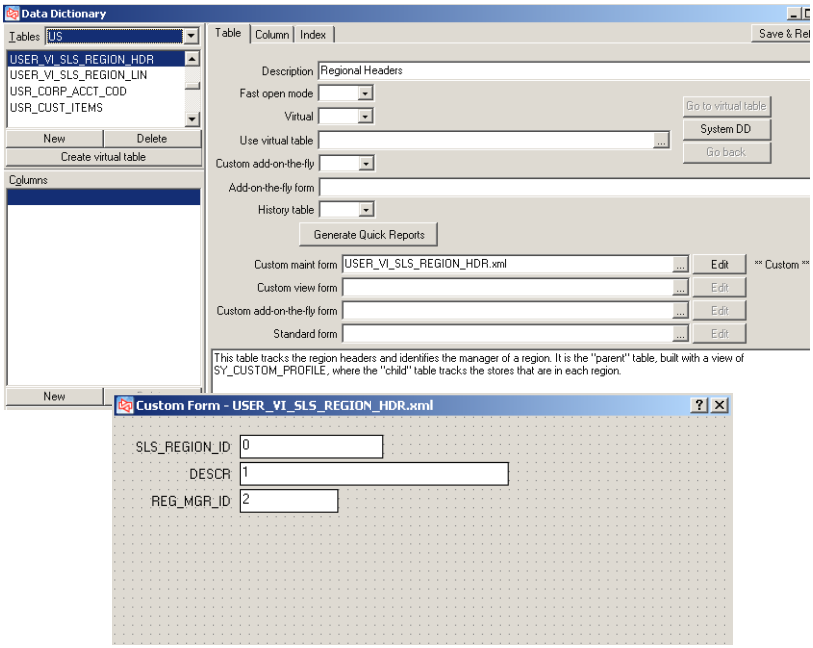
```
create view dbo.USER_VI_SLS_REGION_LIN
as
select
  KEY_NAM_1,                                -- Always 'SLS-REGION'
  KEY_NAM_2,                                -- Always 'SLS-REGION'
  KEY_NAM_3,                                -- Always 'LINE'
  KEY_NAM_4      as SLS_REGION_ID,          -- Sales region ID
  KEY_SEQ_NO     as SEQ_NO,                 -- Sequence number
  PROF_COD_1     as STR_ID,                 -- Store ID
  LST_MAINT_DT,
  LST_MAINT_USR_ID,
  LST_LCK_DT,
  ROW_TS
from SY_CUSTOM_PROFILE
where KEY_NAM_1  = 'SLS-REGION'
  and KEY_NAM_2  = 'SLS-REGION'
  and KEY_NAM_3  = 'LINE'
```

Building a Document Maintenance Form Using Views of SY_CUSTOM_PROFILE

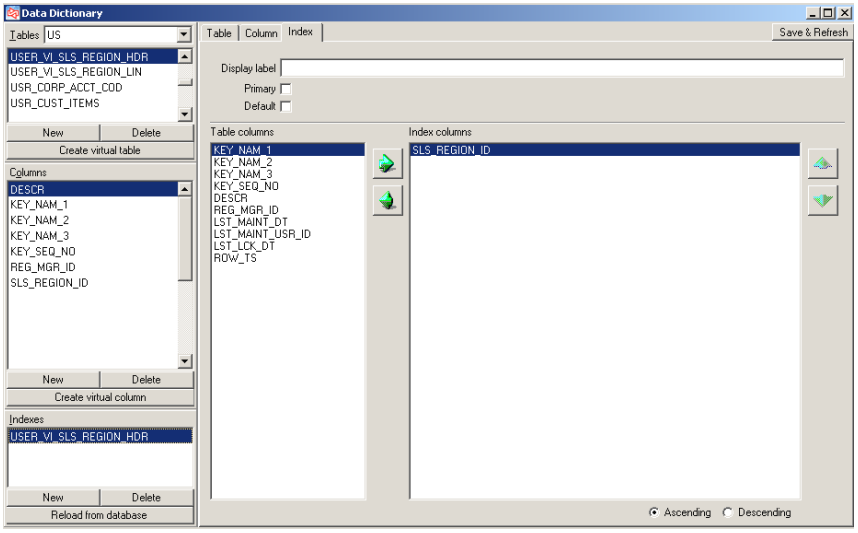
2) Add Data Dictionary entries

Parent view table

- *Enter a description of the Parent view table and build an XML layout for it as a Custom maint form.*

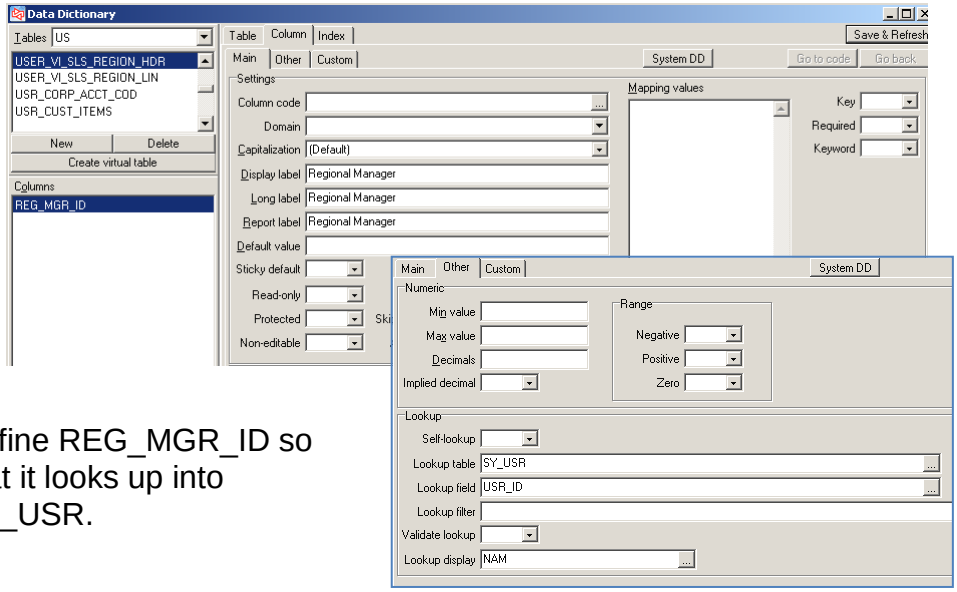


- *On the Index tab, add a logical index for the view table.*



➢ *Enter display labels for these columns, since they will be visible on the maintenance form:*

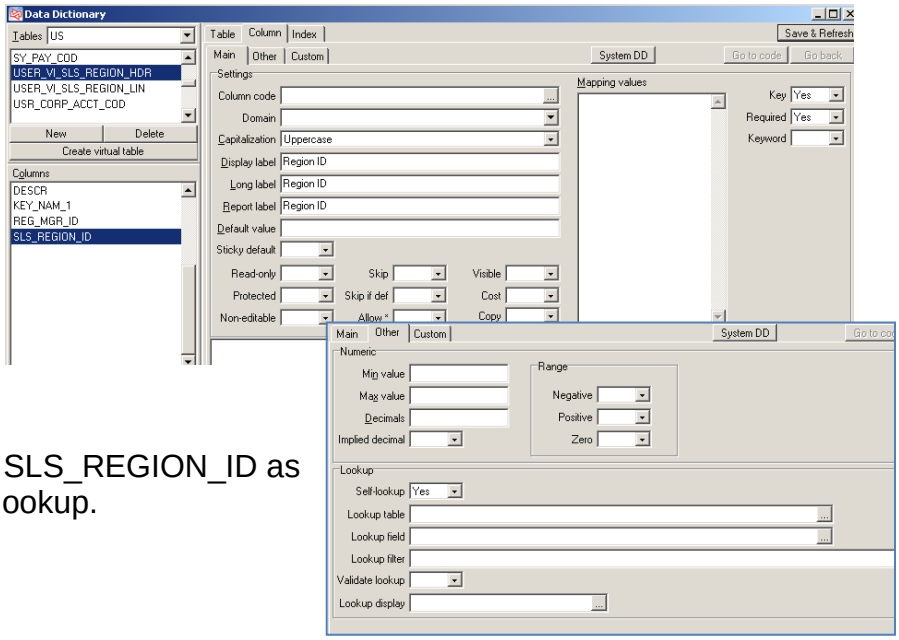
REG_MGR_ID



Define REG_MGR_ID so that it looks up into SY_USR.

SLS_REGION_ID

This column is also marked Yes to both “Key” and “Required”.

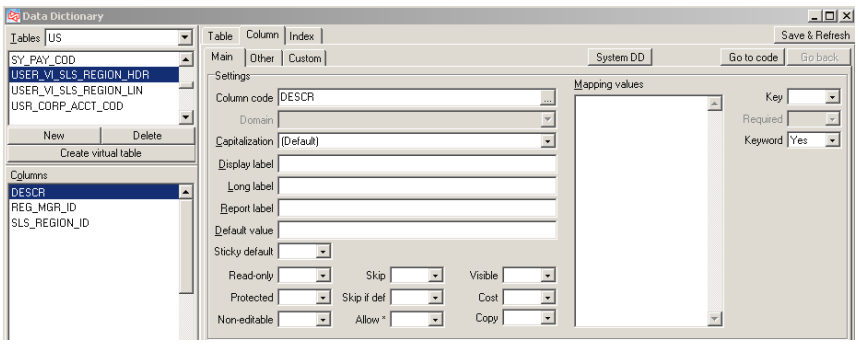


Define SLS_REGION_ID as a Self-lookup.

DESCR

The column code DESCR will provide the display label.

The column is also marked Yes to “Keyword”.

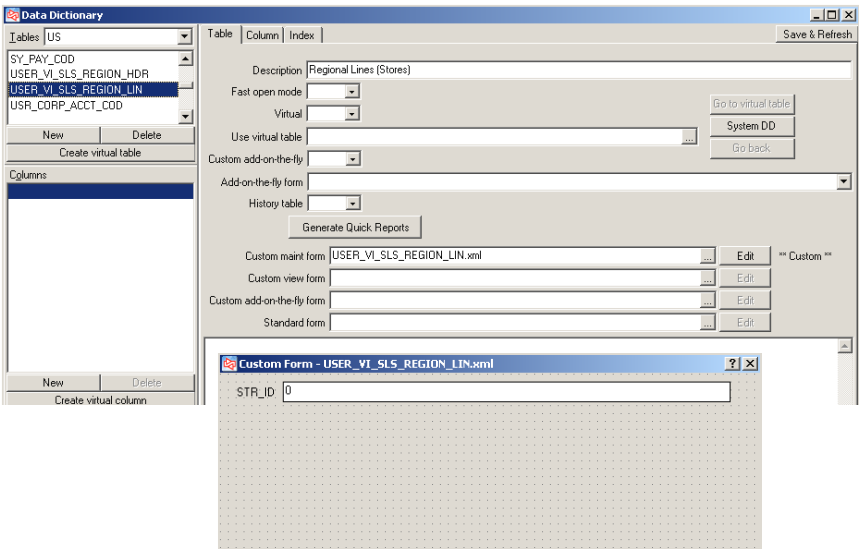


- *Add defaults for these columns, since the columns are keys and do not appear on the maintenance form. Do **not** mark them as "Key" or "Required":*

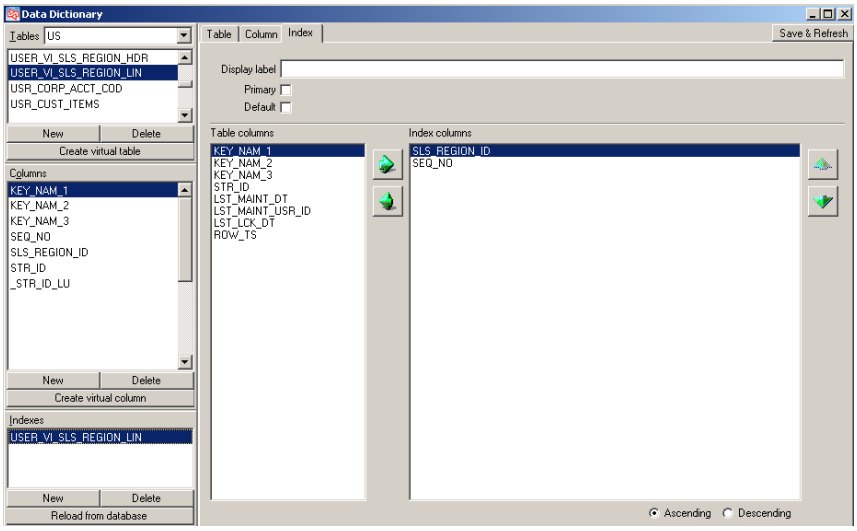
Column	Default Value
KEY_NAM_1	SLS-REGION
KEY_NAM_2	SLS-REGION
KEY_NAM_3	HEADER
KEY_SEQ_NO	0

Child view table

- *Enter a description of the Child view table and build an XML layout for it as a Custom maint form.*

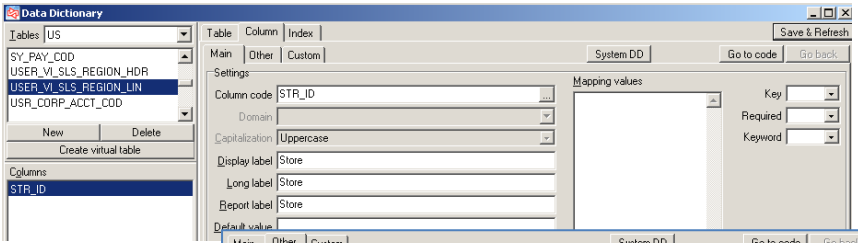


- *On the Index tab, add a logical index for the view table.*

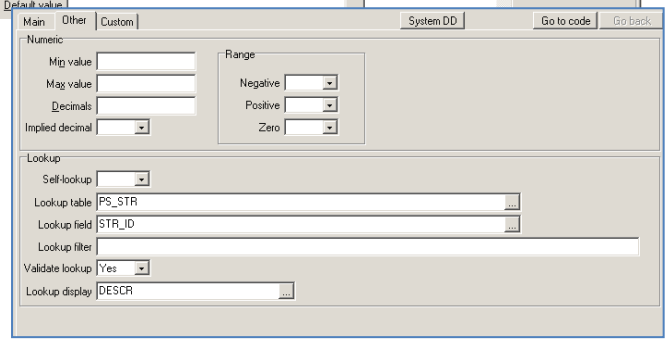


➢ *Enter a display label for this column, since it will be visible on the maintenance form:*

STR_ID

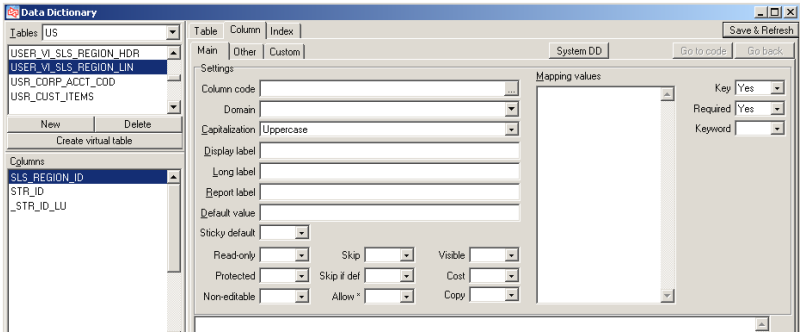


Define STR_ID so that it looks up into PS_STR.



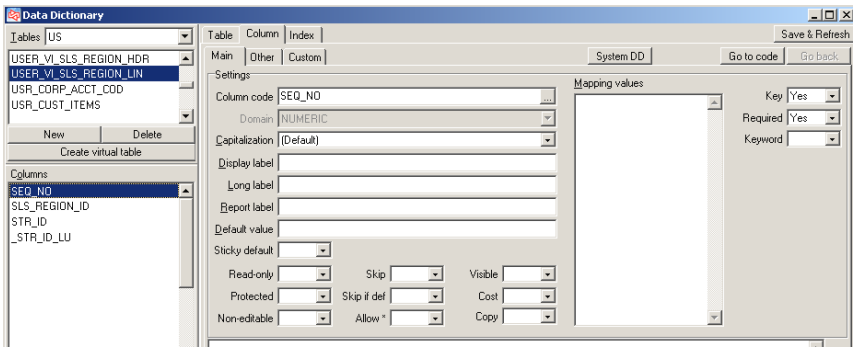
➢ *This column will not require a display label, but must be marked Yes to both "Key" and "Required" so that it matches the dictionary entry for the parent view table:*

SLS_REGION_ID



➢ *This column will not require a display label either, but must be marked Yes to both "Key" and "Required" since the column is a part of the key.*

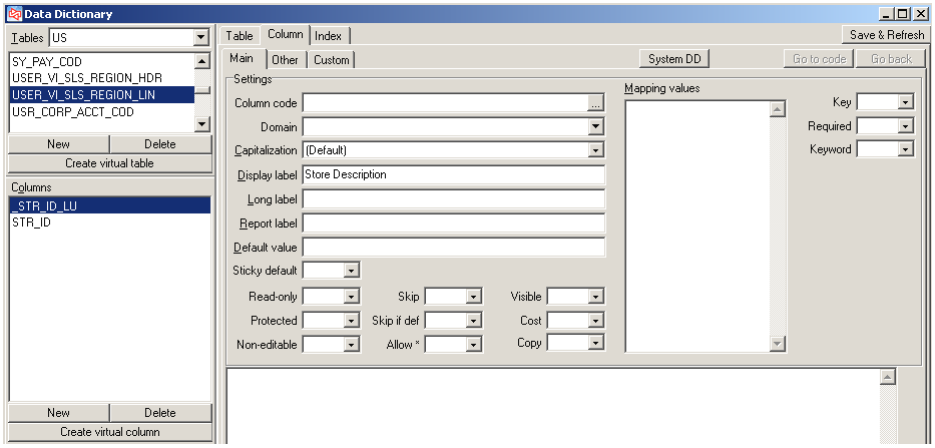
SEQ_NO



- *Add defaults for these columns, since the columns are keys and do not appear on the maintenance for. Do **not** mark them as "Key" or "Required":*

Column	Default Value
KEY_NAM_1	SLS-REGION
KEY_NAM_2	SLS-REGION
KEY_NAM_3	LINE

- *Create a virtual column named _STR_ID_LU so that a column heading will appear for the DESCR lookup display value that appears for the store ID:*

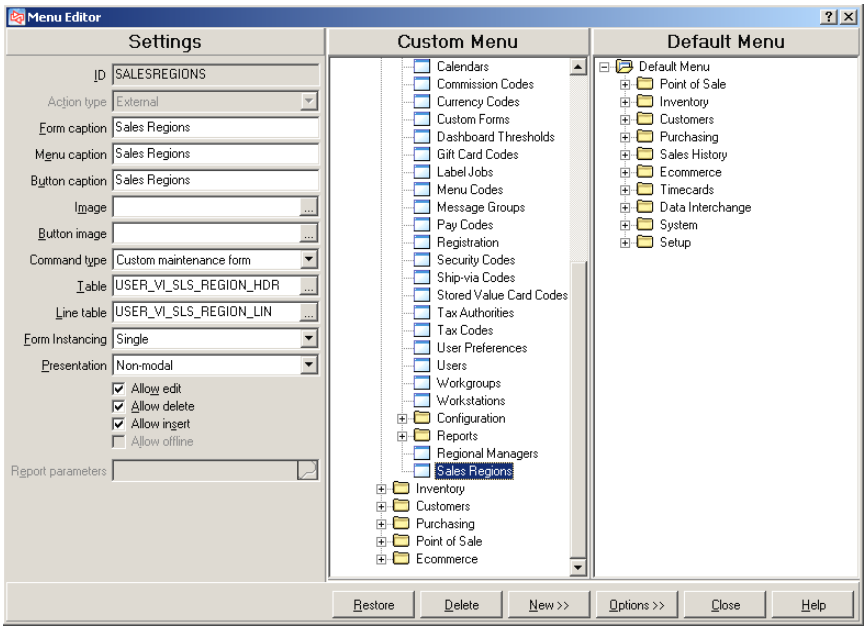


3) Add menu selection to Counterpoint menu

Use **Setup / System / Menu Codes** to add the new menu selection.

For **Table**, specify the parent view table.

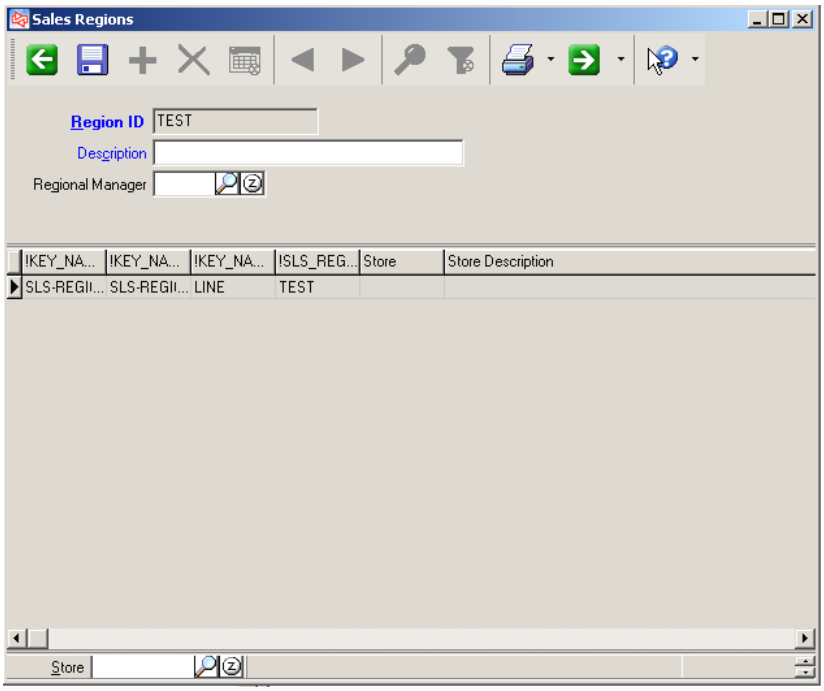
For **Line table**, specify the child view table.



Run the menu selection and edit the line item grid.

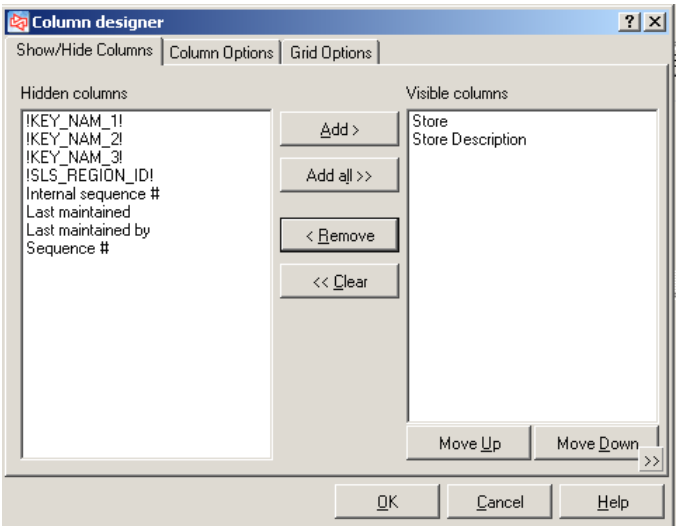
Access the menu selection and edit the line item grid before releasing it to your users.

1) Enter a Region ID so that the editing controls are enabled.

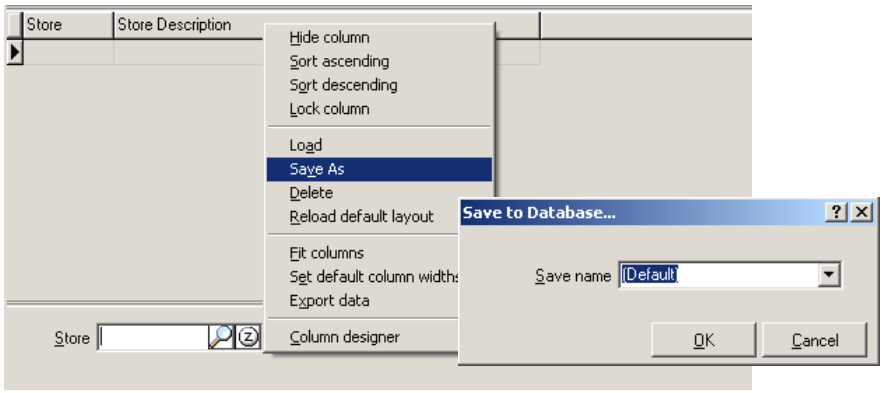


2) Drag the splitter up between the line item grid and the **Store** entry field.

3) Use Column Designer to hide all child fields except **Store** and **Store Description**.



When you finish, save the modified line item grid as the default.



The form is now ready to be used.

APPENDIX 4: Removing Customizations from a Databasse

To disable customizations in a database

- In Counterpoint, select **Setup>System>User Preferences**
- Switch to the User Interface tab
- Clear the Enable Customizations check box

To remove customizations in a database

- In Counterpoint, select **System>Utilities>Database Customizations Report**
- Run the report to determine the names of:
  - customized stored procedures
  - customized triggers
  - custom tables
  - custom columns in standard tables
- Use SQL Server Management Studio to:
  - drop the customized stored procedures and triggers

    ```
    Drop Procedure <StoredProcedureName>
    Drop Trigger <TriggerName>
    ```

  - set the ACTIVE column to “N” for the records in SY_CUSTOMIZATION_EVENT_HANDLER

    ```
    Update SY_CUSTOMIZATION_EVENT_HANDLER
    Set ACTIVE=’N’
    Where EVENT_ID=###

    Update SY_CUSTOMIZATION_EVENT
    Set ACTIVE=’N’
    Where EVENT_ID=###
    ```

    Set the value of EVENT_ID to the number associated with the stored procedure that you have dropped

- drop custom tables

  Drop Table <tablename>

- drop custom columns in standard tables

  Alter Table <tablename>
  Drop Column <columnname>

Specialty Retail
6060 Primacy Parkway, Suite 460 ▫ Memphis, TN 38119
(800) 852-5852 ▫ www.CounterPointPOS.com