LOGALI

Teoría

Asociaciones y Expresiones

ABAP Cloud – Modelado con CDS

Contenido

3. Asociaciones y Expresiones

Las asociaciones se utilizan para definir relaciones entre entidades CDS, lo que permite a los desarrolladores acceder a datos relacionados de manera estructurada y eficiente. Las expresiones, por otro lado, son útiles para realizar transformaciones y cálculos en los datos proyectados, mejorando la flexibilidad y funcionalidad de las vistas CDS.

3.1. Joins

Se utilizan para combinar datos de dos o más tablas, permitiendo la creación de vistas complejas que integran información de diversas fuentes. Esto es especialmente útil en escenarios donde los datos se almacenan en diferentes tablas y es necesario obtener una vista consolidada para análisis o reportes. Es posible aplicar diferentes tipos de joins, como el **INNER JOIN**, el **LEFT OUTER JOIN** y el **RIGHT OUTER JOIN**. Para relacionar tablas o vistas cds entre fuente principal y la secundaria donde la relación se establece con la cláusula ON sintaxis similar del SQL estándar entre componentes.

Para que dichos componentes puedan relacionarse y utilizarse en un **JOIN**, deben compartir un campo o conjunto de campos que actúen como clave común, tener tipos de datos compatibles y valores coincidentes que permitan la relación entre las tablas. Las columnas que se relacionan en un **JOIN** no necesitan tener el mismo nombre, pero se pueden usar alias para las columnas si los nombres difieren. Esto asegura que las combinaciones de filas con coincidencias se incluyan en el conjunto de resultados, proporcionando datos integrados y coherentes.

También es posible agregar condiciones adicionales utilizando **AND** u **OR** para especificar relaciones más complejas entre las columnas.

Sintaxis:

```
define view entity entity_name
 as select from table_name as alias_name
   inner join table_name2 as alias_name on table_name.component
= table_name2 .component
and/or
…
```

```
left outer join
and/or
…
right outer join
and/or
…
{
        table_name.component
        …
        table_name2 .component
        …
}
```

3.2. Categoría de datos Texto

Se refiere a la clasificación y organización de los diferentes tipos de datos utilizados en un modelo de SAP. Estas categorías ayudan a definir cómo se manejan, almacenan y presentan los datos. A continuación, se presentan algunas categorías de datos y cuándo deben ser utilizadas:

- **@ObjectModel.resultset.sizeCategory:** Define el tamaño esperado del conjunto de resultados para optimizar rendimiento.
- **@Semantics.language:** Indica que el campo es una clave del idioma, esencial para la localización.
- **@Semantics.text:** Marca un campo como texto, permitiendo localización y búsqueda.
- **@ObjectModel.text.element: ['component']:** Especifica el componente de texto dentro del objeto, facilitando su manejo.

Ejemplo

```
@ObjectModel.resultSet.sizeCategory: #XS
define view entity entity_name
 as select from /dmo/oall_stat_t
{
    @ObjectModel.text.element: [ 'Text' ]
 key overall_status as OverallStatus,
    @Semantics.language: true
 key language      as Language,
    @Semantics.text: true
    text          as Text
}
```

3.3. Asociaciones

La asociación es esencial en el modelado con CDS (Core Data Services). Permite la integración y relación de datos de diversas fuentes, optimizando su intersección y proyección. Comprender las asociaciones es crucial para desarrollar modelos de datos eficientes, facilitando la navegación y las relaciones entre conjuntos de datos sin necesidad de joins tradicionales, lo cual mejora el rendimiento y la claridad del código. No hay restricciones en la cantidad de asociaciones que se pueden crear en una vista CDS, permitiendo añadir tantas como sean necesarias.

También la cardinalidad desempeña un papel vital en las asociaciones, definiendo la naturaleza de las relaciones entre las fuentes de datos. Utilizar alias es una práctica recomendada para mantener el código claro y comprensible. Prefijos con guion bajo, como "**_alias_name**", ayudan a entender mejor las rutas de navegación y seguir las buenas prácticas en el manejo de datos.

para seleccionar y manipular los datos proyectados de las fuentes principales se realizan a través de las proyecciones, mediante el comando "**$projection**, la cual es una herramienta poderosa, que optimiza las consultas y mejorando el rendimiento de las intersecciones de datos. Estas prácticas y herramientas combinadas aseguran que los modelos de datos no solo sean eficientes, sino también claros y fáciles de mantener.

Sintaxis:

```
define view entity entity_name
 as select from table_name as AliasName
 association [min..max] to table_name2 as _AliasName on
_AliasName.component = _AliasName .component
…
 association [min..max] to table_name2 as _AliasName on
_AliasName.component = $projection.alias_component_name
…
 association …
{
        key component_name
        …
        key …
        component_name,
        …
        _AliasName.component_name                    as alias_name
        …
}
```

3.4. Asociaciones con parámetros

Las asociaciones con parámetros en CDS (Core Data Services) permiten integrar y relacionar datos de diversas fuentes de manera dinámica y flexible. Al utilizar parámetros en las asociaciones, se asegura que cada consulta o proyección considere datos específicos definidos por estos parámetros, optimizando la precisión y relevancia de los resultados obtenidos.

Al consultar una vista CDS con parámetros, es necesario proporcionar el valor correspondiente para esos parámetros en alguno de los componentes del CDS. Además, una nueva vista CDS también puede incluir sus propios parámetros, permitiendo una mayor flexibilidad y especificidad en las consultas y proyecciones de datos.

Sintaxis:

```
define view entity entity_name
with parameters
        pParameter : type
as select from table_name
```

```
association [min..max] to entity_name2 as _AliasName on
_AliasName.component = $Projection.component
{
        components
        …
        _AliasName(pParameter : ‘value’). component as Alias_name
}
where component =  $parameters.pParameter;
```

Ejemplo de una asociación con parámetros en ambas CDS

```
define view entity entity_name
with parameters
        pParameter2 : type
as select from table_name
association [min..max] to entity_name2 as _AliasName on
_AliasName.component = $Projection.component
{
        components
        …
        _AliasName(pParameter1 : $parameters.pParameter2 ).
        component as Alias_name
        …
}
where component=  $parameters.pParameter2;
```

3.5. Asociación – Publicación

El proceso de publicación implica seleccionar y publicar los nombres de las asociaciones en lugar de proyectar elementos específicos. Esto permite que las asociaciones se ejecuten sólo cuando se navega a los datos correspondientes, optimizando el uso de recursos del sistema. Por ejemplo, al proyectar elementos desde una asociación, se ejecuta la consulta para obtener datos de la fuente principal y de la fuente asociada. Si se elimina la proyección y se publica la asociación, esta no se ejecuta automáticamente, sino sólo cuando es necesario.

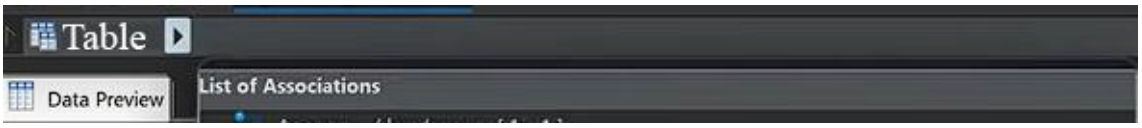
Al realizar la publicación, se deja en manos del cliente o usuario final la decisión de navegar o no a los datos de las asociaciones (Esto se puede realizar con la herramienta de test en el Eclipse presionando

la tecla F8. En la sección de lista de asociaciones en la fecha a la derecha del nombre de la entidad). Esto significa que solo se ejecutan las asociaciones que son necesarias, mejorando así el rendimiento y reduciendo el tiempo de procesamiento de consultas en comparación con los joins tradicionales.

Sintaxis:

```
define view entity entity_name
 as select from table_name as AliasName
 association [min..max] to table_name2 as _AliasName on
_AliasName.component = _AliasName .component
…
 association …
{
        component
        …
        _AliasName
        …
}
```

Visualización de la lista de asociaciones en la herramienta de test.



3.6. Asociación filtrada - Path Expression

Una asociación filtrada utilizando Path Expression permite acceder a datos relacionados a través de una cadena de asociaciones, aplicando filtros en el camino. Este enfoque se utiliza para realizar consultas complejas que implican múltiples niveles de asociaciones, proporcionando un acceso flexible a los datos relacionados.

Aquí es crucial entender que, sin filtros, la proyección de una asociación puede devolver múltiples registros para un mismo identificador. Es necesario aplicar un filtro en el path expression para reducir la cardinalidad de los resultados cuando se necesite.

Este filtro se añade dentro de los corchetes y antes del punto que proyecta el elemento deseado. La aplicación de estos filtros permite que, al realizar la consulta, los datos devueltos sean específicos, evitando la duplicidad de registros.

Sintaxis

```
_AliasName[ component = 'value '].component
```

Ejemplo de una asociación con path expressions para filtrar por el lenguaje

```
define view entity entity_name
as select from table_name
association [min..max] to entity_name2 as _AliasName on
_AliasName.component = $Projection.component
{
        components
        …
        _AliasName[ Language = $session.system_language ].
        component as Alias_name
        …
}
```

Ejemplo de una asociación con parámetros en ambas CDS

```
define view entity entity_name
with parameters
        pParameter2 : type
as select from table_name
association [min..max] to entity_name2 as _AliasName on
_AliasName.component = $Projection.component
{
        components
        …
        _AliasName(pParameter1 : $parameters.pParameter2 )[
        component = 'value' ]. component as Alias_name
        …
}
where component= $parameters.pParameter2;
```

3.7. Asociación filtrada - Cardinalidad

La cardinalidad define la relación entre conjuntos de datos y se puede ajustar mediante filtros para optimizar y especificar la proyección de los elementos asociados, mejorando así la precisión y relevancia de los resultados obtenidos. Las asociaciones filtradas por cardinalidad permiten definir el número mínimo y máximo de elementos relacionados que se pueden recuperar en una asociación.

La cardinalidad juega un papel vital en las asociaciones, definiendo la naturaleza de las relaciones entre las diferentes fuentes de datos. Esto abarca desde relaciones uno a uno, uno a muchos, hasta muchos a muchos, determinando cómo interactúan y se comportan los datos en las consultas y proyecciones. La correcta configuración de estas relaciones asegura la integridad y el rendimiento del modelo de datos, permitiendo manejar múltiples fuentes y relaciones de manera eficiente.

La cardinalidad especifica el número mínimo y máximo de registros que pueden estar relacionados entre dos entidades. En CDS, se define utilizando la notación [min..max] o simplemente [max] si el mínimo es cero.

Tipos de Cardinalidad

- **[1..1]:** Cada registro en la entidad origen está relacionado con exactamente un registro en la entidad destino.
- **[0..1]:** Cada registro en la entidad origen puede estar relacionado con cero o un registro en la entidad destino.
- **[1..*]:** Cada registro en la entidad origen está relacionado con uno o más registros en la entidad destino.

Sintaxis:

```
... [min..max]...
... [min : component = 'value']...
```

Ejemplo:

```
define view entity entity_name
 as select from table_name as AliasName
 association [min..max] to table_name2 as _AliasName on
_AliasName.component = _AliasName .component
…
 association [min..max] to table_name2 as _AliasName on
_AliasName.component = $projection.alias_component_name
…
 association …
{
        key component_name
        …
        key …
        component_name,
        …
        _AliasName.component_name                    as alias_name
        …
}
```

Ejemplo de una asociación con cardinalidad para filtrar por el lenguaje donde se espera un solo resultado.

```
define view entity entity_name
as select from table_name
association [min..max] to entity_name2 as _AliasName on
_AliasName.component = $Projection.component
{
        components
        …
        _AliasName[ 1 : Language = $session.system_language ].
        component as Alias_name
        …
}
```

3.8. Asociaciones - Join explícito

Las asociaciones con join explícito en CDS (Core Data Services) permiten modificar la relación predeterminada de un **left outer join** a un **inner join** en las proyecciones de los componentes de la asociación para modificar el tipo de relación SQL. Mientras que el **left**

outer join incluye todas las filas de la tabla izquierda, completando con **NULLs** donde no hay coincidencias en la tabla derecha, el inner join solo incluye las filas donde hay coincidencias en ambas tablas. Esta modificación se realiza agregando la palabra reservada "**inner**" entre corchetes en las proyecciones de los componentes de la asociación, lo que optimiza la precisión y eficiencia de las consultas al asegurar que solo se devuelvan los registros coincidentes. Al especificar de manera explícita el tipo de join en CDS, se puede controlar mejor el comportamiento de las consultas y adaptarlas a las necesidades específicas del análisis de datos.

Sintaxis:

```
_AliasName[inner].component
```

Ejemplo:

```
define view entity entity_name
 as select from table_name as AliasName
 association [min..max] to table_name2 as _AliasName on
_AliasName.component = $projection.alias_component_name
{
        component_name,
        …
        _AliasName[inner].component_name                    as
        alias_name
        …
        concat_with_space(     _AliasName[inner].component_name,
        _AliasName[inner].component_name) as alias_name
        …
}
```

3.9. Navegación con Path Expression

La navegación con Path Expression es una técnica fundamental que permite acceder a datos de entidades relacionadas de manera jerárquica. Utilizando path expressions, es posible navegar a través de múltiples niveles de relaciones definidas entre entidades. Esto optimiza el acceso a datos relacionados y facilita la exploración de datos en diferentes capas. Las path expressions permiten una

navegación flexible y eficiente, permitiendo que las consultas lleguen a los hijos, nietos y más allá, dependiendo de la estructura del modelo de datos. Publicar asociaciones en lugar de proyectarlas directamente habilita esta navegación, ofreciendo una vista clara y estructurada de las relaciones entre entidades.

No hay restricciones en la cantidad de niveles que se pueden utilizar para la navegación. Es posible navegar a través de múltiples capas de relaciones, accediendo a datos de hijos, nietos y más allá, dependiendo de la complejidad del modelo de datos. La configuración de path expressions en CDS permite un acceso eficiente y jerárquico a los datos, optimizando el rendimiento y proporcionando una vista clara y estructurada de las relaciones entre entidades.

Ejemplo

```
define view entity entity_name
 as select from table_name as AliasName
 association [min..max] to table_name2 as _AliasName on
_AliasName.component = _AliasName .component
…
 association …
{
        component
        …
        _AliasName
        …
}
```

Visualización de la lista de asociaciones en la herramienta de test.

3.10. Query con Path Expression

Una Query con Path Expression en CDS (Core Data Services) permite seleccionar y navegar a través de datos relacionados mediante rutas

definidas por asociaciones. Utilizando path expressions, las consultas pueden acceder a múltiples niveles de relaciones entre entidades, obteniendo datos precisos y específicos según sea necesario. Esta técnica optimiza el rendimiento de las consultas y permite una exploración flexible y eficiente de datos jerárquicos, facilitando el análisis de datos complejos y mejorando la capacidad de decisión sobre qué datos invocar en cada momento.

Una de las ventajas de utilizar path expressions es la posibilidad de decidir qué datos seleccionar en la query, basándose en las necesidades específicas del cliente. Si no se requiere acceder a ciertos elementos de una asociación, simplemente no se incluyen en la query, lo que optimiza el rendimiento evitando consultas innecesarias a la base de datos.

Las path expressions permiten una navegación flexible entre múltiples niveles de relaciones. Es posible acceder a datos de entidades hijas, nietas y más allá, dependiendo de la estructura del modelo de datos. Esta capacidad de navegar entre capas y seleccionar datos específicos mejora la precisión de las consultas y facilita el análisis de datos complejos.

Sintaxis

```
SELECT FROM entity_name AS AliasName
  FIELDS AliasName~AliasComponent,
            …
    \_Association\_AssociationN-component as AliasComponent
    …
  WHERE conditions
  INTO/INTO TABLE structure/internal_table
```

Ejemplo

```
define view entity entity_name
 as select from table name as AliasName
 association [1..1] to entity_name2 as _Alias_name on
_Alias_name.agency_id = $projection._Alias_name
…
association
```

```
...
{
        component
        ...
        _Alias_name
        _Alias_nameN
}
```

Ejemplo de una Query con Path Expression

```
SELECT FROM entity_name AS AliasName
FIELDS AliasName~AliasComponent,
       AliasName~AliasComponent,
  \_Association-component as AliasComponent
WHERE AliasName~AliasComponent EQ 'value'
INTO TABLE @DATA(lt_results)
UP TO 1 ROWS.
if sy-subrc eq 0.
  out->write( lt_results ).
endif.
```