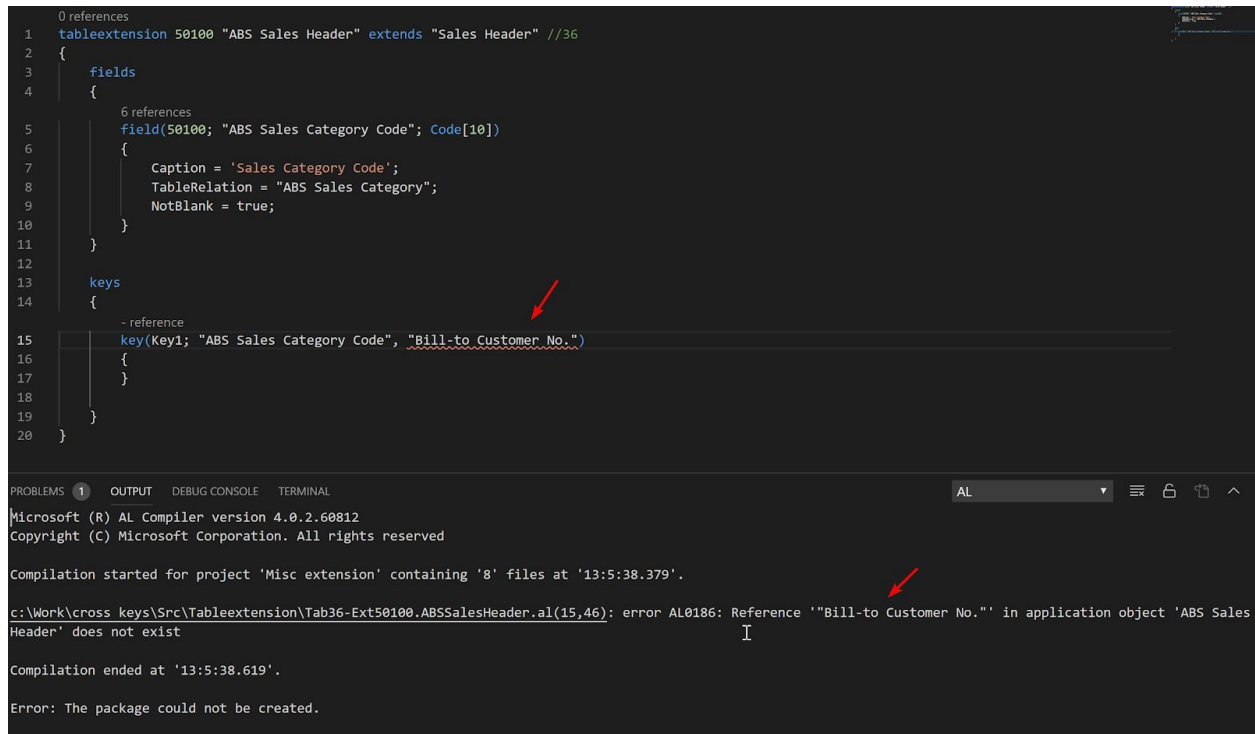


Una solución alternativa para las llaves cruzadas en extensiones de tablas en AL

Algo importante que se debe saber cuándo se está desarrollando una extensión de una tabla en AL, es que se pueden crear una o más llaves, pero el alcance de esa llave es solamente la tabla extendida, no se puede hacer referencia a los campos de la tabla que se está extendiendo.



```
0 references
1 tableextension 50100 "ABS Sales Header" extends "Sales Header" //36
2 {
3     fields
4     {
5         6 references
6         field(50100; "ABS Sales Category Code"; Code[10])
7         {
8             Caption = 'Sales Category Code';
9             TableRelation = "ABS Sales Category";
10            NotBlank = true;
11        }
12    }
13    keys
14    {
15        - reference
16        key(Key1; "ABS Sales Category Code", "Bill-to Customer No.")
17        {
18        }
19    }
20 }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

Microsoft (R) AL Compiler version 4.0.2.60812
Copyright (C) Microsoft Corporation. All rights reserved.

Compilation started for project 'Misc extension' containing '8' files at '13:5:38.379'.

c:\Work\cross keys\Src\Tableextension\Tab36-Ext50100.ABSSalesHeader.al(15,46): error AL0186: Reference ''Bill-to Customer No.'' in application object 'ABS Sales Header' does not exist

Compilation ended at '13:5:38.619'.

Error: The package could not be created.

La llave primaria está implícitamente incluida al final de cualquier llave que se cree en la tabla extendida. Vamos a probarlo:

He creado una extensión de la tabla Sales Header con un nuevo campo llamado **Sales Category Code**, he creado también una página tipo lista ordenada por este nuevo campo, si mi punto es correcto, entonces esta página va a mostrar los datos ordenados por **Sales Category Code + Document Type, Document No. (Llave primaria)**.

```

0 references
1 page 50102 "ABS Sales Order List"
2 {
3     PageType = List;
4     ApplicationArea = All;
5     UsageCategory = Lists;
6     SourceTable = "sales header";
7     Caption = 'Sales Order List';
8     SourceTableView = sorting("ABS Sales Category Code");
9
10    layout
11    {
12        0 references
13        area(Content)
14        {
15            0 references
16            repeater(Control1)
17            {
18                0 references
19                field("ABS Sales Category Code"; "ABS Sales Category Code")
20                {
21                    ApplicationArea = All;
22                }
23                0 references
24                field("Document Type"; "Document Type")
25                {
26                    ApplicationArea = All;
27                }
28                0 references
29                field("No."; "No.")
30                {
31                    ApplicationArea = All;
32                }
33            }
34        }
35    }
36 }

```

Como Podemos ver, probamos que la llave primaria es adicional al final de nuestras llaves extendidas.

CRONUS USA, Inc. | Finance | Cash Management | Sales | Purchasing | Setup & Extensions | Intelligent Cloud Insights

Sales Order List: Search + New Manage Open in Excel

SALES CATEGORY CODE	DOCUMENT TYPE	NO.
	Quote	S-QUO1001
	Quote	S-QUO1002
	Order	S-ORD101005
	Invoice	S-INV102199
	Invoice	S-INV102200
	Invoice	S-INV102201
	Invoice	S-INV102202
	Invoice	S-INV102203
	Invoice	S-INV102204
	Invoice	S-INV102205
A	Order	S-ORD101001
B	Order	S-ORD101003
C	Order	S-ORD101002
C	Order	S-ORD101004

Ahora el problema que debemos resolver de alguna manera es cómo incluir campos que no sean de la llave primaria o incluso campos de la llave primaria, pero en una posición diferente, como por ejemplo al principio de nuestra llave extendida.

Parece que la única solución por el momento es replicar los campos que queremos usar, en nuestra tabla extendida, usando un nombre diferente y creando subscripciones a los eventos de Inserción y modificación para sincronizar esos campos estándar con nuestros campos extendidos. Esto puede ser muy tedioso dependiendo del campo que queramos replicar.

Tratemos entonces de buscar una solución alternativa, que si bien no es la mejor, puede ayudarnos a sortear algunos de los problemas que podamos tener al respecto.

Qué tal usar Consultas? En los objetos tipo consulta se pueden ordenar los datos de la manera que deseamos.

Permítanme explicar la idea con un ejemplo, usando el nuevo campo que creamos en la extensión de la tabla Sales Header.

```

0 references
1 tableextension 50100 "ABS Sales Header" extends "Sales Header" //36
2 {
3   fields
4   {
5     6 references
6     field(50100; "ABS Sales Category Code"; Code[10])
7     {
8       Caption = 'Sales Category Code';
9       TableRelation = "ABS Sales Category";
10      NotBlank = true;
11    }
12  }
13  keys
14  {
15    - reference
16    key(Key1; "ABS Sales Category Code")
17    {
18    }
19  }
20 }

```

Digamos que tenemos la siguiente lista de órdenes de venta con el nuevo campo en ella y que debemos ordenar estos datos por **Sales Category Code + Sell-To Customer No.**

Sales Orders: Search + New Delete Report Order Release Posting Print/Send Navigate Open in Excel Actions Navigate Report													
NO.	SELL-TO CUSTOMER NO.	SELL-TO CUSTOMER NAME	EXTERNAL DOCUMENT NO.	LOCATION CODE	ASSIGNED USER ID	DOCUMENT DATE	STATUS	COMP SHIP...	AMOUNT SHIPPED NOT INVOICED (\$)	AMOUNT SHIPPED NOT INVOICED (\$) INCL. TAX	AMOUNT	AMOUNT INCLUDING TAX	SALES CATEGORY CODE
S-ORD101001	10000	Adatum Corporation				4/2/2019	Open	No	0.00	0.00	16,767.60	17,773.66	A
S-ORD101002	10000	Adatum Corporation				5/1/2019	Open	No	0.00	0.00	2,285.30	2,422.42	C
S-ORD101003	30000	School of Fine Art				4/22/2019	Open	No	0.00	0.00	5,182.40	5,545.17	B
S-ORD101004	40000	Alpine Ski House				5/13/2019	Open	No	0.00	0.00	570.30	610.22	C
S-ORD101005	40000	Alpine Ski House				4/8/2019	Open	No	0.00	0.00	190.10	203.41	C
S-ORD101006	50000	Relecloud				4/8/2019	Open	No	0.00	0.00	190.10	201.51	A

No podemos adicionar una llave que contenga el campo **Sell-To Customer No.**, por lo tanto, vamos a crear una consulta que ordene de la manera que necesitamos.

Aquí está la consulta ordenada de la manera que queremos e incluso podemos agrupar e incluir subtotales.

```
1 reference
1 query 50100 "ABS Sales Orders by Category"
2 {
3   QueryType = Normal;
4   Caption = 'Sales Orders by Category';
5   OrderBy = ascending(ABS_Sales_Category_Code, Sell_to_Customer_No_);
6
7   elements
8   {
9     0 references
10    dataitem(Sales_Header; "Sales Header")
11    {
12      DataItemTableFilter = "Document Type" = const(Order);
13      2 references
14      column(ABS_Sales_Category_Code; "ABS Sales Category Code")
15      {
16        2 references
17        column(Sell_to_Customer_No_; "Sell-to Customer No.")
18        {
19          1 reference
20          column(Amount; Amount)
21          {
22            Method = Sum;
23          }
24        }
25      }
26    }
27  }
```

Podemos crear todas las consultas necesarias para emular las llaves que no tenemos, mientras estas llaves las necesitemos solo para efectos de ordenamiento de los datos.

Para nuestro ejemplo, voy a crear una página para mostrar los datos de la consulta.

```

1 reference
1  page 50101 "ABS Sales Orders by Category"
2  {
3      PageType = List;
4      ApplicationArea = All;
5      UsageCategory = Lists;
6      SourceTable = Integer;
7      Caption = 'Sales Orders by Category';
8
9      layout
10     {
11         0 references
12         area(Content)
13         {
14             0 references
15             repeater(Control100)
16             {
17                 0 references
18                 field(SalesCategoryCode; SalesOrderByCategoryQuery.ABS_Sales_Category_Code)
19                 {
20                     ApplicationArea = All;
21                 }
22                 0 references
23                 field(CustomerNo; SalesOrderByCategoryQuery.Sell_to_Customer_No_)
24                 {
25                     ApplicationArea = All;
26                 }
27                 0 references
28                 field(Amount; SalesOrderByCategoryQuery.Amount)
29                 {
30                     ApplicationArea = All;
31                 }
32             }
33         }
34     }
35 }

```

```

31 trigger OnOpenPage()
32 begin
33     SalesOrderByCategoryQuery.Open();
34     while SalesOrderByCategoryQuery.Read() do
35         NumberOfRows := NumberOfRows + 1;
36     end;
37     SalesOrderByCategoryQuery.Close();
38     SalesOrderByCategoryQuery.Open();
39     SetRange(Number, 1, NumberOfRows);
40 end;
41
42 trigger OnClosePage()
43 begin
44     SalesOrderByCategoryQuery.Close();
45 end;
46
47 trigger OnAfterGetRecord()
48 begin
49     If Not SalesOrderByCategoryQuery.Read() then
50         exit;
51 end;
52
53 var
54     9 references
55     SalesOrderByCategoryQuery: Query "ABS Sales Orders by Category";
56     3 references
57     NumberOfRows: Integer;
58 }

```

Y el resultado es una lista ordenada por **Sales Category Code + Sell-To Customer No.** tal como se esperaba, incluso tenemos un subtotal calculado en la última fila.

←

SALES ORDERS BY CATEGORY | WORK DATE: 4/8/2019

✓ SAVED

Search

New

Edit List

Delete

Open in Excel

SALESCATEGORYCODE		CUSTOMERNO	AMOUNT
A	:	10000	16,767.60
A		50000	190.10
B		30000	5,182.40
C		10000	2,285.30
C		40000	760.40