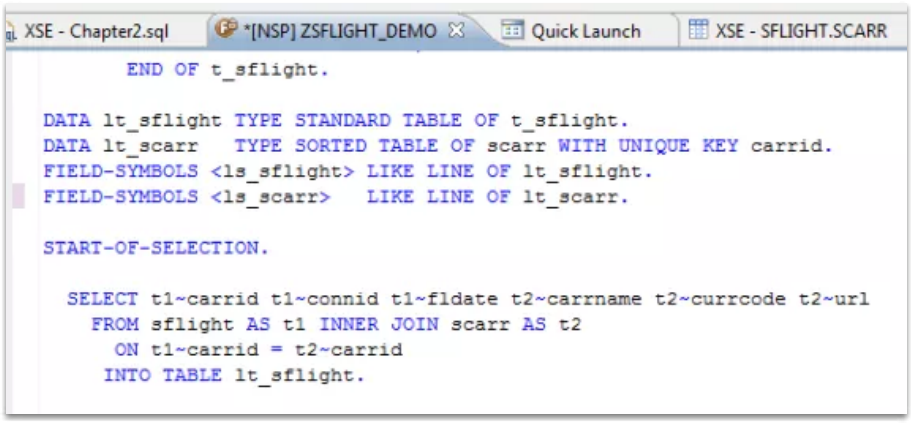
Richard Rey






# Diferencias entre INNER JOIN vs. FOR ALL ENTRIES

¿Qué es más eficiente? Un ***INNER JOIN*** o un ***FOR ALL ENTRIES***.

Seguro te habrás hecho esta pregunta en algún momento, cuando te encontrabas ante un reporte con un largo tiempo de duración, y al buscar por Internet, te habrás dado cuenta de que existe diversidad de opiniones con respecto a este punto.

Considero que la mejor respuesta a esta pregunta es **DEPENDE**..

Si, depende de varios factores, las tablas que estas consultando, los campos que estas extrayendo, si son claves primarias o no, los índices de las tablas, etc.

Pero para poder tomar la decisión correcta, primero debemos tomar en cuenta algunas consideraciones que aplican para cualquiera de las dos opciones.

Lo primero que debemos tener en cuenta, es evitar el uso de instrucciones que consumen mucha memoria, como las explicadas en ***ESTE ARTICULO***.

Adicionalmente, evitar usar a toda costa el **CORRESPONDING** en todas sus formas, es decir, INTO CORRESPONDING en un SELECT, o un MOVE CORRESPONDING en un LOOP, etc.

Esto va de la mano con el uso del **SELECT ***, algo que es imperativo evitar, por muy cómodo que sea.

Es preferible tomarse un tiempo para analizar los campos exactos que necesitaremos, y crear una tabla o estructura destino que contenga exactamente estos campos.

Y por último evitar el **SELECT-ENDSELECT**, en todas sus formas como se menciona en el artículo anterior, siempre es preferible realizar el menor número de accesos a Base de Datos posibles para conseguir optimización de tiempos.

Ahora procedamos a entender un par de cosas más.

**Características del FOR ALL ENTRIES**

El ***FOR ALL ENTRIES*** fue creado para reemplazar el SELECT dentro de un LOOP, y de esta forma reducir los N accesos a Base de Datos a un único acceso con los datos exactos contenidos en la primera tabla. Es decir, el ***FOR ALL ENTRIES*** nace para convertir esto:

```
LOOP AT IT_AFKO.

  SELECT MATNR FROM RESB INTO TABLE IT_RESB WHERE AUFNR = IT_AFKO-AUFNR.

ENDLOOP.
```

En esto:

```
SELECT MATNR FROM RESB INTO TABLE IT_RESB_tmp

FOR ALL ENTRIES IN IT_AFKO WHERE AUFNR = IT_AFKO-AUFNR.
```

Convertir N accesos a Base de Datos (tantos como registros tenga la tabla IT_AFKO), en un único acceso utilizando como clave en la cláusula WHERE los N registros de dicha tabla.

Con la opción ***FOR ALL ENTRIES*** se consigue que los registros existentes en la primera tabla (IT_AFKO en el ejemplo), se consigan ser accedidos en el SELECT a través de un único acceso a Base de Datos (tabla RESB en el ejemplo).

Con la Instrucción ***FOR ALL ENTRIES*** hay que tener mucho cuidado en que la tabla base (IT_AFKO en el ejemplo), con la se va a comparar no se encuentre vacía, porque en ese caso, estaríamos extrayendo **TODA LA DATA** existente en la tabla a consultar, perjudicando el performance del programa, es decir, siguiendo con el ejemplo, si la tabla interna IT_AFKO se encuentra vacía al momento de ejecutar el ***FOR ALL ENTRIES***, el sistema estaría haciendo caso omiso a la cláusula WHERE e intentaría recuperar TODOS LOS REGISTROS de la tabla RESB, la cual es una tabla con millones de registros y en este punto estaríamos dañando por completo el performance de nuestro programa.

Otro punto a tener en cuenta es si la tabla interna tiene muchos registros, ya que en este caso, el SELECT sería mucho más lento, perjudicando el performance por la utilización de mucha memoria.

**Características del INNER JOIN.**

El uso del ***INNER JOIN*** está bien aceptado cuando se requieren grandes extracciones de datos, su uso eficiente en estos casos ofrece muy buenos resultados siempre que la cláusula JOIN se ejecute sobre campos claves entre las tablas a enlazar, y adicionalmente si en la cláusula WHERE se utiliza de forma óptima algún índice de la tabla principal, ya que lo más costoso de un SELECT es cuando la búsqueda no utiliza índices, y esto se ve reflejado aún más si se utilizan ***INNER JOINS***.

El ***INNER JOIN*** sin el uso correcto de índices o campos claves hace tanto daño como el ***FOR ALL ENTRIES*** sobre una tabla interna vacía.

Cuando el ***INNER JOIN*** no está construido correctamente, es habitual encontrar entre los resultados registros duplicados; esto nos obliga a utilizar la sentencia DELETE ADYACENT DUPLICATES para depurar los datos resultantes de la consulta.

Para grandes extracciones de datos, o consultas a tablas muy pesadas, el ***INNER JOIN*** pareciera ser la mejor opción, pero hay un detalle a tener en cuenta, las TABLAS CLUSTERS no permiten este tipo de anidaciones. Tablas como BKPF o BSEG no pueden ser consultadas a través de ***INNER JOINS***, solo pueden ser consultadas a través de SELECT directos.

Por lo tanto, es en estos casos cuando un ***FOR ALL ENTRIES*** pasa a ser de gran utilidad, siempre que se hayan tomado en cuenta todas las consideraciones mencionadas a lo largo de este artículo para la construcción de los SELECT respectivos.

En resumen, si vas a extraer grandes cantidades de datos, el ***INNER JOIN*** es una buena opción, pero si deseas manejar primero una parte de la data antes de pasarla a la siguiente extracción, el ***FOR ALL ENTRIES*** sería la mejor estrategia.

Al final del día todo dependerá de lo que necesites hacer. Pero siempre debes tener en cuenta todas las consideraciones para optimizar las respectivas consultas, si vas a utilizar ***INNER JOINS*** asegúrate de utilizar claves primarias y/o índices de las tablas a enlazar, además de asegurarte de no duplicar registros en la tabla resultante.

Si vas a utilizar ***FOR ALL ENTRIES*** es imperativo que te asegures que la tabla interna base no este vacía porque toda la estrategia de performance se perderá.

Y por último, siempre tienes en tus manos la opción de comprobar por ti mismo el rendimiento de la lógica a implementar, a través de la **transacción SE30**.

¿Te ha sido de utilidad este artículo? Házmelo saber con tus comentarios, y suscríbete a mi Comunidad para recibir mi curso gratuito “AprendeABAP en 10 lecciones”, además de contenidos exclusivos sobre el mundo ABAP.

***Richard Rey***

***“Programando tu Exito”***



## Contenido Relacionado:

Tips para mejorar las consultas ABAP

Plantilla para construir un ALV

Navegación a través de un reporte ALV

Campos del sistema SAP más utilizados en ABAP



**RELATED POSTS**

**Como entender el Log de errores…**

**0 Comments**

**Como crear una Ayuda de Búsqueda…**

**0 Comments**

**Diferencias entre Puntos de…**

**0 Comments**

**[MEGAPOST] Creación de un…**

**0 Comments**

## Comentarios