

Tablo Ve View Farkı Nedir?

Tablo (Table) ve View (Görünüm) arasındaki temel farklar şunlardır:

Verilerin Saklanma Yeri:

- Tablo: Gerçek verileri depolar. Tablolar, veritabanında fiziksel olarak saklanan yapısal veri depolama birimleridir.
- View: Gerçek verileri depolamaz, yalnızca bir veya birden fazla tablodan gelen verilere erişim sağlayan sanal bir yapıdır.

Veri Saklama ve Güncelleme:

- Tablo: Gerçek verileri saklar ve bu verilere doğrudan ekleme, güncelleme ve silme işlemleri yapılabilir.
- View: Yalnızca birleştirilmiş veya filtrelenmiş verilere erişim sağlar, bu nedenle genellikle sadece okunabilir (read-only)dir. View'lar üzerinden yapılan değişiklikler genellikle altındaki tablolara etki eder.

Veri Yapısı ve İçerik:

- Tablo: Kendine özgü bir veri yapısına ve alanlara sahiptir. Veri tipleri, uzunluklar ve diğer özellikler tabloya özgüdür.
- View: Tablolardan gelen verilere dayalı bir yapıya sahiptir. View, genellikle birleştirilmiş tabloların alanlarını içerir.

Performans ve Bellek Kullanımı:

- Tablo: Veriler fiziksel olarak depolandığından, doğrudan erişim sağlar ve genellikle hızlıdır. Ancak, daha fazla bellek kullanabilir.
- View: Verileri fiziksel olarak depolamadığı için, sorgu çalıştığında verileri birleştirir. Performans, altındaki tabloların karmaşıklığına ve birleştirilen alanlara bağlı olarak değişebilir.

Güvenlik ve Erişim Kontrolü:

- Tablo: Doğrudan erişime açık olduğu için güvenlik konularını yönetmek daha karmaşıktır.
- View: İhtiyaca bağlı olarak belirli alanları veya verileri gizleme, filtreleme ve sınırlama gibi güvenlik ayarları daha kolay uygulanabilir.

Mantıksal Görünüm:

- Tablo: Fiziksel veri depolama birimidir ve genellikle uygulamanın iş mantığına odaklanmaz.
- View: Mantıksal bir görünüm sunar, belirli bir kullanım senaryosuna uygun olarak birleştirilmiş veya filtrelenmiş verilere odaklanabilir ve databasede bunun için yer ayrılmaz

Bu temel farklar, tablo ve view kullanımının farklı senaryolarda uygun olmasını sağlar. Tablolar genellikle gerçek veri depolama ve işleme için kullanılırken, view'lar daha çok verilere erişimde esneklik sağlamak veya veri üzerindeki belirli işlemleri kolaylaştırmak amacıyla kullanılır.

ABAP programlama dilinde, SAP sistemlerinde kullanılan VIEW türleri, kullanıcıların belirli veritabanı ihtiyaçlarına yönelik esnek ve özelleştirilebilir erişim sağlamak amacıyla kullanılır. İşte bazı yaygın VIEW türleri:

Database View (Veritabanı Görünümü): Bu tür VIEW'lar, bir veya daha fazla tablonun birleştirilmesiyle oluşturulur. İki ana türü vardır:

- Join View: Birden fazla tabloyu birleştirir.
- Projection View: Belirli alanları ve verileri seçer, bu sayede daha özelleştirilmiş bir görünüm sunar.

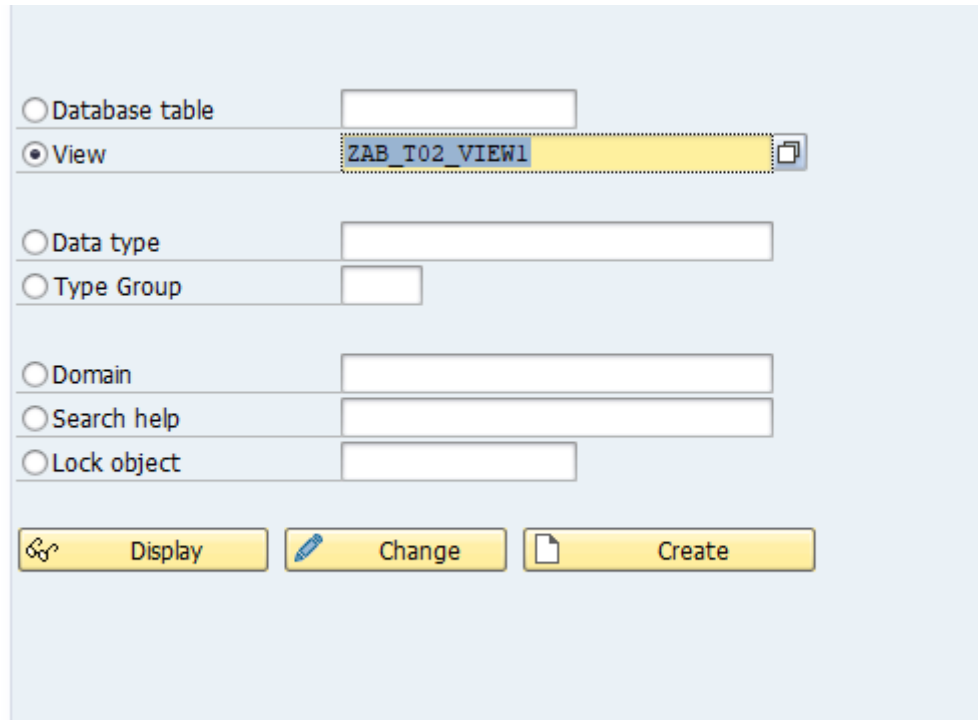
Maintenance View (Bakım Görünümü): Bu tür VIEW'lar, tabloların verilerini düzenleme, ekleme ve silme işlemleri için kullanılır. Maintenance View, genellikle bir veya birden fazla tablonun birleştirilmiş görünümünü sağlar ve bu görünüm üzerinden veri bakım işlemlerini gerçekleştirmeyi mümkün kılar.

Help View (Yardım Görünümü): Bu tür VIEW'lar, belirli bir veri nesnesini açıklamak veya daha anlaşılır hale getirmek amacıyla kullanılır. Yardım görünümleri, kullanıcıların bir alanı veya bir veri değerini daha iyi anlamalarına yardımcı olmak için oluşturulabilir.

Projection View (Projeksiyon Görünümü): Bu tür VIEW'lar, bir veya daha fazla tablodan belirli alanları seçerek, bu alanları birleştirilmiş bir görünümde kullanmak amacıyla oluşturulur. Genellikle kullanıcıların belirli bir işleme odaklanmasını sağlar.

Bu VIEW türleri, SAP uygulamalarının veritabanı erişimini ve yönetimini esnek ve etkili bir şekilde gerçekleştirmek için kullanılır. Her bir VIEW türü, belirli bir senaryoya veya ihtiyaca uygun olarak seçilebilir.

Örnek View Ve Tablo Kullanımı:



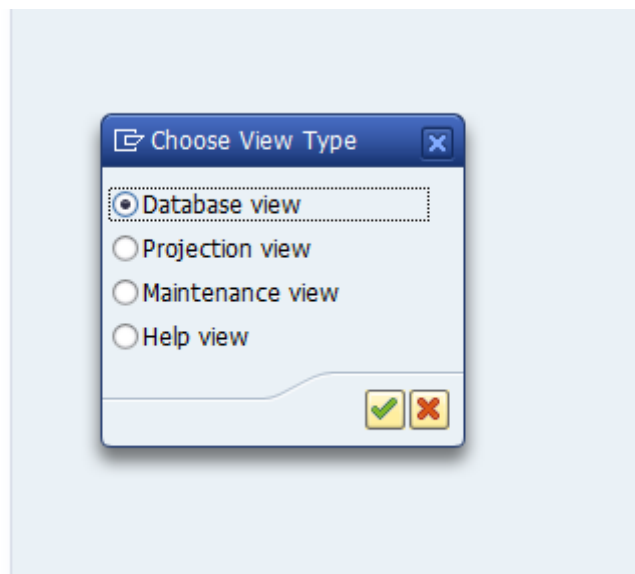
The screenshot shows the SE11 TCODE interface with the following options and fields:

- ☐ Database table
- ☒ View (Selected option, with a text field containing "ZAB_T02_VIEW1" and a copy icon)
- ☐ Data type
- ☐ Type Group
- ☐ Domain
- ☐ Search help
- ☐ Lock object

At the bottom, there are three buttons: "Display" (with a magnifying glass icon), "Change" (with a pencil icon), and "Create" (with a document icon).

Şekil 1. SE11 TCODE

Şekil 1 deki tcode üzerinden view alanını seçiyoruz.



Şekil 2. SE11 TCODE SEÇİM

Şekil 2 de hangi tür view kullanacaksak onu seçiyoruz.

The screenshot shows a software interface for creating a database view. At the top, there are fields for 'Database View' (containing 'ZAB_T01_DBVIEW') and 'Short Description' (containing 'DB VIEW'). Below these are tabs: 'Attributes', 'Table/Join Conditions', 'View Fields', 'Selection Conditions', and 'Maint.Status'. The 'Table/Join Conditions' tab is active. On the left, a 'Tables' list contains 'ZMM_T001' and 'ZAB_PERSONEL T', with the latter selected. On the right, a 'Join conditions' table is visible with columns 'Table', 'Field Name', '=', 'Table', and 'Field'. At the bottom, there is a 'Relationships' button.

Şekil 3. TABLO SEÇİMİ

Bu Şekilde view oluşturmak istediğimiz alanları tanımlarız ve ilişki oluşturacağımız alanların relationshiplerini join olarak yazarız.

```
SELECT *
```

```
FROM employees
```

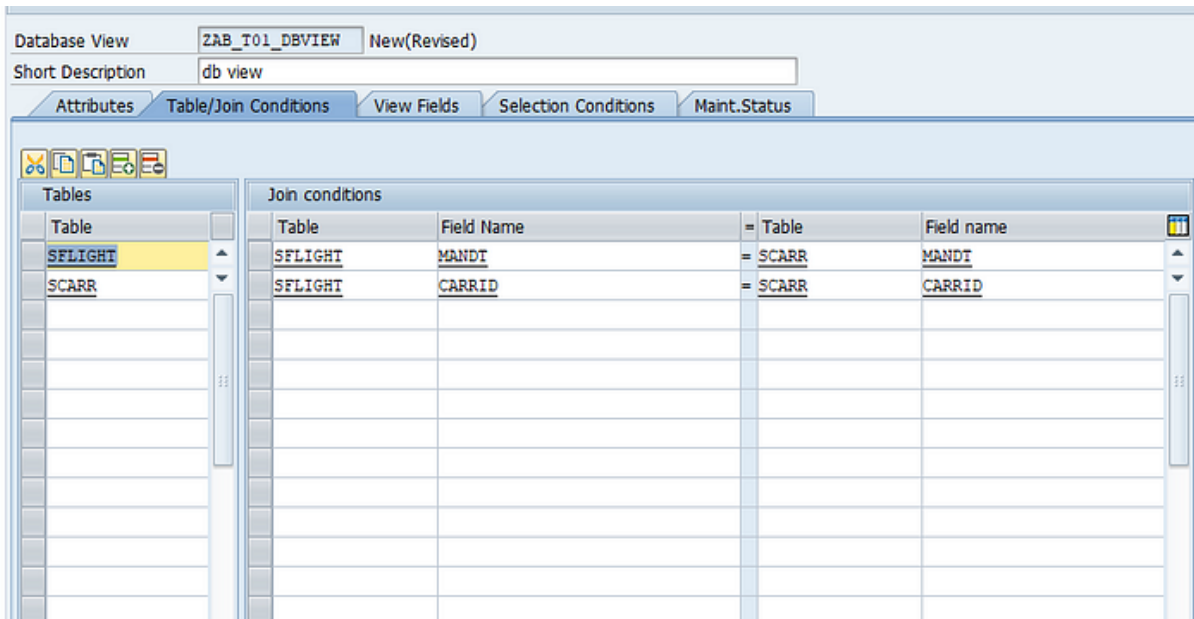
```
JOIN departments ON employees.department_id = departments.department_id;
```

tablonun işlevi aslında yukarıdaki kod bloğunun yaptığı işi yapmak .

Bu SQL sorgusu, “employees” ve “departments” adlı iki tabloyu birleştirir. “employees” tablosu çalışanların bilgilerini içerirken, “departments” tablosu departmanların bilgilerini içerir. Bu iki tablo arasındaki ilişki “department_id” alanı üzerinden kurulmuştur. Bu alan, her çalışanın hangi departmana ait olduğunu belirtmek için kullanılır.

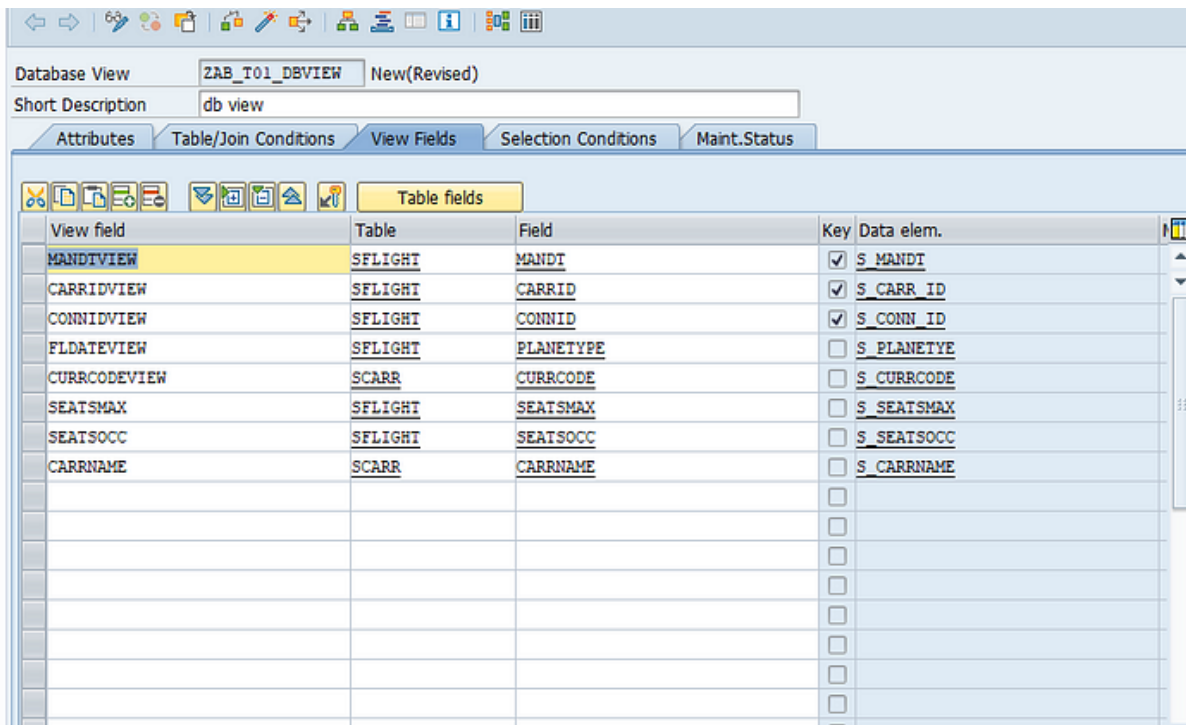
İlk olarak, FROM ifadesi ile sorgu, "employees" tablosu ile başlar. Daha sonra JOIN ifadesi kullanılarak "departments" tablosu bu sorguya eklenir. ON ifadesi, birleştirme işlemini gerçekleştirecek olan alanları belirtir ve bu örnekte "employees.department_id" ile "departments.department_id" eşleştirilmiştir.

Sonuç olarak, her çalışanın bilgisi ile ilişkilendirilmiş departman bilgisi elde edilir. Bu sorgunun sonucu, her bir çalışanın tüm bilgilerini ve onların hangi departmana ait olduklarını içeren geniş bir sonuç kümesidir.



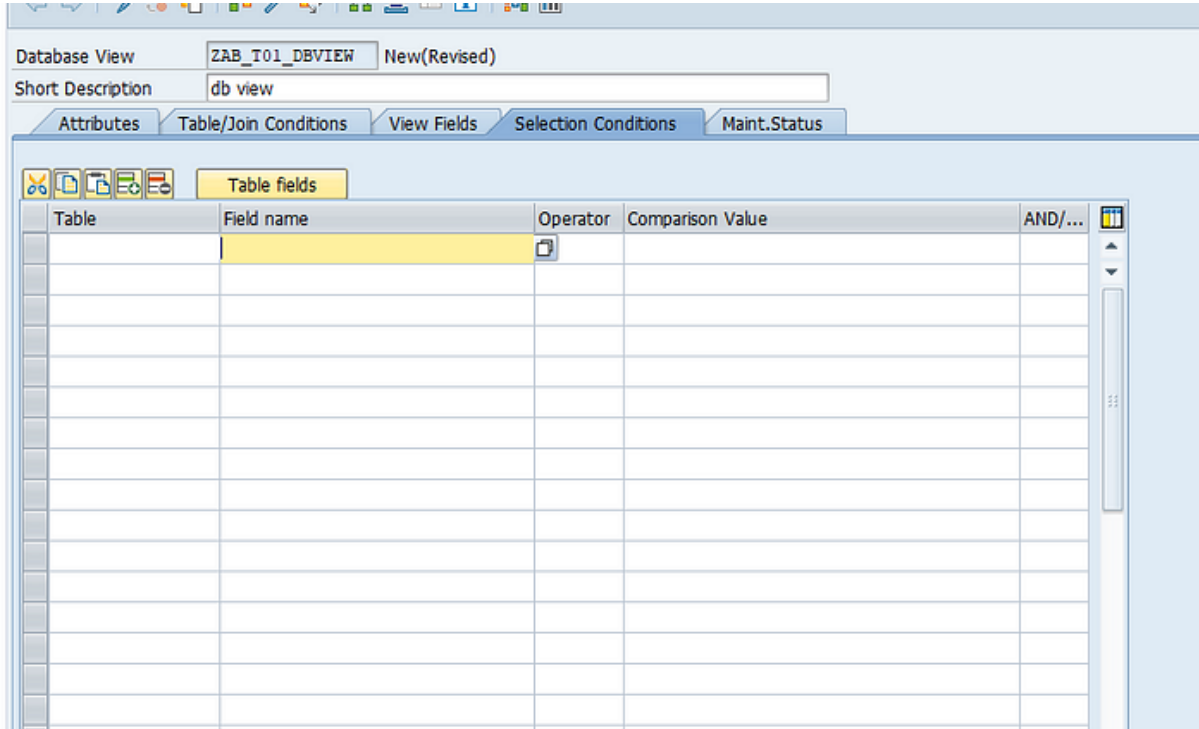
Şekil 4. Ortak İlişkiler

Şekil 4. de görüldüğü gibi ortak alanlar tanımlandı.



Şekil 5. Gösterilmesi İstenen Alanlar

Şekil 5. deki ekranda gösterilmesi istenen alanlar seçilmektedir. Seçilen bu alanlar database view'e basılacaktır



Şekil 6. Selection Condition

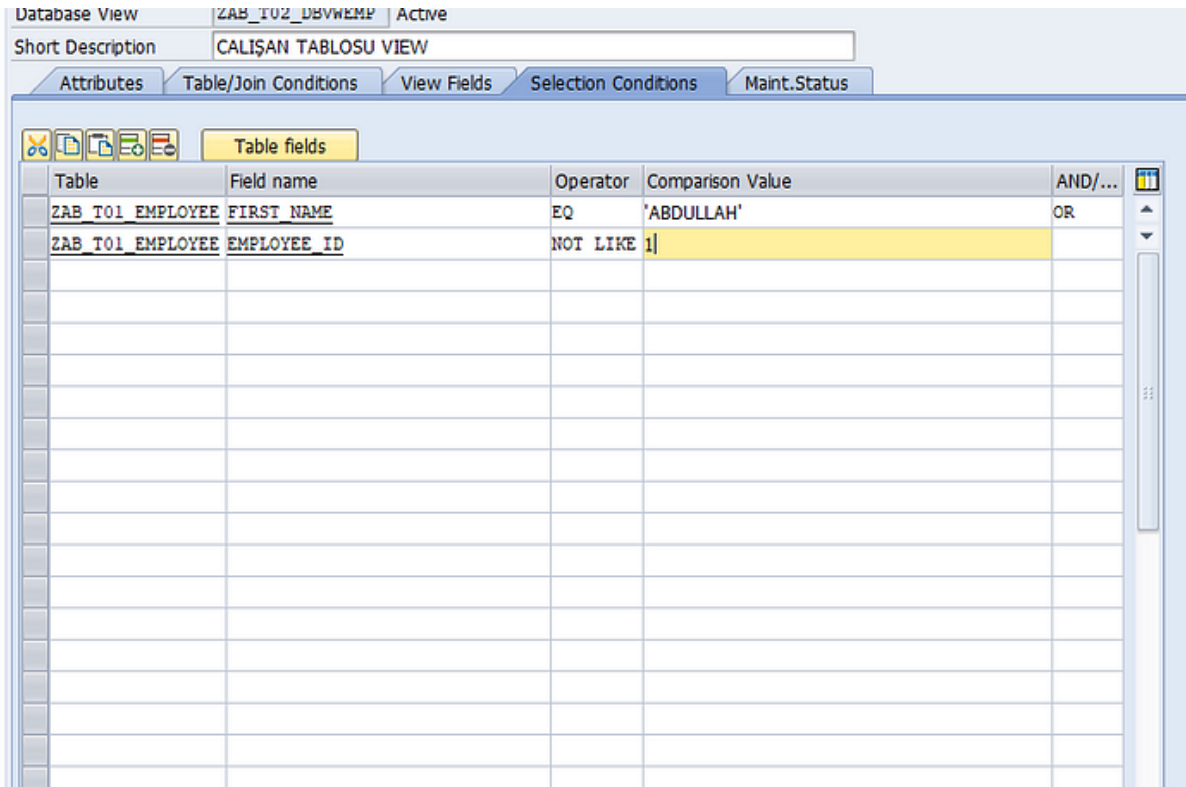
Selection Condition ekranında view'de görünmesini istediğimiz koşulları sağlayan satırları view'e basacak koşulları yazabiliriz.

```
SELECT product_id, product_name, price
```

```
FROM products
```

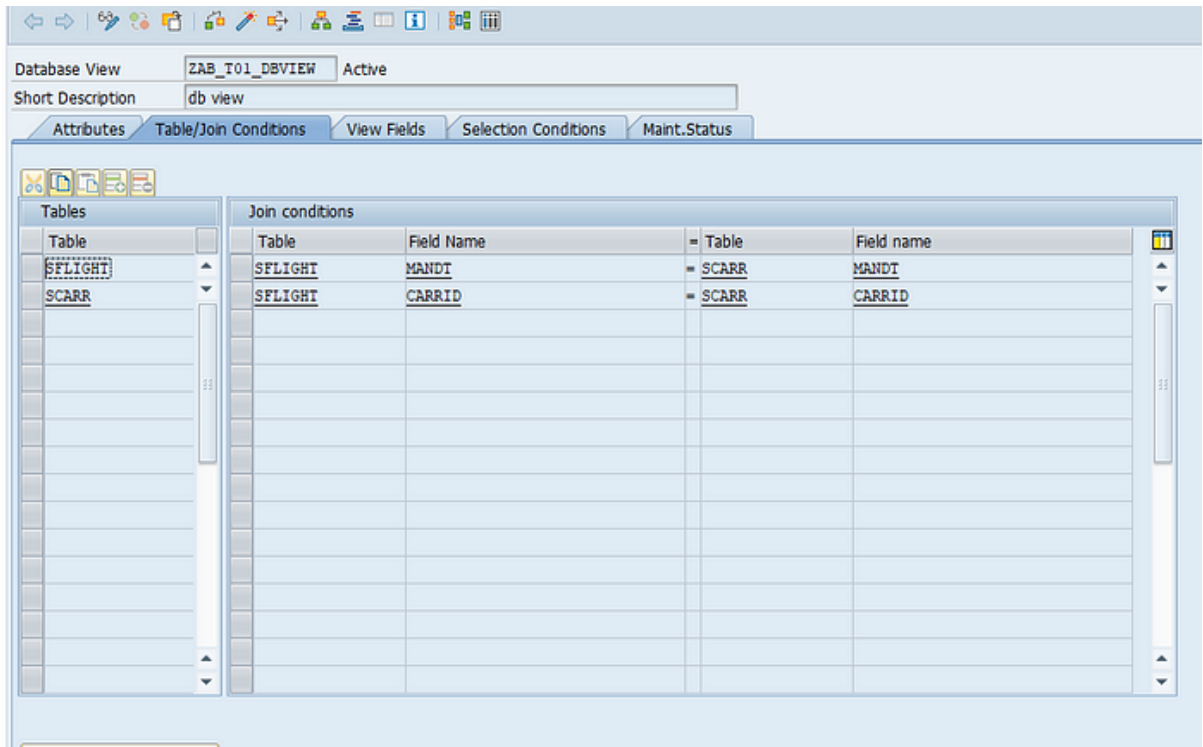
```
WHERE price BETWEEN 50 AND 100;
```

Yukarıdaki kod örneği gibi bir çalışma işlevi vardır.



Şekil 7. Selection Condition

Yukarıdaki örnekte adı abdullah olan 'OR' employee id si 1 olmayan değerleri ekrana bas gibi bir koşul koyduk.



Şekil 8. Ortak Alanlar

ABAP View'ler CDS View'ler

ABAP View'ler ve CDS (Core Data Services) View'ler, SAP sisteminde veri modelleme ve veritabanı bağlamında kullanılan iki farklı yaklaşımdır. Her ikisi de veri tabanındaki veriye erişim sağlama ve veri modelleme yeteneklerini geliştirme amacını taşısa da, aralarında önemli farklar bulunmaktadır.

ABAP View'ler:

- Dil ve Ortam: ABAP (Advanced Business Application Programming) View'ler, ABAP dilinde yazılan programlardır. Bu view'ler genellikle SAP ERP sistemlerinde kullanılır.
- Teknoloji ve Mimarlık: ABAP View'ler, Open SQL (SAP'nin özel SQL türevidir) kullanarak veritabanına erişim sağlarlar. Veritabanına bağlıdır ve veri tabanı bağlamında birçok özelliğe erişim sağlayabilirler.
- Fiziksel olarak tutulmazlar yani databasede veri kaydı view için oluşturulmaz.
- Veri güvenliği için idealdir çünkü veritabanından çekip listeleyeceğimiz bir veriyi direk manipüle edebiliriz ama view kullanılırsa database görünümünü göstereceği için manipülasyon ihtimali ortadan kalkar.

CDS View'ler:

- Dil ve Ortam: CDS (Core Data Services) View'ler, SQL tabanlı bir dil olan CDS dilinde yazılırlar. CDS, SAP'nin modern uygulama geliştirme ve analitik çözümleri için tasarlanmış bir dil ve teknolojidir.
- Teknoloji ve Mimarlık: CDS View'ler, SAP HANA veritabanı gibi modern veritabanlarına özgü avantajları kullanarak çalışırlar. Özellikle, HANA'nın in-memory veritabanı yeteneklerinden faydalanabilirler.
- Geniş Bağlamda Kullanım: CDS View'ler, sadece SAP ERP sistemleriyle sınırlı değildir. SAP Fiori uygulamaları, SAP BW (Business Warehouse), SAP S/4HANA gibi çeşitli SAP uygulamalarında ve senaryolarında kullanılabilirler.

İki yaklaşım da veri tabanı erişimi ve veri modelleme yetenekleri sağlar, ancak CDS, özellikle SAP'nin daha modern ve geleceğe yönelik uygulama geliştirme stratejilerine daha iyi uyum sağlar. CDS ayrıca HANA veritabanının sunduğu performans avantajlarından faydalanarak daha etkili bir şekilde çalışabilir.

3 Z'li Tablo Oluşturup İlişkilendirilip View Oluşturulması

Tablo 1: Çalışanlar ZAB_T01_EMPLOYEE

Alanlar:

- employee_id (ZAB_EMPLOYEE_ID_DE) //PK,
- department_id (ZAB_DEPARTMENT_ID_DE) //FK,
- position_id (ZAB_POSITION_ID_DE) //FK,
- first_name (ZAB_FIRST_NAME_DE),
- last_name (ZAB_LAST_NAME_DE)

Tablo 2: Departmanlar ZAB_T01_DPRTMENT

Alanlar:

- department_id (ZAB_DEPARTMENT_ID_DE) PK,
- department_name (ZAB_DEPARTMENT_NAME_DE),
- manager_id (ZAB_MANAGER_ID_DE)

Tablo 3: Pozisyonlar ZAB_T01_POSITION

Alanlar:

- position_id (ZAB_POSITION_ID_DE) PK,
- position_name (ZAB_POSITION_NAME_DE),
- salary (ZAB_SALARY_DE)

Senaryo: Bir şirkette çalışanların bilgilerini takip etmek için bir veritabanı tasarlanmıştır. Çalışanlar, departmanlar ve pozisyonlar arasında belirli bir hiyerarşi bulunmaktadır.

1. Her çalışan bir kimlik numarası, adı, soyadı ve bağlı olduğu departman bilgisi ile temsil edilmektedir. “employees” tablosu, çalışanların bu bilgilerini içerir. Bir çalışanın bağlı olduğu departman, “departments” tablosundaki ilgili departmanın kimlik numarasına referans verir.
2. Departmanlar, bir kimlik numarası, departman adı ve departman müdürünün kimlik numarası ile temsil edilmektedir. “departments” tablosu, departmanların bu bilgilerini içerir. Departman müdürünün kimlik numarası, “employees” tablosundaki bir çalışanın kimlik numarasına referans verir.
3. Pozisyonlar, bir kimlik numarası, pozisyon adı ve pozisyonun maaşı ile temsil edilmektedir. “positions” tablosu, pozisyonların bu bilgilerini içerir. Çalışanlar, “employees” tablosundaki bir çalışanın pozisyon kimlik numarasına referans verir.

Bu senaryo, çalışanların, departmanların ve pozisyonların birbirine bağımlı bir şekilde tutulduğu bir iş dünyasını yansıtmaktadır. Çalışanlar belirli bir departmana bağlıdır, departmanlar müdürleri tarafından yönetilir, ve her çalışan belirli bir pozisyonda görev yapar. Bu tablolar ve bağımlılıkları, iş süreçlerini ve organizasyonel yapının daha etkili bir şekilde yönetilmesini sağlamak için kullanılabilir.

Transparent Table **ZAB_T01_EMPLOYEE** Active

Short Description **Çalışan tablosu**

Attributes Delivery and Maintenance Fields Input Help/Check Currency/Quantity Fields Indexes

1 / 6

Field	Key	Ini...	Data element	Data Type	Length	Deci...	Coordinate	Short Description
<u>MANDT</u>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3	0	0	Client
<u>EMPLOYEE_ID</u>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	ZAB_EMPLOYEE_ID..	INT4	10	0	0	ÇALIŞAN ID DATA ELEMENT
<u>DEPARTMAN_ID</u>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZAB_DEPARTMENT ..	CHAR	25	0	0	DEPARTMAN ID DE
<u>POSITION_ID</u>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZAB_POSITION_ID ..	INT4	10	0	0	ZAB_POSITION_ID_DE
<u>FIRST_NAME</u>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZAB_FIRST_NAME ..	CHAR	25	0	0	ADI DATA ELEMENT
<u>LAST_NAME</u>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZAB_LAST_NAME DE	CHAR	25	0	0	SOYADI DE

Şekil 1. Foreign Key Atama Tuşu

VIEW OLUŞTURMA //ZAB_T02_DBVWEMP

Database View **ZAB_T02_DBVWEMP** New(Revised)

Short Description **ÇALIŞAN TABLOSU VIEW**

Attributes Table/Join Conditions View Fields Selection Conditions Maint.Status

Table	Field Name	=	Table	Field name
ZAB_T01_EMPLOYEE	MANDT	=	ZAB_T01_EMPLOYEE	MANDT
ZAB_T01_POSITION	DEPARTMAN_ID	=	ZAB_T01_EMPLOYEE	DEPARTMAN_ID
ZAB_T01_POSITION	MANDT	=	ZAB_T01_EMPLOYEE	MANDT
ZAB_T01_POSITION	POSITION_ID	=	ZAB_T01_EMPLOYEE	POSITION_ID

Şekil 2. TABLOLARDAKİ JOIN ALANLARI

Runtime

Maximum No. of Hits

Insert Column

DEP ID	DEPARTMAN ADI	manager id
1	IT	123
2	HR	421
3	BUSINESS	5,435
4	LAW	1,234
5	MATERIAL	51
6	MANUFACTURING	343

Şekil 5. Departman tablomuz

Search in Table

ZAB_T01_POSITION

Pozisyon tablosu

Number of Hits

3

Runtime

0

Maximum No. of Hits

500

Insert Column

POSITIONID	POS NAME	SALARY
3	3	12,312
2	2	21,413
1	1	13,123

Şekil 6. Pozisyon tablomuz

SQL SORGU SONUCU

```
ÇALIŞAN ID:          5
DEPARTMAN ID: 5
ZAB_POSITION_ID:      1
POSITION_NAME:        1
ÇALIŞAN ID: MATERIAL
DEPARTMAN ADI: AHMET
SOYADI: YILMAZ
Client: 400
```

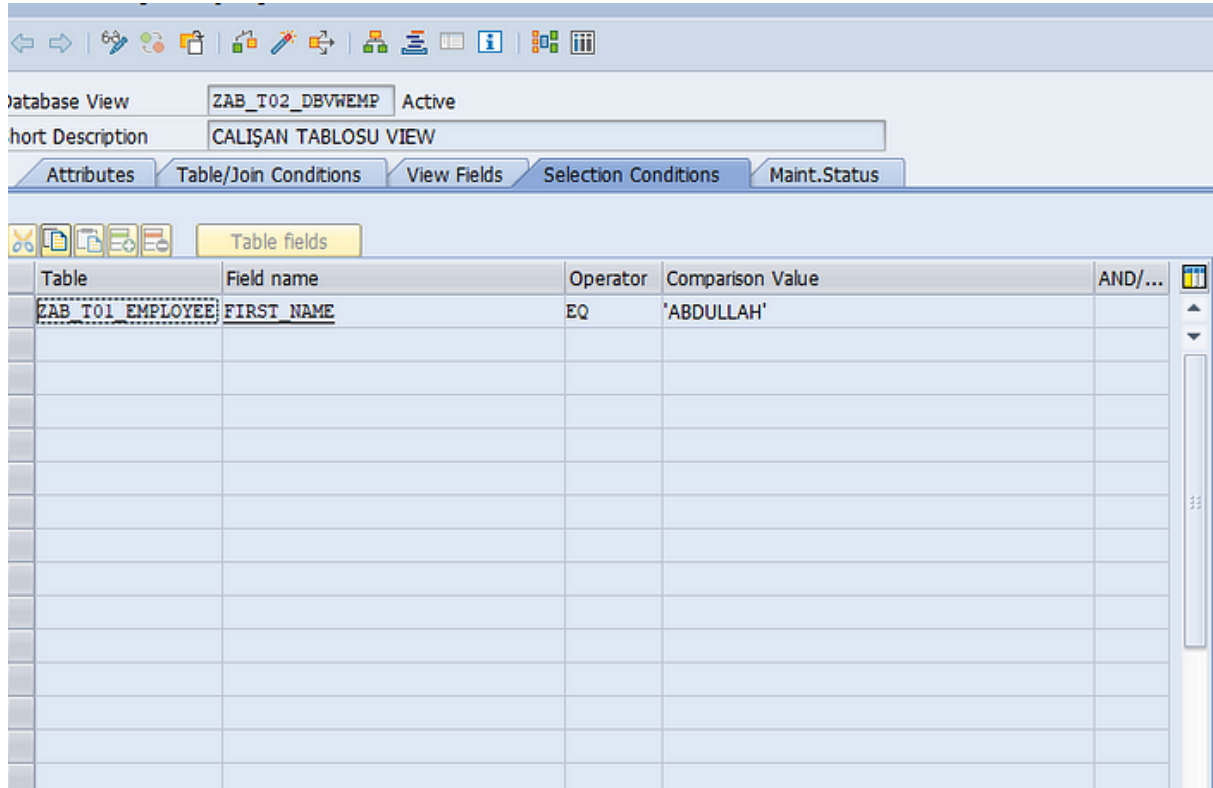
```
ÇALIŞAN ID:          4
DEPARTMAN ID: 4
ZAB_POSITION_ID:      3
POSITION_NAME:        3
ÇALIŞAN ID: LAW
DEPARTMAN ADI: ERCIYES
SOYADI: HOLDING
Client: 400
```

```
ÇALIŞAN ID:          3
DEPARTMAN ID: 1
ZAB_POSITION_ID:      2
POSITION_NAME:        2
ÇALIŞAN ID: IT
DEPARTMAN ADI: ABDULLAH
SOYADI: BOZLAĞAN
Client: 400
```

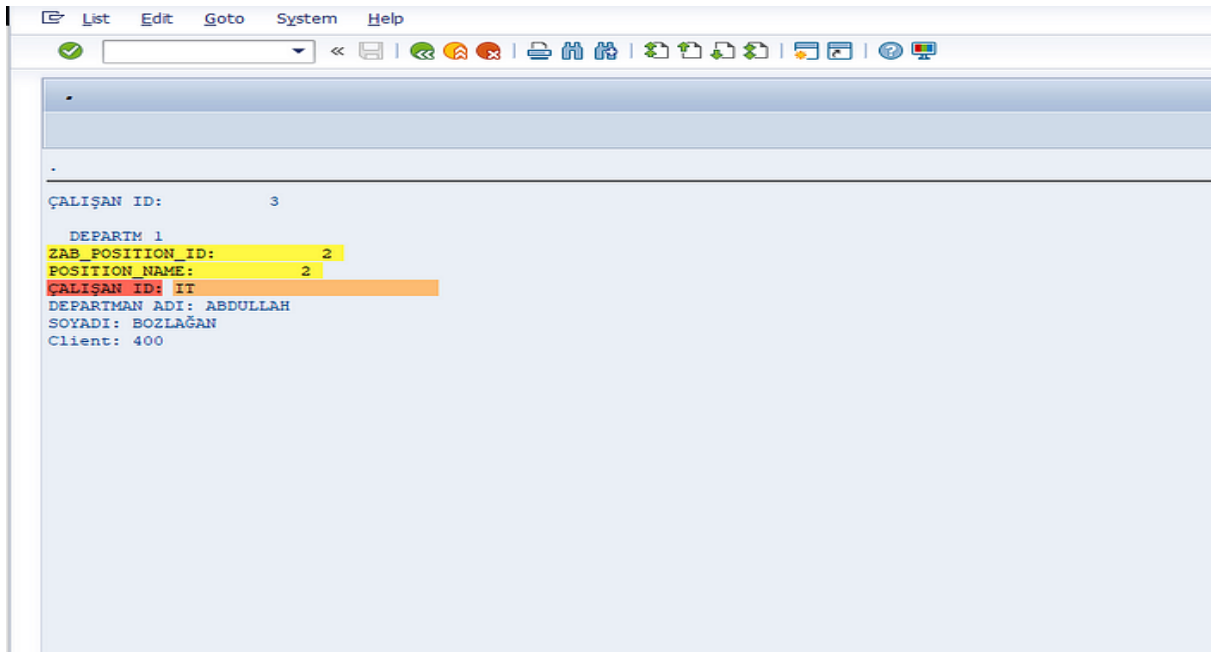
```
ÇALIŞAN ID:          2
```

Şekil 9. Sorgu Sonucu

CONDITION DEĞİŞİKLİĞİ



Şekil 9. Yeni Conditionlar



Şekil 10. Sorgu Sonucu

Görüldüğü üzere adı abdullah olan kişileri getiren yeni şartlanma sonucu output ekranımız bu şekilde değişti.