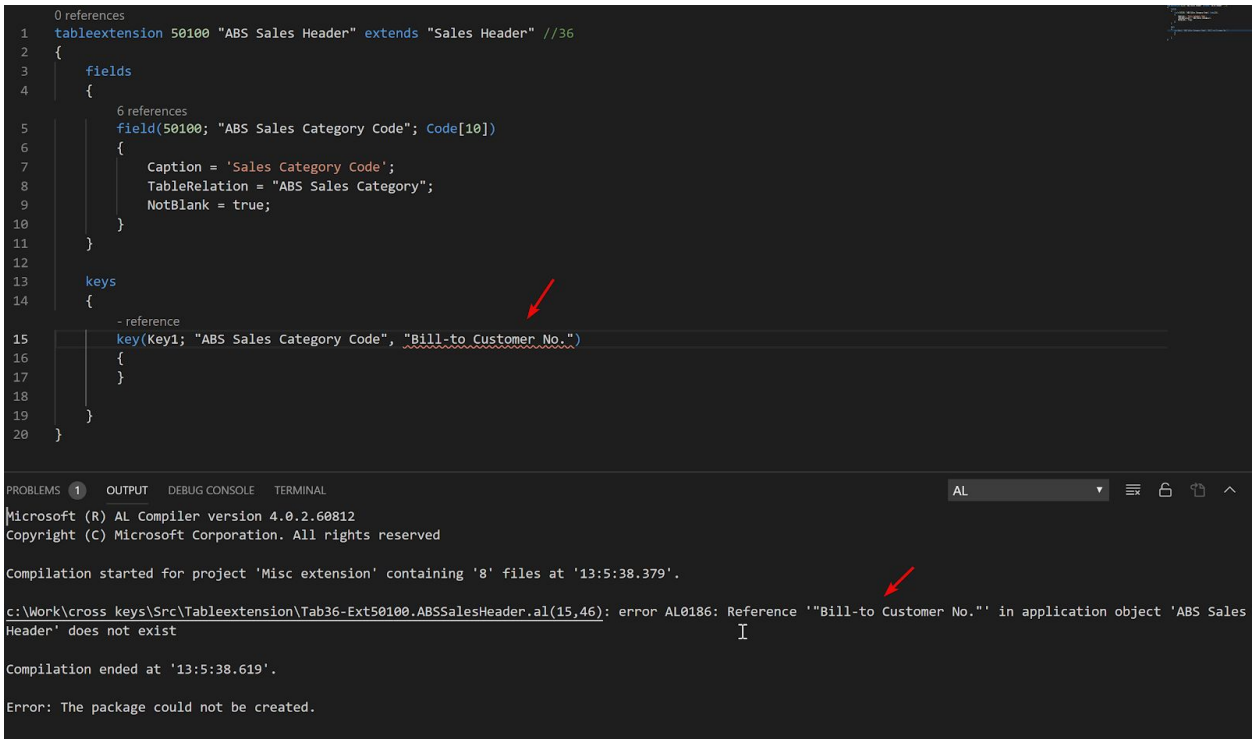
Una solución alternativa para las llaves cruzadas en extensiones de tablas en AL

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.

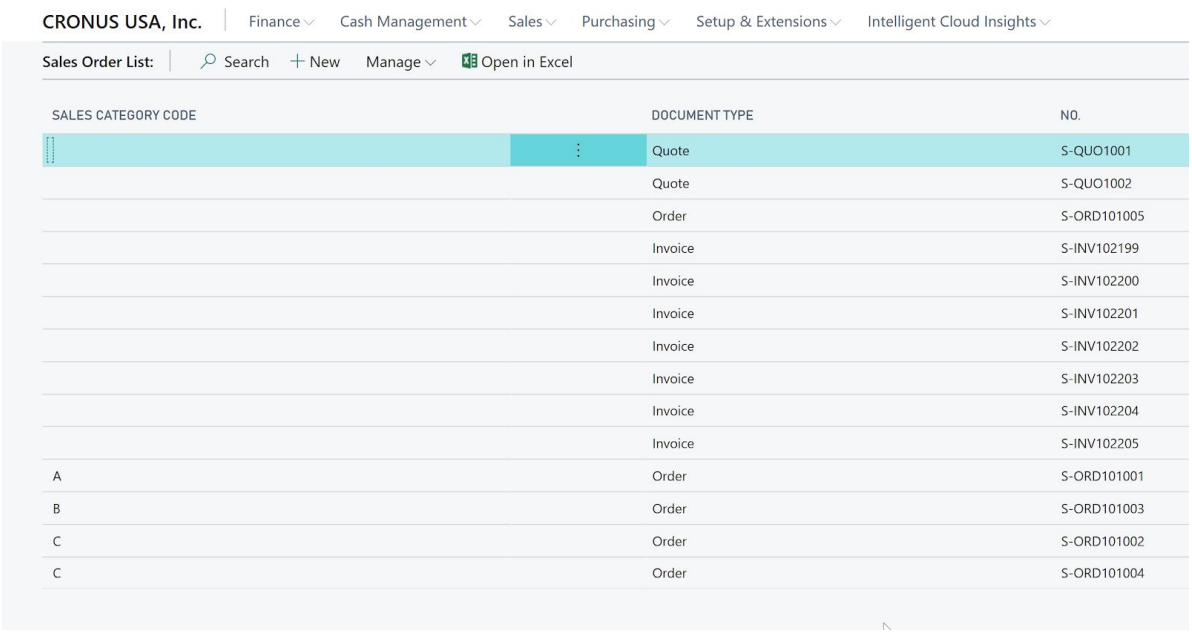


La llave primaria está implícitamente incluida al final de cualquier llave que se cree en la tabla extendida. Vamos a probarlo:

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
page 50102 "ABS Sales Order List"
{
    PageType = List;
    ApplicationArea = All;
    UsageCategory = Lists;
    SourceTable = "sales header";
    Caption = 'Sales Order List';
    SourceTableView = sorting("ABS Sales Category Code");

    layout
    {
        0 references
        area(Content)
        {
            0 references
            repeater(Control1)
            {
                0 references
                field("ABS Sales Category Code"; "ABS Sales Category Code")
                {
                    ApplicationArea = All;
                }
                0 references
                field("Document Type"; "Document Type")
                {
                    ApplicationArea = All;
                }
                0 references
                field("No."; "No.")
                {
                    ApplicationArea = All;
                }
            }
        }
    }
}
```

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



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.

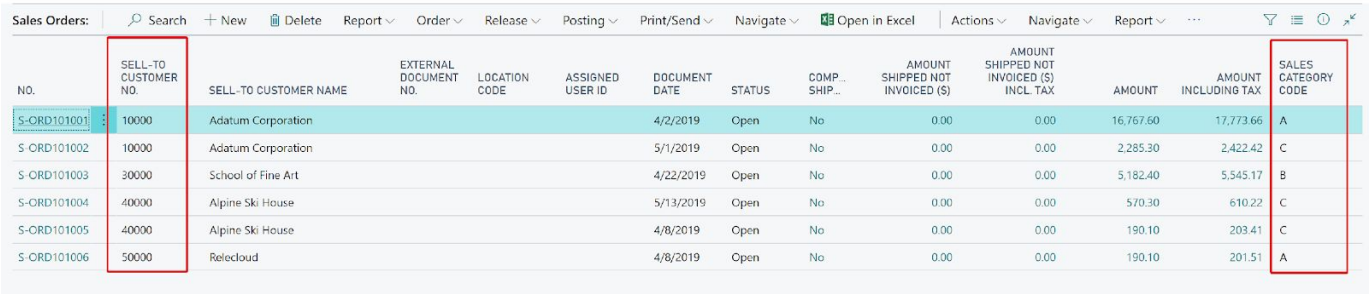
Tratemos entonces de buscar una solución alternativa, que si bien no es la mejor, puede ayudarnos a sortear algunos de los problemas que podamos tener al respecto.

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       {
            6 references
5           field(50100; "ABS Sales Category Code"; Code[10])
6           {
7               Caption = 'Sales Category Code';
8               TableRelation = "ABS Sales Category";
9               NotBlank = true;
10          }
11      }
12
13      keys
14      {
            - reference
15          key(Key1; "ABS Sales Category Code")
16          {
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.**



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
query 50100 "ABS Sales Orders by Category"
{
    QueryType = Normal;
    Caption = 'Sales Orders by Category';
    OrderBy = ascending(ABS_Sales_Category_Code, Sell_to_Customer_No_);

    elements
    {
        0 references
        dataitem(Sales_Header; "Sales Header")
        {
            DataItemTableFilter = "Document Type" = const(Order);
            2 references
            column(ABS_Sales_Category_Code; "ABS Sales Category Code")
            {
            }
            2 references
            column(Sell_to_Customer_No_; "Sell-to Customer No.")
            {
            }
            1 reference
            column(Amount; Amount)
            {
                Method = Sum;
            }
        }
    }
}
```

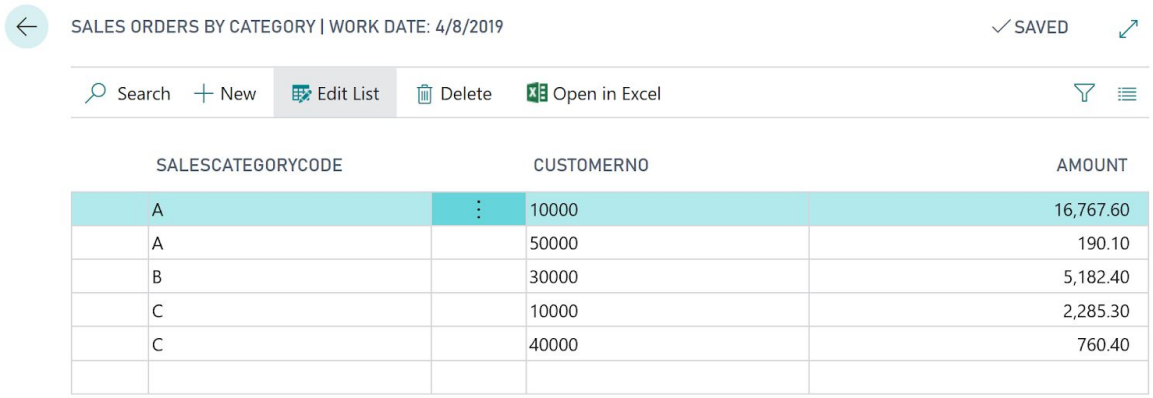
Podemos crear todas las consultas necesarias para emular las llaves que no tenemos, mientras estas llaves las necesitemos solo para efectos de ordenamiento de los datos.

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

```
1 reference
page 50101 "ABS Sales Orders by Category"
{
    PageType = List;
    ApplicationArea = All;
    UsageCategory = Lists;
    SourceTable = Integer;
    Caption = 'Sales Orders by Category';

    layout
    {
        0 references
        area(Content)
        {
            0 references
            repeater(Control100)
            {
                0 references
                field(SalesCategoryCode; SalesOrderByCategoryQuery.ABS_Sales_Category_Code)
                {
                    ApplicationArea = All;
                }
                0 references
                field(CustomerNo; SalesOrderByCategoryQuery.Sell_to_Customer_No_)
                {
                    ApplicationArea = All;
                }
                0 references
                field(Amount; SalesOrderByCategoryQuery.Amount)
                {
                    ApplicationArea = All;
                }
            }
        }
    }
```

```
    trigger OnOpenPage()
    begin
        SalesOrderByCategoryQuery.Open();
        while SalesOrderByCategoryQuery.Read() do
            NumberOfRows := NumberOfRows + 1;

        SalesOrderByCategoryQuery.Close();
        SalesOrderByCategoryQuery.Open();
        SetRange(Number, 1, NumberOfRows);
    end;

    trigger OnClosePage()
    begin
        SalesOrderByCategoryQuery.Close();
    end;

    trigger OnAfterGetRecord()
    begin
        If Not SalesOrderByCategoryQuery.Read() then
            exit;
    end;

    var
        9 references
        SalesOrderByCategoryQuery: Query "ABS Sales Orders by Category";
        3 references
        NumberOfRows: Integer;
}
```

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.



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