Bölüm6: İlişkisel Hesap

➢ Anlatım gücü açısından ilişkisel cebir ve ilişkisel hesap birbirine denktir. Bunların her ikisi de ilişkisel olarak yetkindir.

- ➢ İlişkisel hesap tanımlayıcı (*descriptive*) iken, ilişkisel cebir kural koyucudur (*prescriptive*). Bir başka deyişle ilişkisel cebir, ne kadar üst düzey olursa olsun, yordamsal (*procedural*) iken, ilişkisel hesap yordamsal değildir (*non procedural*).

➢ İlişkisel hesaba dayalı dillerin kuramsal temelleri:

- ◆ Çoklulara yönelik ilişkisel hesap (*tuple-oriented relational calculus*), ve
- ◆ Alanlara yönelik ilişkisel hesap (*domain-oriented relational calculus*)

olmak üzere ikiye ayrılır. Bu derste "çoklulara yönelik ilişkisel hesap" incelenecektir.

➢ İlişkisel hesap deyimininin tanımını verip, kullanım örneklerini incelemeden önce, niceleme mantığı ile ilgili önemli kavram ve kuralların bilinmesi gerekmektedir.

6.1. Niceleme Mantığı

6.1.1. Varlıksal Niceleyici ve Varlıksal Önerme

➢ Varlıksal niceleyici (*existential quantifier*), "∃" simgesi ile gösterilir ve "vardır, bulunur (*their exists*)" biçiminde okunur.

➢ Eğer x bir değişkeni, Fx de x değişkenine bağlı bir önermeyi gösteriyorsa, x'in olurlu her değeri için Fx önermesinin değeri doğru ya da yanlış olur.

➢ Varlıksal önerme genel olarak:

$$\exists x\ Fx$$

biçiminde gösterilir.

➢ Örnek: $\exists x$ (x.yaşı > 15)

➢ Eğer x'in olurlu değerlerinden (alabileceği değerlerden) en az biri için Fx önermesi doğru ise, $\exists x\ Fx$ varlıksal önermesi de doğrudur.

6.1.2. Tümel Niceleyici ve Tümel Önerme

➢ Tümel niceleyici (*universal quantifier*) " √ " simgesi ile gösterilir ve "tümü/her biri için (*for all, for each*)" anlamını taşır.

➢ Bir değişkene bağımlı önermenin (Fx) başına tümel niceleyici konularak tümel önerme oluşturulur: $\sqrt{x}\ Fx$

➢ Örnek: $\sqrt{x}$ (x.yaşı > 15)

➢ Eğer x'in tüm olurlu değerleri için Fx önermesi doğru ise, $\sqrt{x}\ Fx$ tümel önermesi de doğrudur.

6.1.3. Niceleyicilerin Değillenmesi ve Dağılması

➢ Niceleyicilerin değillenmesi ve dağılma özelliği ile ilgili eşitlikler.

1. $\neg\ \sqrt{x}\ Fx \equiv \exists x\ (\neg\ Fx)$
2. $\neg\ \exists x\ Fx \equiv \sqrt{x}\ (\neg\ Fx)$
3. $\sqrt{x}\ Fx \equiv \neg\ \exists x\ (\neg\ Fx)$
4. $\exists x\ Fx \equiv \neg\ \sqrt{x}\ (\neg\ Fx)$
5. $\sqrt{x}\ (Fx \wedge Gx) \equiv (\sqrt{x}\ Fx) \wedge (\sqrt{x}\ Gx)$
6. $\exists x\ (Fx \vee Gx) \equiv (\exists x\ Fx) \vee (\exists x\ Gx)$
7. $\exists x\ (Fx \wedge Gx) \Rightarrow (\exists x\ Fx) \wedge (\exists x\ Gx)$
8. $(\sqrt{x}\ Fx) \vee (\sqrt{x}\ Gx) \Rightarrow \sqrt{x}\ (Fx \vee Gx)$

6.1.4. Birden Çok Niceleyicinin Birlikte Kullanılması

➢ Niceleyici içermeyen önermeye tekil (*singular*) önerme denir. Tekil önermelerin önüne niceleyici konularak tekil olmayan önermeler elde edilir.

- ➢ Tekil olmayan (niceleyici içeren) önermelerin önüne de niceleyici konulabilir.
- ➢ Birden çok niceleyici içeren önermelerde niceleyiciler genellikle önermenin başına toplanır.
- ➢ Örnekler:

1. $\exists x \exists y\ Fxy \equiv \exists y \exists x\ Fxy$
2. $\surd x \exists y\ Fxy$
3. $\exists x \surd y\ Fxy$
4. $\surd x \surd y\ Fxy \equiv \surd y \surd x\ Fxy$
5. $\exists x \exists \neg y\ Fxy \equiv \exists x \surd y \neg Fxy$
6. $\surd x \exists \neg y\ Fxy \equiv \surd x \surd y \neg Fxy$
7. $\exists x \surd \neg y\ Fxy \equiv \exists x \exists y \neg Fxy$
8. $\surd x \surd \neg y\ Fxy \equiv \surd x \exists y \neg Fxy$
9. $\exists x \exists y \exists z\ Fxyz$

6.2. İlişkisel Hesap Deyimi

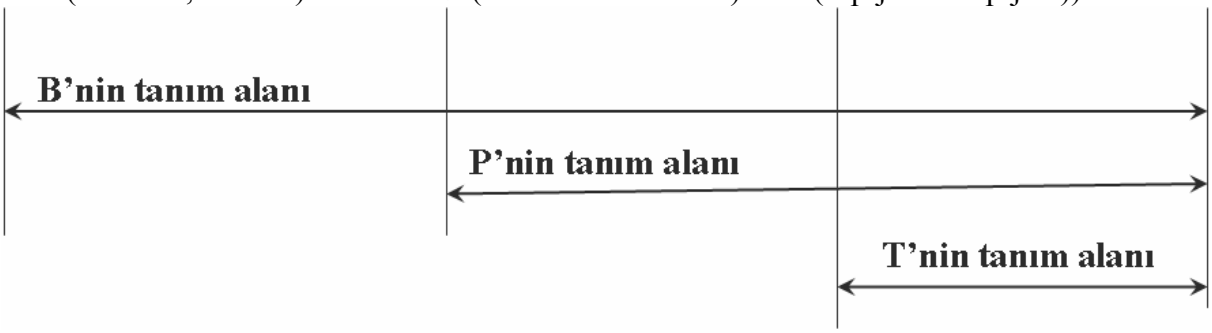
- Bir ilişkisel hesap deyimi, belirli ilişkilerden türetilecek yeni bir ilişkiyi tanımlar. Türetilecek yeni ilişki genellikle bir sorgunun yanıtını oluşturan verileri içerir. İlişkisel hesap deyimiyle yapılan tanım, yordamsal olmayan bir tanımdır.
- Genel yazılış: amaç-liste [WHERE önerme]
- Amaç-liste, bir "çoklu-değişkeni . nitelik" listesidir. Örneğin B, P ve M değişkenleri sırasıyla Bölüm, Personel ve Malzeme ilişkileri için tanımlanmış çoklu değişkenleri ise:

  B.bölNo, B.bölAdı
  P.sicNo, P.ücreti, P.adı
  B.bölNo, M.mKodu, M.mAdı
- Önerme bir WFF'dir (*Well-Formed Formula*) ve özyineli olarak tanımlanır:
- Önerme tanımı:
  1. X, Y, Z, .. çoklu değişkenleri; a, b, c, .. nitelik adları; k bir değişmez; θ ise bir aritmetik karşılaştırma işleci olmak üzere: X.a θ Y.b, X.a θ k ve k θ Y.b birer önermedir.
  2. Eğer F ve G birer önerme ise: (F) , $\neg F$, $F \wedge G$ ve $F \vee G$ de birer önermedir.
  3. Eğer Fx bir önerme ise: $\exists x\ Fx$ ve $\surd x\ Fx$ de birer önermedir.
- Geçerli önerme örnekleri:
  1. X.a = Y.b
  2. $(X.a = 5) \vee (X.b = Y.b)$
  3. $((X.a > 5) \wedge (Y.b = Y.c))$
  4. $\neg ((X.a = 3) \vee \exists Y\ (Y.b = 3))$
  5. $\exists X\ ((X.a = Y.b) \wedge (Y.c = 'K3'))$
  6. $\surd X\ ((X.a < 7) \vee (X.a >= 3))$
  7. $\surd X \exists Y\ ((X.a = Y.b) \wedge (Y.c = 'DB65'))$

Serbest ve sınırlı çoklu değişkenleri

- İlişkisel hesap deyimin tümünde tanımlı olan serbest çoklu değişkeni hem amaç listede hem de önermede yer alabilir. Serbest çoklu değişkeni bir niceleyicinin önünde yer alamaz. İstenirse ilişkinin adı bu ilişki için bir serbest çoklu değişkeni olarak kullanılabilir.
- Sınırlı çoklu değişkeni amaç listede yer alamaz, yalnız önermede yer alabilir. Deyimde ilk kez bir niceleyiciden sonra yer alır ve bu niceleyicinin sağında tanımlıdır. İlişkinin adı sınırlı çoklu değişkeni olarak kullanılamaz.
- Değişkenlerin Tanım Alanlarının Belirlenmesi

  (B.bölNo, B.bAdı) WHERE $\surd P(P.bölNo \neq B.bölNo) \vee \exists T(T.prjNo = P.prjNo)$



➢ Yalnız serbest çoklu değişkenleri içeren (hiç niceleyici içermeyen) bir ilişkisel hesap deyiminin anlamı: değişkenlerin gösterdiği çokluların her birleşimi için, önerme sağlanıyorsa, amaç listede tanımlanan çoklu sonuç ilişkiye konulur.

➢ Geçerli İlişkisel Hesap Deyimleri

1. (X.a, X.b) (X : serbest değişken)
2. (X.a, X.b, Y.c) (X ve Y serbest değişken)
3. (X.a, X.b, Y.c) WHERE (X.b = Y.d) (X ve Y serbest değişken)
4. (X.a, X.b) WHERE ∃Y (Y.c = X.b) (X serbest, Y sınırlı değişken)
5. (X.a, Y.b) WHERE √Z ∃W ((Z.e > X.a) ∧ (Z.f = W.g) ∧ (W.g = Y.c))) (X ve Y serbest, Z ve W sınırlı değişken

➢ Geçersiz İlişkisel Hesap Deyimleri

1. (X.a, X.b) WHERE Y.c < 3
2. (X.a, Y.b) WHERE ∃X (X.c < 5)

6.3. Yalın Bir İlişkisel Hesap Dili

➢ İlişkisel hesap deyimlerinin seçme gücünü göstermek ve sorguların ilişkisel hesap deyimleriyle anlatılmasını öğretmek için ALPHA dilinde yer alan deyimlerden yalnız ikisinin yer aldığı yalın bir dil kullanılacaktır.

6.3.1. RANGE Deyimi

➢ RANGE deyimi çoklu değişkenlerini tanımlamak için kullanılacaktır. Serbest çoklu değişkenleri için istenirse ilişki adları kullanılabilir. İstenirse de RANGE deyimi ile yeni değişkenler tanımlanır. Sınırlı çoklu değişkenlerinin RANGE deyimi ile tanımlanması zorunludur.

➢ Genel yazılış: RANGE OF <değişken-adı> IS <ilişki-adı> [{ SOME / ALL }] ;

➢ Örnekler: RANGE OF P IS Personel;
RANGE OF M1 IS Malzeme SOME;
RANGE OF PJ IS Proje ALL;
RANGE OF Çalışan IS Personel;

6.3.2. GET Deyimi

➢ GET deyiminin sözdizimi, ilişkisel hesap deyiminin önüne GET sözcüğü eklenerek oluşturulmuş ve böylece ilişkisel hesap deyimine bir komut yapısı kazandırılmıştır.

➢ Genel yazılış: GET amaç-liste [WHERE önerme];

➢ Tanımı verilen RANGE ve GET deyimlerinden oluşan komut kümesi, tüm uygulama gereksinimlerini karşılayan yetkin bir işlem kümesi oluşturmaz. Ancak yalnız bu iki deyim kullanılarak oluşturulacak ilişkisel hesap deyimleri, ilişkisel cebir işlemleri ile anlatılabilen tüm sorguları ifade etmek için yeterlidir.

6.4. İlişkisel Hesap Deyimleri İle Sorgu Örnekleri

➢ Sorgu örneklerinde, bir önceki bölümde tanımlanan ve ilişkisel şeması verilen Sanayi Veri Tabanı kullanılacaktır.

➢ Sorgular için verilecek çözümler örnek çözümler olacaktır. Aynı sorgular için genellikle farklı çözümler bulmak olanaklıdır.

➢ **SÖ.6.1.** "Bölümlerin numaralarını ve adlarını bul".
GET Bölüm.bölNo, Bölüm.bölAdı;

➢ **SÖ.6.2.** "Kurumda üretilen malzemelerin kodlarını bul".
GET (Üretim.mKodu);

➢ **SÖ.6.3.** "Firmaların ad ve adreslerini bul".
RANGE OF F IS Firma;
GET F.fAdı, F.adresi;

➢ **SÖ.6.4.** "Ücreti 5000'den küçük olan personelin sicil numarasını ve adını bul".
GET (Personel.sicNo, personel.adı) WHERE Personel.ücreti < 5000;

➢ **SÖ.6.5.** "7 numaralı projede en az 5 adet tüketilen malzemelerin kodlarını bul".

RANGE OF X IS Tüketim;
GET (X. mKodu) WHERE ((X.prjNo = 7) ∧ (X.miktar > 4));

➢**SÖ.6.6.** "Bölümlerin adları ile bölüm yöneticilerinin adlarını bul".

GET (Bölüm.bölAdı, Personel.adı) WHERE Bölüm.yönSicNo = Personel.sicNo;

➢ **SÖ.6.7**. "DB3967 kodlu malzemeyi üreten bölümlerin numara ve adları ile üretim miktarlarını bul".

RANGE OF B IS Bölüm;
RANGE OF Ü IS Üretim;
GET (B.bölNo, B.bölAdı, Ü.miktar) WHERE
((Ü.bölNo = B.bölNo) ∧ (Ü.mKodu = 'DB3967'));

➢ **SÖ.6.8.** "DB3967 kodlu malzemeyi üreten bölümlerin numara ve adlarını bul".

RANGE OF B IS Bölüm;
RANGE OF Ü IS Üretim;
GET (B.bölNo, B.bölAdı) WHERE
∃Ü((Ü.bölNo = B.bölNo) ∧ (Ü.mKodu = 'DB3967'));

Not: Bir önceki sorguda Ü amaç listede yer alıyordu ve bir serbest değişken idi. Bu sorguda ise Ü amaç listede yer almamaktadır. Ü bir sınırlı değişkendir ve önünde bir niceleyici bulunmaktadır.

➢ **SÖ.6.9.** "BilSis adlı firmadan satın alınan malzemelerin kodlarını bul".

RANGE OF F IS Firma;
GET (Alım.mKodu) WHERE (F.fNo = Alım.fNo) ∧ (F.fAdı = 'BilSis'));

➢ **SÖ.6.10.** "BilSis adlı firmadan satın alınan malzemelerin kod ve adlarını bul".

RANGE OF M IS Malzeme;
RANGE OF A IS Alım;
RANGE OF F IS Firma;
GET (M.mKodu, M.mAdı) WHERE ((M.mKodu = A.mKodu) ∧
(A.fNo = F.fNo) ∧ (F.fAdı = 'BilSis'));

➢ **SÖ.6.11.** "Kaya Mert adlı personelin çalıştığı bölümün yöneticisinin adını bul".

RANGE OF P IS Personel;
RANGE OF Y IS Personel;
RANGE OF B IS Bölüm;
GET (Y.adı) WHERE ∃P ∃B ((Y.sicNo= B.yönSicNo) ∧
(B.bölNo = P.bölNo) ∧ (P.adı = 'Kaya Mert'));

➢ **SÖ.6.12.** "TK0542 kodlu malzemeyi tüketmeyen projelerin adlarını bul".

RANGE OF T IS Tüketim;
GET (PROJE.prjAdı) WHERE ∃¬T ((T.prjNo = Proje.prjNo) ∧
(T.mKodu = 'TK0542'));

Bu sorguda, GET deyimi, tümel niceleyici kullanılarak aşağıdaki gibi de yazılabilir.

GET (Proje.prjAdı) WHERE √T ((T.prjNo ≠ Proje.prjNo) ∨
(T.mKodu ≠ 'TK0542'));

➢ **SÖ.6.13.** "3 numaralı projede tüketilen malzemelerden en az birini tüketen (3 numaralı proje ile tükettikleri ortak bir malzeme bulunan) projelerin adlarını bul".

RANGE OF T1 IS Tüketim;
RANGE OF T2 IS Tüketim;
GET (Proje.prjAdı) WHERE ∃T1 ∃T2 ((T1.prjNo = Proje.prjNo) ∧
(T1.mKodu = T2.mKodu) ∧ (T2.prjNo = 3));

➢ **SÖ.6.14.** "Mevcut miktarı sıfır olan ve kurumda üretilmeyen malzemelerin kod ve adlarını bul".

RANGE OF M IS Malzeme;
RANGE OF Ü IS Üretim;

GET (M.mKodu, M.mAdı) WHERE (M.mevMik = 0) ∧
∃¬Ü (Ü.mKodu= M.mKodu);

GET deyiminin yukarıdaki yazılışı doğrudur. Ancak niceleyiciyi önermenin başına çekerek deyimi aşağıdaki gibi yazmak da mümkündür.

GET (M.mKodu, M.mAdı) WHERE ∃¬Ü ((M.mevmik = 0) ∧
(Ü.mKodu = M.mKodu));

➢ **SÖ.6.15.** "Satın alınan tüm malzemeleri (her bir malzemezi) tüketen projelerin numara ve adlarını bul".

RANGE OF T IS Tüketim;
RANGE OF A IS Alım;
GET (Proje.prjNo, Proje.prjAdı) WHERE
√A ∃T ((Proje.prjNo = T.prjNo) ∧ (T.mKodu = A.mKodu));

➢ **SÖ.6.16.** "6 numaralı firmadan satın alınan tüm malzemeleri tüketen (tükettiği malzemeler arasında 6 numaralı firmadan satın alınan malzemelerin tümü bulunan) projelerin numara ve adlarını bul".

RANGE OF A IS Alım;
RANGE OF T IS Tüketim;
GET (Proje.prjNo, Proje.prjAdı) WHERE √A ((A.fNo ≠ 6) ∨
∃T ((T.prjNo = Proje. prjNo) ∧ (T.mKodu = A.mKodu)));

Aşağıda bu sorgu için iki seçenek daha sunulmaktadır:

a) Niceleyiciler önermenin başında ardarda yazılabilir.

RANGE OF A IS Alım;
RANGE OF T IS Tüketim;
GET (PROJE.PNO, PROJE.ADI) WHERE √A ∃T ((A.FNO ≠ 6) ∨
((T.PNO = PROJE. PNO) ∧ (T.MKODU = A.MKODU)));

b) İçerme (*implication*) mantıksal işleci kullanılarak GET deyimini aşağıdaki gibi yazmak da olanaklıdır.

RANGE OF A IS Alım;
RANGE OF T IS Tüketim;
GET (Proje.prjNo, Proje.prjAdı) WHERE √A ((A.fNo = 6) →
∃T ((T.prjNo = Proje. prjNo) ∧ (T.mKodu = A.mKodu)));

Not: a → b diye yazılan ve (*if a then b*) anlamını taşıyan içerme işlemi diğer mantıksal işlemler cinsinden aşağıdaki gibi ifade edilir: a → b = a' + b

➢ **SÖ.6.17.** "Satın alınan hiçbir malzemeyi tüketmeyen (tükettiği malzemeler arasında satın alınan malzemelerden hiçbiri yer almayan) projelerin numara ve adlarını bul".

RANGE OF J IS Proje;
RANGE OF T IS Tüketim;
RANGE OF A IS Alım;
GET (J.prjNo, J.prjAdı) WHERE
∃¬A ∃T ((T.prjNo = J.prjNo) ∧ (T.mKodu = A.mKodu));

Bu sorgu için GET deyimi aşağıdaki gibi de yazılabilir.

GET (J.prjNo, J.prjAdı) WHERE √T ((T.prjNo ≠ J.prjNo) ∨
∃¬A ((A.mKodu = T.mKodu) ∧ (T.prjNo = J.prjNo)));

➢ **SÖ.6.18.** "Yöneticisinden daha yüksek ücret alan personelin bulunduğu bölümlerin numara ve adlarını bul".

RANGE OF B IS Bölüm;
RANGE OF P IS Personel;
RANGE OF Y IS Personel;

GET (B.bölNo, B.bölAdı) WHERE ∃P ∃Y ((P.bölNo = B.bölAdı) ∧
(Y.sicNo = B.yönSicNo) ∧ (P.ücreti > Y.ücreti));

➢ **SÖ.6.19.** "Yapıldığı bölümde üretilenler dışında malzeme tüketmeyen (tükettiği malzemelerin tümü yapıldığı bölümde üretilen malzemeler olan) projelerin numara ve adlarını bul".

RANGE OF J IS Proje;
RANGE OF Ü IS Üretim;
RANGE OF T IS Tüketim;
GET (J.prjNo, J.prjAdı) WHERE √T ((T.prjNo ≠ J.prjAdı) ∨
∃Ü ((Ü.mKodu = T.mKodu) ∧ (Ü.bölNo = J.bölNo)));

Not: Niceleyicileri önermenin başına toplayarak, ya da yalnız varlıksal niceleyiciler kullanarak GET deyimini aşağıdaki gibi da yazabiliriz.

GET (J.prjNo, J.prjAdı) WHERE √T ∃Ü ((T.prjNo ≠ J.prjNo) ∨
((Ü.mKodu = T.mKodu) ∧ (Ü.bölNo = PJ.bölNo)));
GET (J.prjNo, J.prjAdı) WHERE ∃¬T ((T.prjNo = J.prjNo) ∧
∃¬Ü ((Ü.mKodu = T.mKodu) ∧ (Ü.bölNo = J.bölNo)));

➢ **SÖ.6.20.** "En az iki proje yöneten kişilerin sicil numarasını ve adını bul".

RANGE OF P IS Personel;
RANGE OF J1 IS Proje;
RANGE OF J2 IS Proje;
GET (P.sicNo, P.adı) WHERE ∃J1 ∃J2 ((J1.yönSicNo = P.sicNO) ∧
(J2.yönSicNo = P.sicNO) ∧ (J1.prjNo ≠ J2.prjNO);

Bölüm7: SQL Standart İlişkisel Dil

➢ SQL dilinin kökleri *SYSTEM R* adlı prototip Veri Tabanı Yönetim Sistemine kadar uzanır.
  ➢ Dilin adı : SQUARE → SEQUEL → SQL
    SEQUEL : Structured English QUEry Language
    SQL : SEQUEL'in okunuşu

➢ SQL dili ilişkisel Veri Tabanı Yönetim Sistemlerinin hemen hemen tümünde kullanılan standart bir dil niteliği kazanmıştır.

➢ İngilizceye benzer yapısal bir dil olan SQL oldukça kapsamlı standart bir dildir. Bu derste SQL dilinin veri tanımlama ve veri seçme amaçlı başlıca komutlarının temel yapılarının incelenmesi ve sorgu örnekleri ile dilin olanaklarının gösterilmesi ile yetinilecektir.

➢ ISO (*International Standart Organization*) tarafından yayınlanan SQL standartları:
- 1992 yılında yayınlanan SQL-92 ya da SQL2.
- 1999 yılında yayınlanan SQL:1999 ya da SQL3 (*regular expression matcing, recursive queries, triggers, non-scalar types and some object-oriented features*)
- 2003 yında yayınlanan SQL:2003 (*XML-relqated features, window functions, standardized sequnces, colums with auto-generated values (including idendity columns*)

➢ 7.1. Veri Tanımlama Olanakları

➢ SQL dilinin veri tanımlama amacıyla kullanılan deyimlerden birkaçı aşağıdakilerdir.

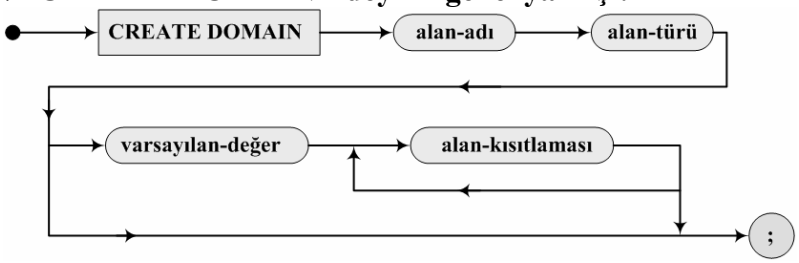
CREATE DOMAIN	ALTE R DOMAIN	DROP DOMAIN
CREATE TABLE	ALTER TABLE	DROP TABLE
CREATE VIEW	ALTER VIEW	DROP VIEW
CREATE INDEX	ALTER INDEX	DROP INDEX
CREATE SNAPSHOT	ALTER SNAPSHOT	DROP SNAPSHOT

  ➢ CREATE deyimleri ile sırasıyla alan, çizelge, görünüm, dizin ve anlık-çizelge tanımları yapılır.

- ALTER deyimleri ile daha önce yapılmış olan tanımlarda değişiklik yapılır.
- DROP deyimleri ile de daha önce yapılmış tanımlar yok edilir.
- Bu derste yalnız CREATE DOMAIN, CREATE TABLE, CREATE VIEW, CREATE INDEX ve CREAT SNAPSHOT deyimleri tanıtılacaktır.

7.1.1. CREATE DOMAIN Deyimi

- SQL standartlarındaki alan kavramı, ilişkisel kuramdaki alan kavramına göre oldukça dar kapsamlı bir kavramdır. SQL-2'de gerçek anlamda kullanıcı tarafından tanımlanan (*user-defined*) alanlar oluşturmak mümkün değildir. SQL-2'de alan tanımlama olanağı, sistem tarafından tanımlanmış hazır (*built-in*) veri türleri cinsinden yeni tür tanımları yapılması ve tanımlanan yeni veri türlerinin birçok nitelik için kullanılabilmesi ile sınırlıdır
- **CREATE DOMAINE deyimi genel yazılışı:**



- alan-türü : tanımlanan alanın, hazır veri türleri ya da daha önce tanımlanmış alanlar cinsinden tür tanımı

  varsayılan-değer-tanımı : varsa, tanımlanan alanın varsayılan-değeri (*default-value*)

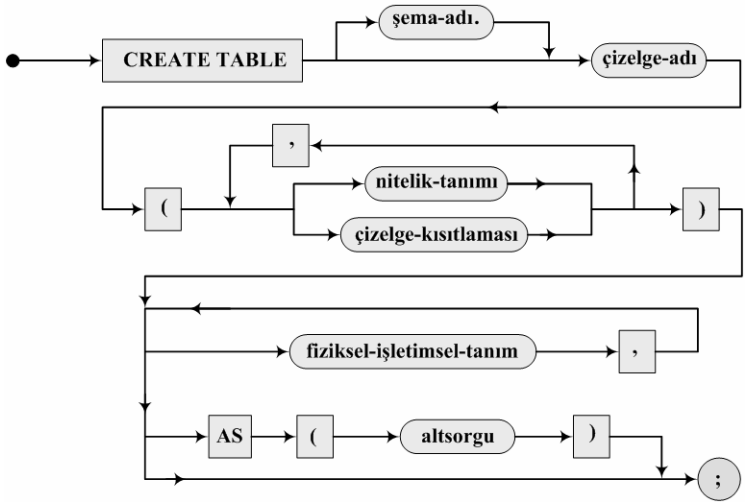
  alan-kısıtlaması : varsa, tanımlanan alanla ilgili kısıtlamalar (*costraints*).

- **Örnekler**:

1. CREATE DOMAIN renk CHAR(8) DEFAULT '???'
   CHECK (VALUE IN ('Kırmızı', 'Mavi', 'Sarı', 'Yeşil', '???'));
2. CREATE DOMAIN cinsiyet CHAR(5) DEFAULT '-'
   CHECK (VALUE IN ('Erkek', 'Kadın', '-'));

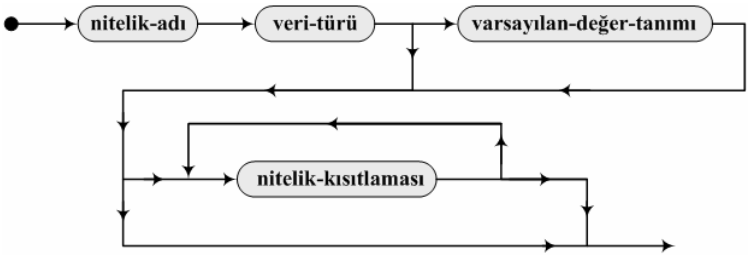
7.1.2. CREATE TABLE Deyimi

- Veri tabanındaki temel çizelgeler (*base relations*) CREATE TABLE deyimi ile tanımlanır. Herhangi bir niteleme yapmadan sadece "ilişki" ya da "çizelge" denildiğinde veri tabanının temel çizelgeleri anlaşılır.
- Temel çizelgeler dışında veri tabanında türetilmiş ilişkiler (*derived relations*) de yer alır. Türetilmiş ilişki kavramı, CREATE VIEW deyimi kapsamında ele alınacaktır.
- SQL'deki çizelge ya da ilişki kavramı ilişkisel kuramdaki ilişki kavramından kimi farklılıklar gösterir. Bunların en önemlisi SQL çizelgelerinde tekrarlı satırların bulunabilmesidir. Bunun tanımlama düzeyindeki anlamı, SQL çizelgelerinde anahtar adayı (*candidate key*) bulunmasının zorunlu olmamasıdır.
- CREATE TABLE deyiminin genel yazılışı aşağıda verilecektir. Ancak verilecek genel yazılış tüm seçenekleri içeren eksiksiz bir tanım olmayacaktır. Genel yazılışta daha çok kavramsal çizelge tanımı ile ilgili önemli seçeneklere yer verilecektir.
- **CREATE TABLE deyimi genel yazılışı:**

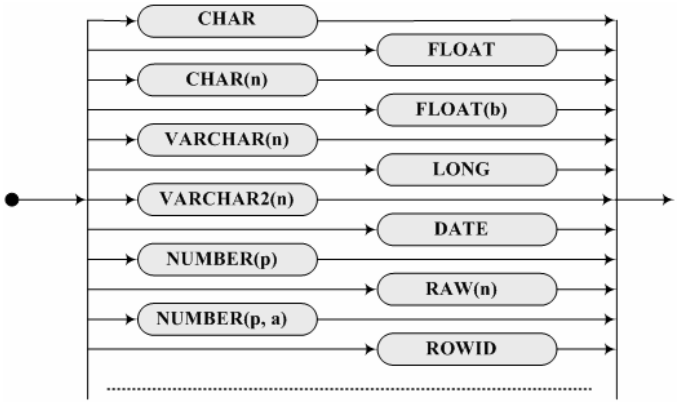


altsorgu : SELECT deyimi ile tanımlanabilen herhangi bir SQL sogusudur.

➢ **nitelik-tanımı ::=**



➢ **veri-türü ::=**



➢**Veri türü.** Ticari VTYS'lerin veri türleri ANSI SQL veri türlerinden farklı olabilmektedir. Aşağıda ANSI SQL-2 veri türleri ile ORACLE-8 veri türleri açıklanmakta daha sonra bu veri türlerinin karşıkları verilmektedir.

ANSI SQL-2 Veri Türleri

1. CHARACTER(n) (n > 0)
2. CHARACTER VARYING(n) (n > 0)
3. BIT(n) (n > 0)
4. BIT VARYING(n) (n > 0)
5. NUMERIC(p) (p > 0)
6. NUMERIC(p,q) (p > 0 , 0 ≤ q ≤ p)
7. DECIMAL(p) (p > 0).
8. DECIMAL(p,q) (p > 0 , 0 ≤ q ≤ p).
9. INTEGER
9. SMALLINT
10. FLOAT(p)
11. REAL
12. DOUBLE PRECISION

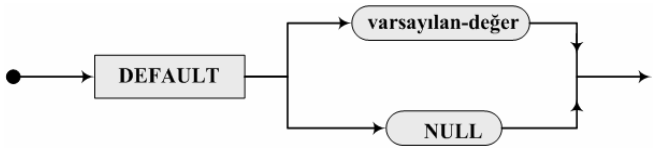
➢**ORACLE-8 Veri Türleri**

1. VARCHAR2(n) (1 ≤ n ≤ 2000).
2. CHAR
3. CHAR(n) (1 ≤ n ≤ 255).
4. NUMBER(p) (1 ≤ p ≤ 38).
5. NUMBER(p,s) 1 ≤ p ≤ 38
6. NUMBER
7. LONG (en çok 1 GByte)
8. RAW(n) (1 ≤ n ≤ 255)
9. ROWID.
10. DATE
11. FLOAT (ondalık duyarlığı 38, ikili duyarlığı ise 128)
12. FLOAT(b) İkili duyarlığı b olan kayan noktalı sayı.

➢ **ANSI SQL-2 Veri Türlerinin Karşılıkları Olan ORACLE-8 Veri Türleri**

ANSI SQL-2 Veri Türü	**ORACLE-8 Veri Türü**
CHARACTER(n) / CHAR(n)	CHAR(n)
CHARACTER VARYING(n) / CHAR VARYING(n)	VARCHAR2(n) /VARCHAR(n)
NUMBER(p,s) ve DECIMAL(p,s)	NUMBER(p,s)
INTEGER / INT / SMALLINT	NUMBER(38)

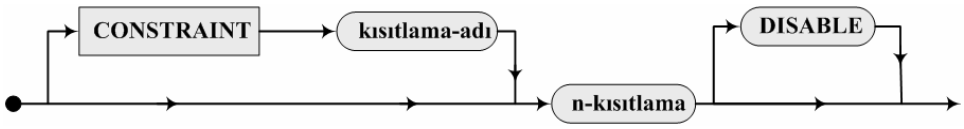
➢ **varsayılan-değer-tanımı ::=**



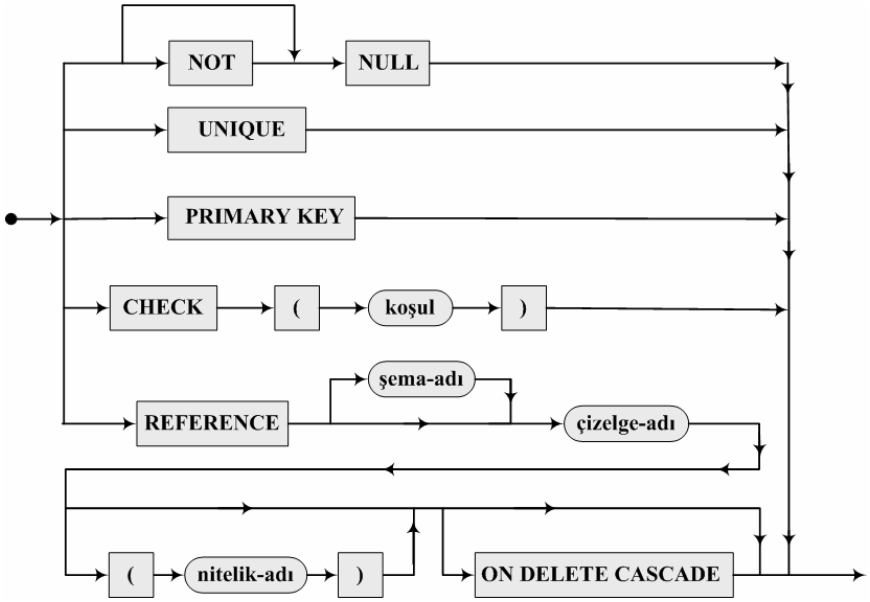
Varsayılan-değer-tanımı, INSERT deyimi ile çizelgeye yeni bir satır eklenirken, ilgili niteliğin değeri belirtilmemişse, niteliğe verilecek değeri tanımlamak için kullanılır. Varsayılan değer, niteliğin türü ile uyumlu bir değer olabileceği gibi, NULL da olabilir.

➢ Örnekler: adı CHAR(16) DEFAULT '???' …..
yarıyıl CHAR(11) DEFAULT '1999-2000 Güz' …..
adres VARCHAR(60) DEFAULT NULL …..

➢ **nitelik-kısıtlaması ::=**



➢ **n-kısıtlama ::=**



➢ **Nitelik kısıtlaması :**

Aşağıda, örnek olarak, ORACLE-8'de bulunan nitelik kısıtlamaları verilmektedir.

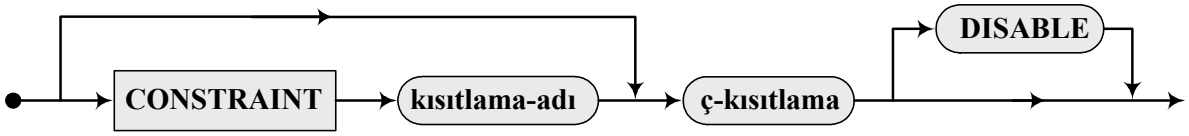
1. [NOT] NULL: İlgili niteliğin değerinin eksik ya da belirsiz (NULL) olup olamıyacağını belirtir. Bu kısıtlama yazılmazsa, ilgili niteliğin değeri NULL olabilir.
2. UNIQUE: İlgili niteliğin bir anahtar adayı olduğunu belirtir.
3. PRIMARY KEY: İlgili niteliğin bir anahtar adayı olduğunu belirtir. Bir çizelgenin birden çok birincil anahtarı olamaz.
4. REFERENCE ... : İlgili niteliğin çizelgede bir yabancı anahtar (*foreign key*) olduğunu ve adı belirtilen çizelgenin bir anahtarını referans gösterdiğini belirtir. Referans gösterilen anahtar ilgili çizelgede PRIMARY KEY ya da UNIQUE olarak tanımlanmış bir nitelik olabilir. Hangi niteliğin referans gösterildiği belirtilmezse, çizelgenin birincil anahtarının

referans gösterildiği varsayılır. ON DELETE CASCADE seçimi yazılırsa, referans gösterilen çizelgeden bir satırın silinmesi, bu satırı referans gösteren satırların da otomatik olarak silinmesine neden olur.

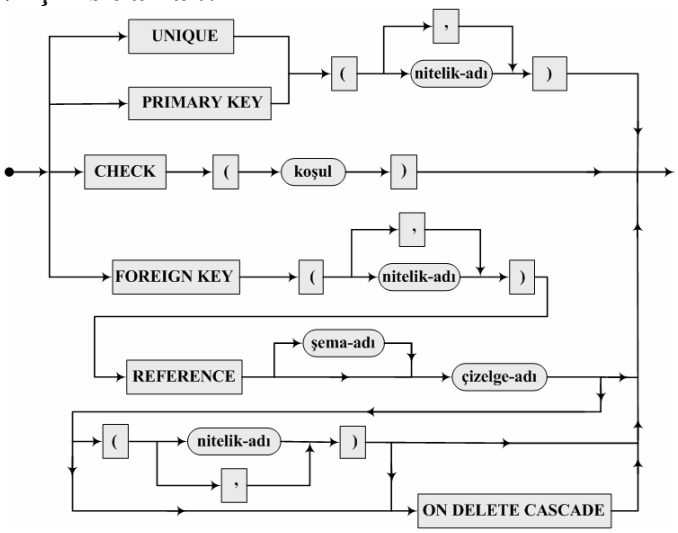
5. CHECK(koşul): İlgili nitelik tarafından çizelgenin tüm satırlarında sağlanması gereken bir koşulu belirtir. Bu koşulda, tanımı yapılan nitelik dışındaki bir niteliğin adı geçemez.

➢ **Çizelge kısıtlaması.** Veri tabanının tutarlılığı açısından, çizelgedeki bir ya da birden çok niteliğin değeri ile ilgili kısıtlamalara çizelge kısıtlaması diyoruz. Birden çok niteliği ilgilendiren kısıtlamalar çizelge kısıtlaması olarak tanımlanmalıdır. Her nitelik kısıtlaması yalnız bir niteliği ilgilendirdiği için, o niteliğin tanımının bir parçası olarak yazılır. Çizelge kısıtlamaları birden çok niteliği ilgilendirebildiği için, nitelik tanımlarından ayrı olarak yazılır.

➢ **çizelge-kısıtlaması ::=**



➢ **ç-kısıtlama ::=**

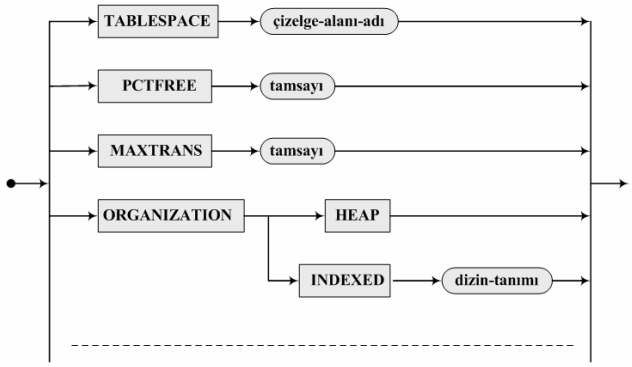


➢ **Çizelge Kısıtlaması:**

Aşağıda, örnek olarak, ORACLE-8'de bulunan çizelge kısıtlamaları verilmektedir.

1. UNIQUE(nitelik-1, ...) : İlgili nitelik ya da niteliklerin bir anahtar adayı olduğunu belirtir.
2. PRIMARY KEY (nitelik-1, ...) : İlgili nitelik ya da niteliklerin çizelgenin birincil anahtarı olduğunu belirtir. Bir çizelgenin birden çok birincil anahtarı olamaz.
3. FOREIGN KEY ...: İlgili nitelik ya da niteliklerin çizelgede bir yabancı anahtar (*foreign key*) olduğunu ve adı belirtilen çizelgenin bir anahtarını referans gösterdiğini belirtir. Referans gösterilen çizelgede hangi anahtarın referans gösterildiği belirtilmezse, çizelgenin birincil anahtarının referans gösterildiği varsayılır. Eğer bu kısıtlama ON DELETE CASCADE seçimi ile birlikte yazılırsa, referans gösterilen çizelgeden bir satırın silinmesi, bu satırı referans gösteren satırların da otomatik olarak silinmesine neden olur.
4. CHECK(koşul): Çizelgenin tüm satırlarında sağlanması gereken herhangi bir koşulu belirtir. Bu koşulda çizelgenin istenilen nitelikleri yer alabilir.

➢ **fiziksel-işletimsel-tanım ::=**



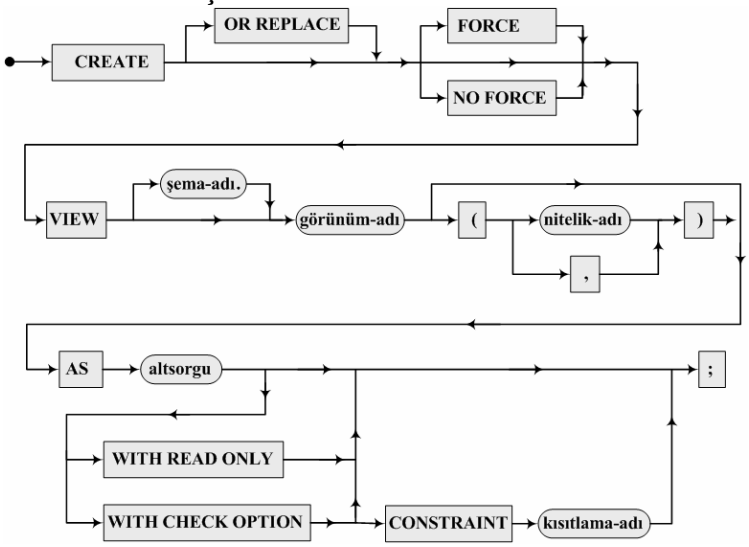
- Fiziksel-işletimsel-tanım'larla çizelgelerin fiziksel ve işletimsel özellikleri ile saklama düzeni ile ilgili çok sayıda tanım yapılabilir. Kullanılan Veri Tabanı Yönetim Sistemine göre değişen fiziksel-işletimsel-tanım öğelerinin tümü bu derste ele alınmamakta; tanım öğelerinden birkaç örnek vermekle yetinilmektedir.
  - "TABLESPACE çizelge-alanı-adı" ile, çizelgenin hangi çizelge alanında yer alacağı belirtilir.
- "PCTFREE tamsayı" tanımı ile çizelgenin veri öbeklerindeki boş alan yüzdesi belirtilir. Tamsayı değeri 0 - 99 arasında olabilir.
- "MAXTRANS tamsayı" tanımı ile, çizelge için ayrılmış bir veri öbeğinin aynı anda en çok kaç hareket (*transction*) tarafından günlenebileceği belirtilir. Tamsayı değeri 1 – 255 arasında olabilir.
- "ORGANIZATION HEAP" seçimi çizelge belekte saklanırken satırların herhangi bir sırada tutulmayacağını tanımlar. "ORGANIZATION INDEXED" seçimi ise çizelgenin bellekte dizinli kütük düzeninde saklanacağını gösterir.
  - **Örnek (Sanayi Veri Tabanı):**

```
CREATE TABLE Bölüm (bölNo NUMBER(4) PRIMARY KEY,
                    bölAdı  CHAR(24) NOT NULL,
                    yönSicNo NUMBER(6)  NULL REFERENCE Personel (sicNo))
                    TABLESPACE VTAlan1;
CREATE TABLE Personel (sicNo NUMBER (6) PRIMARY KEY,
                    adı VARCHAR2(36) NOT NULL,
                    ücreti NUMBER(4) CHECK(ücreti > 500 AND ücreti < 6001),
                    bölNo NUMBER(4) NOT NULL REFERENCE Bölüm)
                    TABLESPACE VTAlan1;
CREATE TABLE Proje (prjNo NUMBER(3) PRIMARY KEY,
                    prjAdı VARCHAR2(64) NOT NULL,
                    bölNo NUMBER(4) NOT NULL REFERENCE Bölüm(bölNo),
                    yönSicNo NUMBER(6) NULL REFERENCE Personel(sicNo))
                    TABLESPACE VTAlan1;
CREATE TABLE Malzeme (mKodu CHAR(6) PRIMARY KEY,
                     mAdı  VARCHAR2(64) NOT NULL,
                     mevMik NUMBER(9,3) NOT NULL CHECK(mevMik >= 0))
                    TABLESPACE VTAlan2;
CREATE TABLE Üretim(bölNo NUMBER(4)NOT NULL REFERENCE Bölüm(bölNo),
                     mKodu CHAR(6) NOT NULL REFERENCE Malzeme(mKodu),
                     miktar NUMBER(9,3) CHECK (miktar >= 0),
                     PRIMARY KEY(bölNo, mKodu))
                     TABLESPACE VTAlan2;
CREATE TABLE Tüketim
                    (prjNo NUMBER(3) NOT NULL REFERENCE Proje(prjNo),
                     mKodu CHAR(6) NOT NULL REFERENCE Malzeme(mKodu),
                     miktar NUMBER(9,3) CHECK (miktar >= 0),
                     PRIMARY KEY (prjNo, mKodu))
                     TABLESPACE VTAlan1;
CREATE TABLE Firma (fNo CHAR(3) PRIMARY KEY,
                     fAdı  VARCHAR2(64) NOT NULL,
                     fAdresi VARCHAR2(80) NULL)
                    TABLESPACE VTAlan2;
CREATE TABLE Alım (fNo NUMBER (3) NOT NULL REFERENCE Firma(fNo),
                     mKodu CHAR(6) NOT NULL REFERENCE Malzeme(mKodu),
```

```
    miktar NUMBER(9,3) CHECK (miktar >= 0),
    PRIMARY KEY(fNo, mKodu))
    TABLESPACE VTAlan2;
```

7.1.3. CREATE VIEW Deyimi

- İlişkisel veri tabanlarında görünüm (*view*), bir ya da birkaç ilişki üzerinde tanımlanan ve fiziksel olarak varolmayan mantıksal bir ilişkiyi (ya da çizelgeyi) gösterir.
- Bir ya da birkaç ilişki üzerinde tanımlanan mantıksal bir pencere olarak da algılanabilecek görünüm aşağıdaki özellikleri taşır.
  - Görünüm veri içermez, tanımlandığı ilişkilerden elde edilen verileri gösterir.
  - Görünüm temel ilişkiler ve/ya da görünümler üzerinde tanımlanır.
  - Tüm görünümler SELECT deyiminde temel ilişkiler gibi kullanılabilir.
  - Her görünüm INSERT, UPDATE ve DELETE deyimlerinde kullanılamaz. Tanımında birleştirme (*join*) işleci, küme işleçleri, grup fonksiyonları, DISTINCT işleci, GROUP BY seçimi, CONNECT BY seçimi, ve START WITH seçimi yer almayan görünümler INSERT, UPDATE ve DELETE deyimlerinde kullanılabilir.
- Görünümlerin kullanım amaçlarından başlıcaları aşağıda sıralanmaktadır.
  - Temel ilişkilerin belirli satırlarını/kolonlarını kullanıcıdan gizlemek; ek bir güvenlik düzeyi oluşturmak.
  - Veri karmaşıklığını kullanıcıdan gizlemek; kullanıcıya daha basit yapılar sunmak.
  - Veriye değişik bir bakış açısı oluşturmak (izelge yapısını, niteliklerin adlarını ya da sıralarını değiştirmek, ...vb).
  - Sık kullanılan altsorguları birer görünüm olarak tanımlayarak, sorgu oluşturmayı kolaylaştırmak.
- **Genel Yazılış:**



- **CREATE VIEW** deyiminin genel yazılışı ile ilgili kısa açıklamalar aşağıda verilmektedir.
  - OR REPLACE seçimi ile, varsa mevcut görünümün tanımı değiştirilir.
  - FORCE seçimi yazılırsa, temel çizelge ve görünümlerin tanımlanmış olduğuna ve kullanıcının yetkilerine bakılmaksızın görünüm tanımı kabul edilir.
  - NO FORCE seçimi yazılırsa, temel çizelge ve görünümlerin tanımlanmış olduğuna ve kullanıcının yetkilerine bakılır (varsayılan değer NO FORCE).
  - Eğer tanımlanan görünümün nitelikleri, kaynak ilişkilerin/görünümlerin belirli niteliklerine eşit ise ve nitelik adlarının değiştirilmesi istenmiyorsa, görünüm tanımında nitelik adlarının belirtilmesine gerek yoktur.
  - Eğer görünümdeki niteliklerden biri bir deyim olarak tanımlanıyorsa ya da niteliklerden birine farklı bir ad verilmek isteniyorsa, görünüm tanımında nitelik

adlarının tanımlanması gereklidir.

- AS altsorgu görünümün tanımını oluşturur. Bu tanım, ORDER BY ya da FOR UPDATE seçimi içermeyen herhangi bir SELECT deyimi olabilir.
- WITH READ ONLY seçimi görünümün yalnız sorgularda kullanılacağını gösterir.
- WITH CHECK OPTION seçimi, yalnız görünümün kapsamına giren bir satır için ekleme, silme ve günleme yapılabileceğini gösterir.

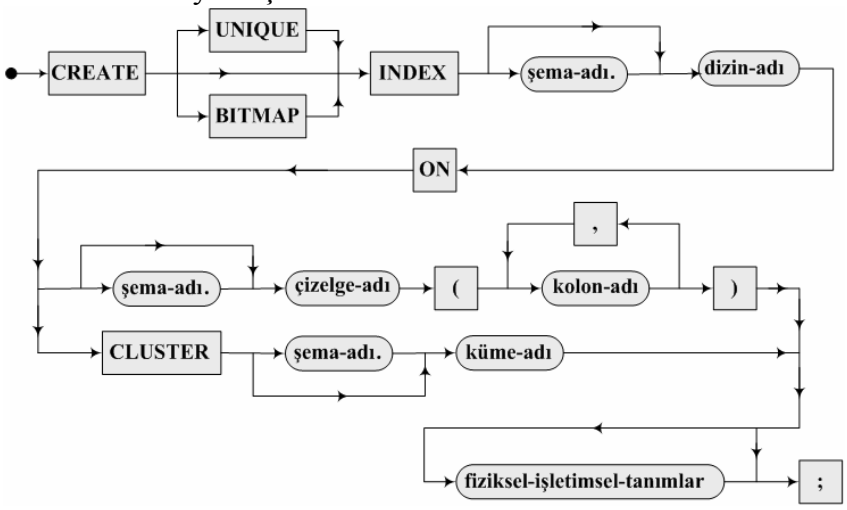
➢ **Örnekler:**

1. CREATE VIEW KMalzeme
   AS SELECT mKodu, mAdı FROM Malzeme
   WHERE mKodu NOT IN (SELECT mKodu FROM Tüketim);
   Bu örnekte, hiçbir projede tüketilmeyen malzemelerin kod ve adları seçilerek KMalzeme (kullanılmayan malzeme) adlı bir görünüm tanımlanmaktadır.
2. CREATE VIEW Yönetici (ySicNo, yAdı, yBölNo)
   AS SELECT sicNo, adı, bölNo FROM Personel
   WHERE sicNo IN (SELECT yönSicNo FROM Bölüm)
   OR sicNo IN (SELECT yönSicNo FROM Proje)
   WITH CHECK OPTION;
   Bu örnekte, bölüm ya da proje yöneticisi olan personelin sicil numarasını, adını ve bölümünün (kadrosunun bulunduğu bölümün) numarasını içeren YÖNETİCİ adlı bir görünüm tanımlanmaktadır.
3. CREATE VIEW Proje15(prjNo, tMalSay)
   AS SELECT prjNo, COUNT(mKodu) FROM Tüketim
   WHERE prjNo IN (SELECT prjNo FROM Proje WHERE bölNo = 15)
   GROUP BY prjNo
   WITH READ ONLY;
   Bu örnekte 15 numaralı bölümde yapılan projelerin numaraları ile bu projelerden her birinde tüketilen malzeme sayısını içeren Proje15 adlı bir görünüm tanımlanmaktadır.

7.1.3. CREATE INDEX Deyimi

➢ Çizelge ve kümelere (*cluster*) erişimde kullanılacak dizinleri yaratmak için kullanılır.

➢ Genel yazılış:



➢ Dizin yaratma ile ilgili seçenekler:

1. UNIQUE: dizin yaratılacak kolon ya da kolonlardaki değerlerin biricik (her satırda farklı) olduğunu gösterir.

2. BITMAP: Dizinin B-ağacı dizini değil bitmap dizini olacağını gösterir. *Bitmap* dizini *unique* dizinler için yaratılamaz. Başka bir deyişle UNIQUE ve BITMAP seçeneklerinden en çok biri kullanılabilir.

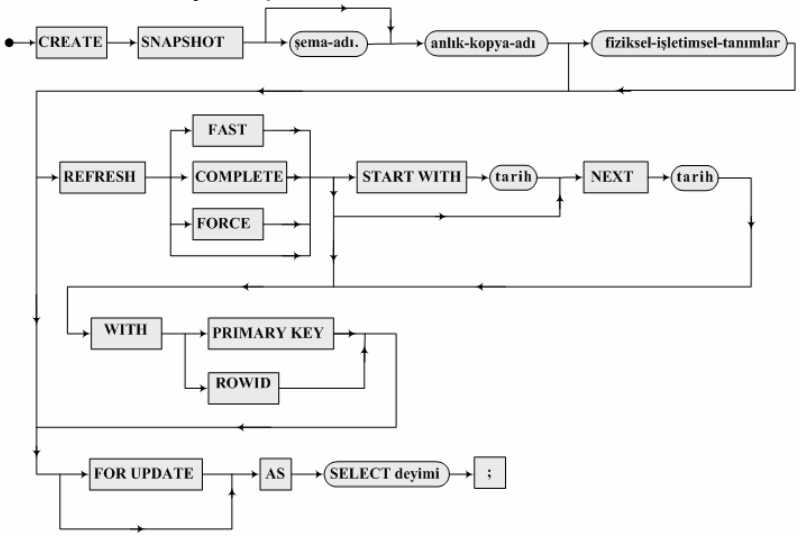
➢ Örnekler

1. CREATE UNIQUE INDEX sicNoDizin ON Personel (sicNO);
2. CREATE INDEX adDizin ON Personel (adı);
3. CREATE BITMAP INDEX ON Personel (bölNo);
4. CREATE UNIQUE INDEX ON Üretim (bölNo, mKodu);

7.1.4. CREATE SNAPSHOT Deyimi

➢ Anlık-kopya (*snapshot*) bir sorgunun sonucu olarak genellikle uzak bir sitedeki veri tabanında yaratılan ve zaman zaman günlenebilen bir çizelgedir.

➢ Genel yazılış:



➢ **Anlık-kopyanın (snapshot) türü ve güncellenmesi ile ilgili seçenekler:**

1. FAST: ana çizelgenin "snapshot log" kütüğü kullanılarak anlık çizelgenin hızlı güncellenmesi.
2. COMPLETE: anlık-kütüğün yaratılmasını sağlayan sorgu kullanılarak anlık-kopyanın güncel olarak yeniden yaratılması.
3. FORCE: mümkünse hızlı güncelleme, değilse komple güncelleme yapılması (*default*).
4. START WITH: Otomatik güncellemenin yapılacağı ilk tarihi belirtir.
5. NEXT: Otomatik güncellemeler arasındaki zaman aralığını belirtir. START WITH yazılır NEXT yazılmazsa otomatik güncelleme bir kez yapılır. START WITH yazılmaz, NEXT yazılırsa ilk güncelleme anlık-kopyanın yaratıldığı zamana göre NEXT kullanılarak belirlenir. START WITH ve NEXT seçeneklerinin her ikisi de yazılmazsa otomatik güncelleme yapılmaz.
6. WITH PRIMARY KEY: Anlık-koyanın birincil anahtar tabanlı yaratılcağını gösterir. Bu durumda ana çizelgenin yeniden düzenlenmesi anlık çizelgeyi etlkilemez.
7. WITH ROWID: Anlık-kopyanın ROWID tabanlı yaratılacağını belirtir.
8. FOR UPDATE: Anlık-kopya üzerinde güncelleme yapılabileceğini gösterir. Eğer bu seçenek *Replication* seçeneği ile birlikte kullanılırsa, anlık-kopya üzerinde yapılan günlemeler ana (*master*) kopyaya yansıtılır.

➢ Örnekler:

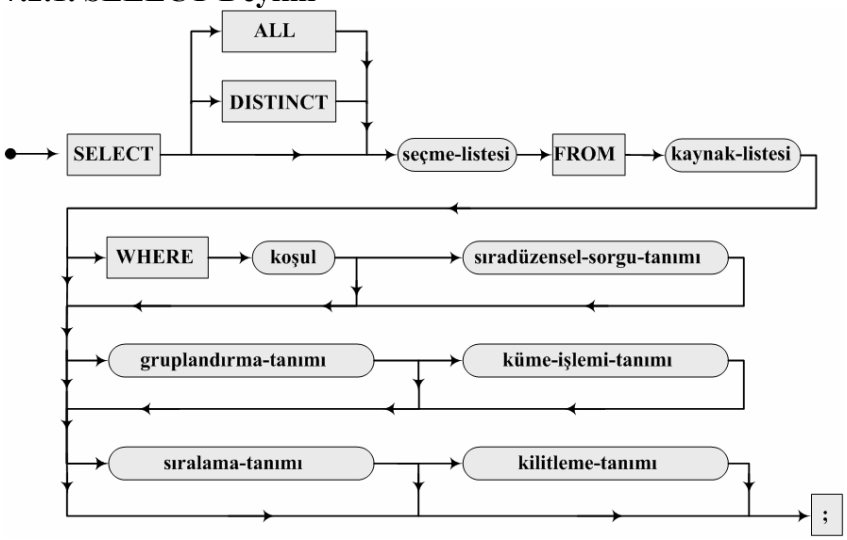
```
1. CREATE SNAPSOT perKopya
   REFRESH FAST NEXT sysdate + 7
   AS SELECT * FROM merkzVT.Personel@ankara;
2. CREATE SNAPSHOT üretimKopya
   REFRESH COMPLETE START WITH sydate NEXT sysdate + 7
   AS SELECT * FROM Üretim;
```

7.2. Veri İşleme Olanakları

➢ SQL'de çok sayıdaki veri tanımlama deyimine karşın, sınırlı sayıda veri işleme deyimi vardır. SQL DML deyimlerinin başlıcaları:

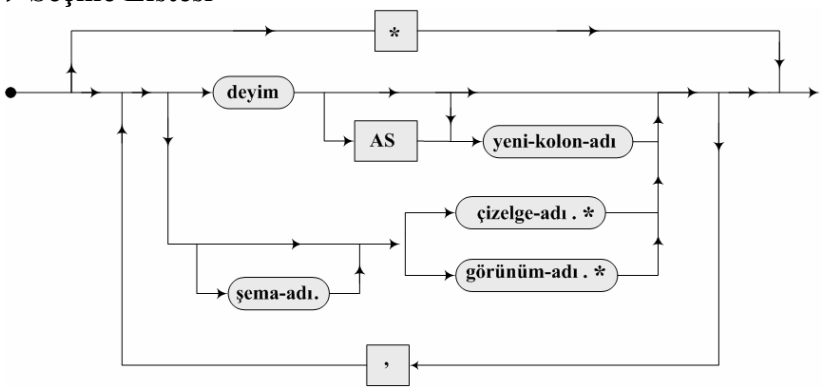
1. SELECT Deyimi
2. INSERT Deyimi
3. UPDATE Deyimi
4. DELETE Deyimi
4. LOCK TABLE Deyimi : Bir çizelge ya da görünümü kilitleyerek diğer kullanıcıların erişimini kısıtlamak için kullanılır.
5. EXPLAIN PLAN Deyimi : Bir SQL deyimi ile ilgili işletim planını öğrenmek için kullanılır.

7.2.1. SELECT Deyimi



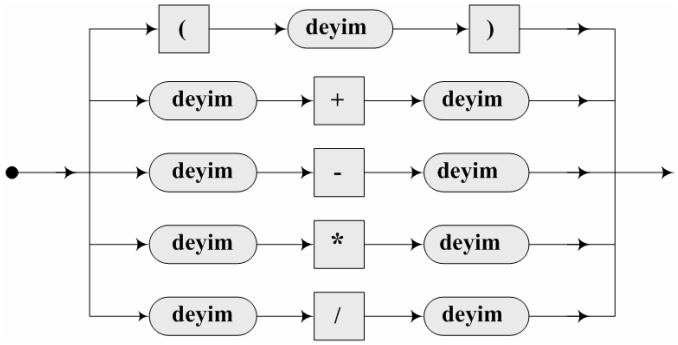
- SELECT deyiminin genel yazılışı ile ilgili açıklamalar aşağıda yer almaktadır.
- DISTINCT seçimi elde edilecek amaç çizelgede tekrarlı satırların atılması için yazılır. (varsayılan değer ALL'dur).

➢Seçme Listesi

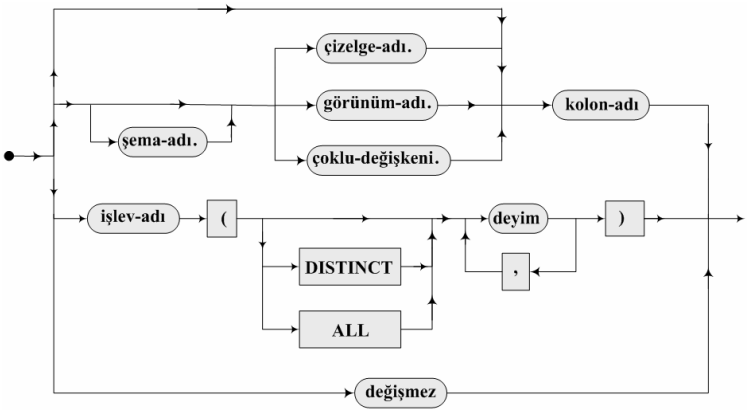


- Seçme-listesi ile sorgunun yanıtını oluşturacak amaç çizelgenin kolonları (nitelikleri) tanımlanır.
- Seçme-listesi olarak (*) yazılırsa, kaynak-listesindeki çizelgelerin tüm kolonları amaç çizelgede yer alır.
- Seçme-listesi olarak bir dizi amaç-kolon-tanımı yazılırsa, bunlardan her biri aşağıdaki türlerden birinde olabilir.
  a. [şema-adı.] çizelge-adı.*
  b. deyim [AS yeni-kolon-adı]
- Örnekler: 1. SELECT * FROM Personel ...
  2. SELECT Proje.*, Bölüm.adı FROM Proje, Bölüm ...
  3. SELECT sicNo, adı, ücreti FROM Personel ...
  4. SELECT sicNo, (ücreti * 0.25) AS vergi FROM Personel ...
  5. SELECT mKodu, mAdı, GREATEST(mevMik/2, 100) AS sipMik...

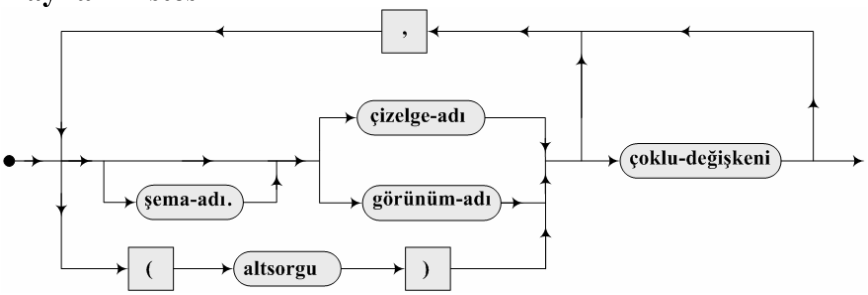
Deyim



deyim ::=



Kaynak Listesi



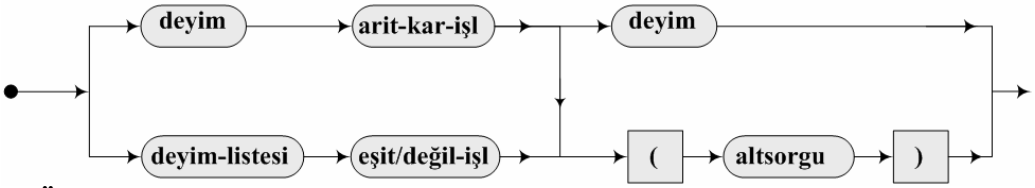
- Her kaynak çizelge bir temel çizelge ya da görünüm olabileceği gibi altsorgu ile tanımlanan bir çizelge de olabilir. Bu çizelgelerden her biri için, istenirse bir çoklu değişkeni tanımlanabilir. Çoklu değişkeni tanımlanmasının zorunlu olduğu durumlar şunlardır:

a.Bir altsorguyla tanımlanan kaynak çizelgeye gönderme yapılıyorsa.
b.Birden çok kaynak çizelgenin adı aynı ise.
c.Kaynak çizelgeye (temel çizelge ya da görünüm) farklı bir bloktan gönderme yapılıyorsa.

- **Örnekler:** 1. SELECT ..... FROM Personel ...
  2. SELECT ..... FROM Proje, Bölüm ...
  3. SELECT ..... FROM Personel X, Bölüm
  4. SELECT ..... FROM Malzeme M1, Malzeme M2 .....
  5. SELECT ..... FROM Proje, (SELECT ...) YP, Bölüm .....
- SELECT-FROM-WHERE bloklarından oluşan yalın bir SQL sorgusunun yanıtının şu şekilde elde edileceği söylenebilir:
  a.Kaynak ilişkilerin Karteziyen çarpımı alınır,
  b.Çarpım ilişkisinde WHERE koşulunu sağlamayan satırlar atılır,
  c.Koşulu sağlayan her satır için, amaç listedeki değerler seçilir ve amaç çizelge oluşturulur.

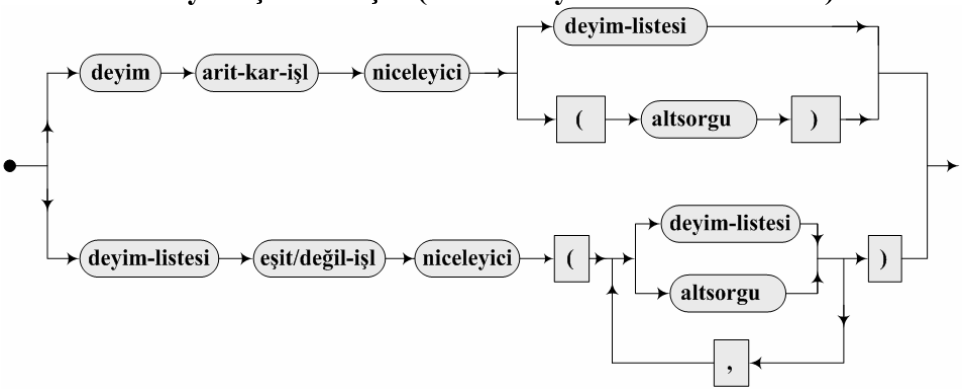
SQL koşulları

- SQL koşulları aşağıdaki gibi sınıflandırılabilir.
- K1. Aritmetik karşılaştırma koşulları (tekil önerme)



- **Örnekler:** 1. ücreti > 1000
  2. miktar < (mevMik - 10)
  3. sicNo = (SELECT yönSicNo FROM Bölüm WHERE bölNo = 3)
  4. mKodu != (SELECT mKodu FROM Tüketim WHERE prjNo = 7)
  5. (bölNo, sicNo) = (SELECT bölNo, yönSicNo FROM Proje WHERE prjNo = 5)

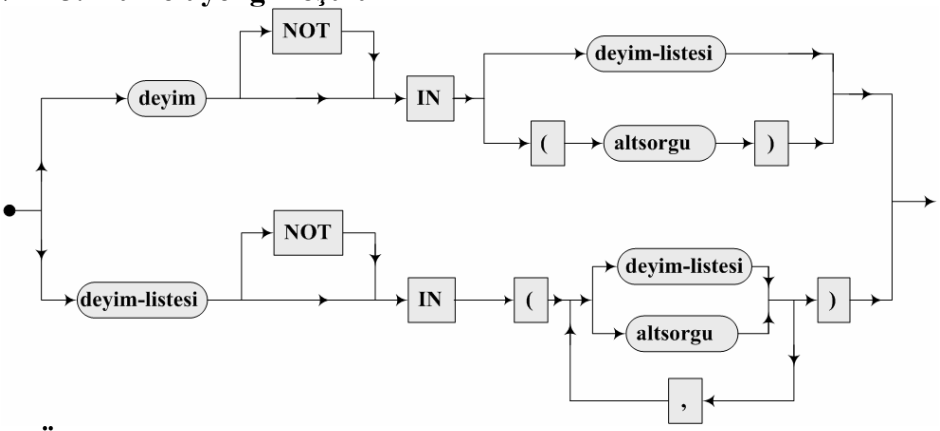
➢ **K2. Niceleyici içeren koşul (varlıksal ya da tümel önerme)**



niceleyici : ANY , SOME ya da ALL

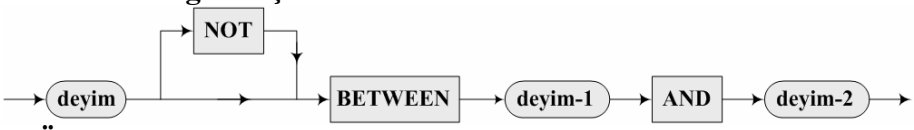
➢ **Örnekler:** 1. miktar > ALL ((mevmik * 2), (mevmik + 100))
2. mKodu = ANY (SELECT mKodu FROM Üretim WHERE Miktar > 100)
3. ücreti > ALL (SELECT ücreti FROM Personel WHERE bölNo = 3)
4. mKodu != ALL (SELECT mKodu FROM Tüketim WHERE prjNo = 7)
5. (mKodu, miktar) = SOME
(SELECT mKodu, miktar FROM Proje WHERE prjNo = 5)

➢ **K3. Küme üyeliği koşulu**



➢ **Örnekler:** 1. sicNo IN (SELECT sicNo FROM Personel
WHERE ücreti > 1000)
2. mKodu NOT IN (SELECT mKodu FROM Malzeme
WHERE mevMik = 0)
3. sicNo IN (SELECT yönSicNo FROM Bölüm WHERE bölNo = 3)
4. bölNo IN (12, 14, 18, 25)
5. (bölNo, ücreti) NOT IN ((5, 5000), (6, 4000), (7, 3500))
6. (bölNo, sicNo) IN (SELECT bölNo, yönSicNo FROM Proje
WHERE prjNo = 5)

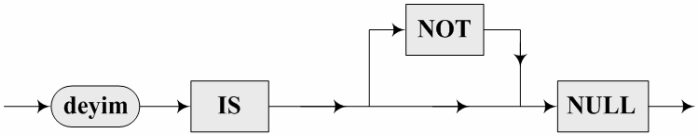
➢ **K4. Ara değer koşulu**



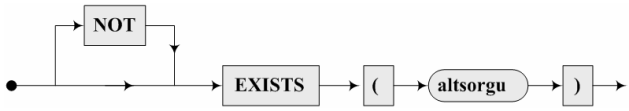
➢ **Örnekler:** 1. ücreti BETWEEN 1000 AND 1500
2. miktar BETWEEN mevMik*0.9 AND mevMik*1.1

➢ **K5. Eksik/belirsiz (NULL) değer koşulu**

➢ **Örnekler:** 1. yönSicNo IS NULL
2. bölNo IS NOT NULL

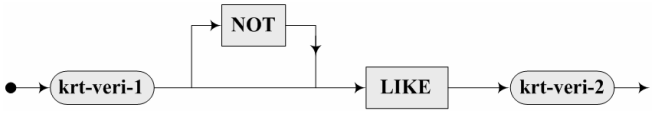


➢ **K6. Varolma koşulu**



➢ **Örnekler:** 1. EXISTS (SELECT * FROM Personel
WHERE ücreti > 5000)
2. NOT EXISTS (SELECT * FROM Tüketim
WHERE prjNo = 5 AND miktar > 1000)

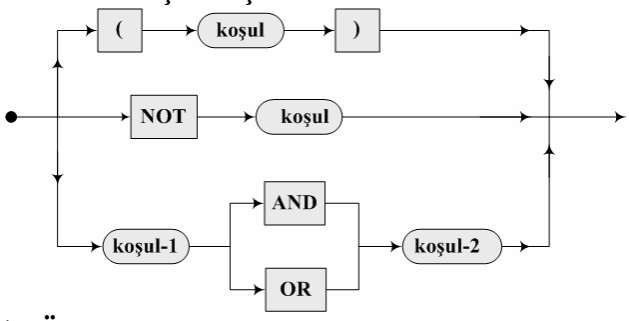
➢ **K7. Benzerlik koşulu**



krt-1 : karakter türü bir değer,
krt-2 : karakter türü bir örüntüdür.

➢ Örnekler: 1. adı LIKE 'MEH&'
2. adı LIKE '&TÜRK'

➢ **K8. Birleşik koşul**

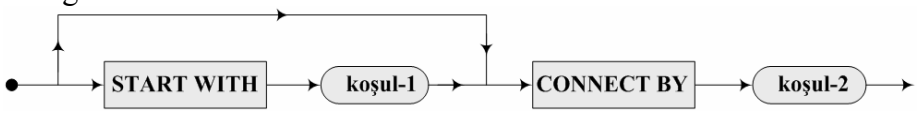


➢ **Örnekler:**

1. (ücreti > 1000) AND (bölNo != 8)
2. ücreti > ALL (SELECT ücreti FROM Personel WHERE
bölNo = 18) OR NOT EXISTS (SELECT * FROM Personel
WHERE ücreti. P.ücreti)
3. bölNo IS NULL OR bölNo IN (SELECT bölNo FROM bölüm)

Sıradüzensel Sorgu Tanımı

➢ Sorgunun yanıtını oluşturan çokluların kullanıcıya sıradüzensel sırada gösterilmesini sağlar.

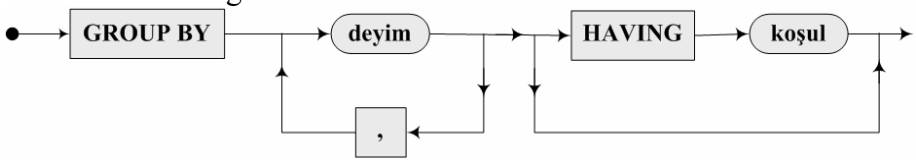


➢ START WITH koşul-1, sıradüzensel sorgunun kökünü oluşturacak satırı/satırları; CONNECT BY koşul-2 ise, sıradüzensel sorguda üst (*parent*) ve alt (*child*) satırlar arasındaki bağıntıyı tanımlar.

➢ Örnek: SELECT P.sicNo, P.adı FROM Personel P, Bölüm B
WHERE P.bölNo = B.bölNo
START WITH B.bölNo = 1 CONNECT BY PRIOR P.sicNo = B.yönSicNo;

Gruplandırma Tanımı

➢Gruplandırma tanımı ile, seçilen satırların tek tek işlenmeyeceği, satırların gruplandırılacağı, gruplardan bir kısmının eleneceği ve seçilen her grup için amaç çizelgede bir satır bulunacağı belirtilir.

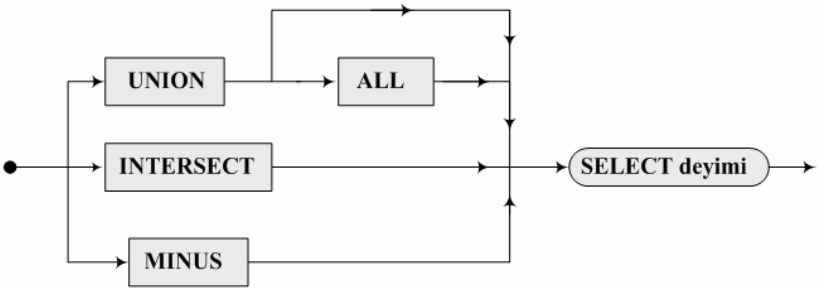


➢ Gruplandırma yapıldığında, gerek seçme listesinde, gerekse HAVING koşulunda, grupta değeri değişmeyen deyimler (gruplandırma için kullanılan deyim ya da deyimler ile bunlardan türetilenler) ile grup fonksiyonları kullanılabilr.

➢ Örnekler: 1. GROUP BY bölNo

2. GROUP BY mKodu HAVING SUM(ALL miktar) > 10000
3. GROUP BY bölNo, ücreti HAVING COUNT(*) > 5

Küme İşlemi Tanımı



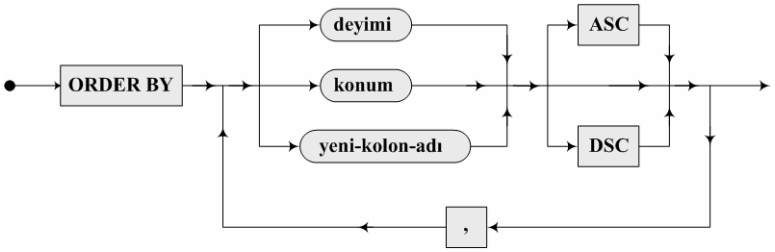
- SELECT deyimleri ile seçilen ara çizelgeler aralarına küme işlemi yazılabilmesi için bu çizelgelerin birleşim-uyar (*union-compatible*) olması, başka bir deyişle amaç kolon sayılarının eşit olması ve aynı sıradaki kolonların değer alanlarının aynı olması gerekir.

- Örnekler: 1. UNION (SELECT mKodu FROM Üretim)
2. UNION ALL (SELECT mKodu FROM Tüketim)
3. INTERSECT (SELECT bölNo FROM Proje)
4. MINUS (SELECT mKoduu FROM Alım WHERE fNo = 9)

Sıralama Tanımı

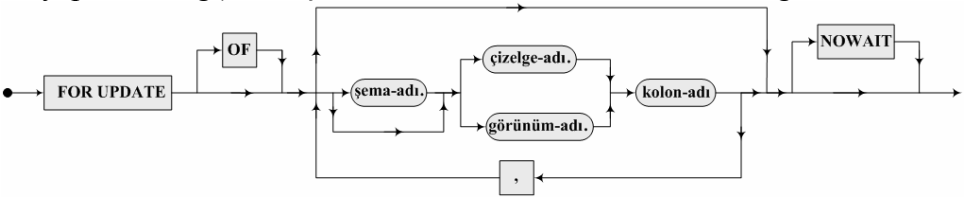
- **Seçilen satırların belirli değerlere göre sıralanarak gösterilmesini (döndürülmesini) sağlamak için kullanılır.**

- Örnekler:
1. ORDER BY sicNo
2. ORDER BY bölNo ASC, sicNo DESC



Kilitleme Tanımı

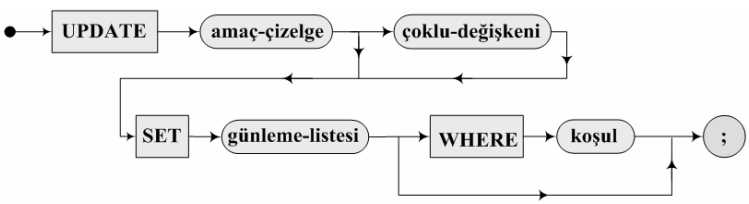
- Seçme işleminin günleme amacıyla yapıldığı (seçme işleminden sonra günleme yapılabileceği) ve seçilen verilerin kilitlenmesinin istendiği belirtilir.



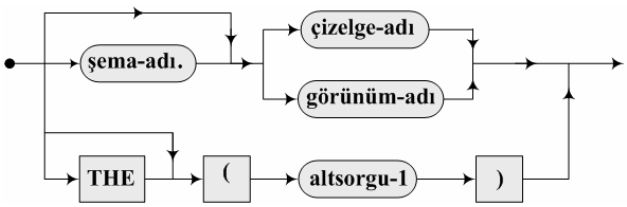
- Yalnız FOR UPDATE yazılırsa, seçilen tüm satırlar kilitlenir. FOR UPDATE OF ... yazılarak, yalnız belirli çizelgelerin belirli kolonlarının kilitlenmesi sağlanabilir.
- NOWAIT yazılırsa kilitlerin çözülmesi beklenmez, deyim hemen sonlandırılır. NOWAIT yazılmazsa, deyimin işletimi kilitlerin çözülmesine kadar bekletilir.
  - Örnekler: 1. FOR UPDATE
  2. FOR UPDATE OF Bölüm.bölNo NOWAIT
  3. FOR UPDATE OF Personel.sicNo, Personel.adı

7.2.2. UPDATE Deyimi

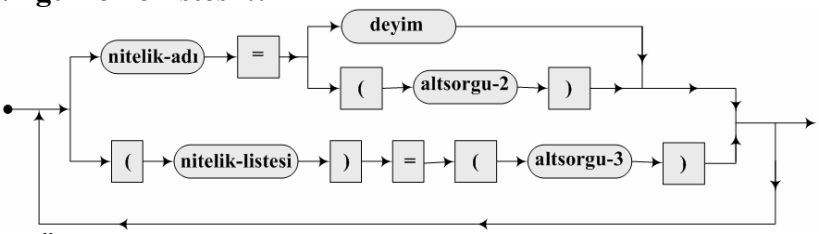
- Bir temel çizelge ya da bir görünümün tüm ya da belirli satırlarında, belirli niteliklerin değerleri günlemek için kullanılır.



- **amaç-çizelge ::=**



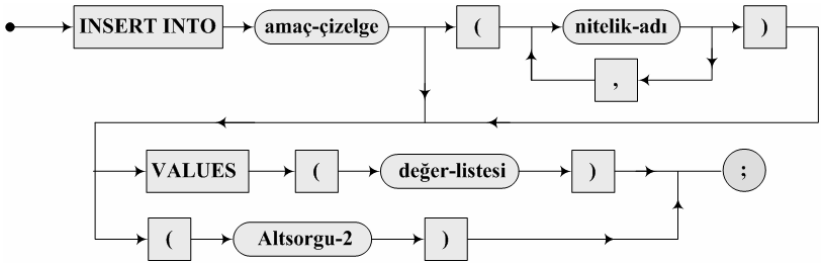
➢ **günleme-listesi ::=**



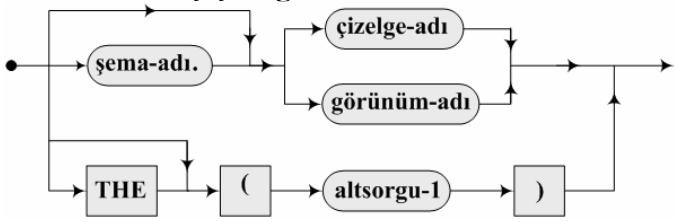
➢ **Örnekler:** 1. UPDATE Personel SET ücreti = ücreti*1.45;
2. UPDATE Üretim SET miktar = miktar + 5 WHERE bölNo = 7;
3. UPDATE Firma SET adresi = (SELECT adresi FROM Firma WHERE fNo = 7)
WHERE fAdı = 'Dökümsan';
4. UPDATE Personel SET (ücreti, bölNo) =
(SELECT ücreti, bölNo FROM Personel WHERE sicNo = 12345)
WHERE sicNo = 54321;

7.2.3. INSERT Deyimi

➢ Bir temel çizelge ya da görünüme bir ya da birkaç satır eklemek için kullanılır.



➢ **amaç-çizelge ::=**



➢ Amaç-çizelge'den sonra nitelik adları yazılmazsa, çizelgenin tüm niteliklerine değer aktarılması gerekir. Amaç-çizelge'den sonra belirli niteliklerin adları yazılırsa, yalnız yazılan niteliklere değer aktarılması gerekir. Adı yazılmayan niteliklere ise varsayılan değerleri aktarılır.

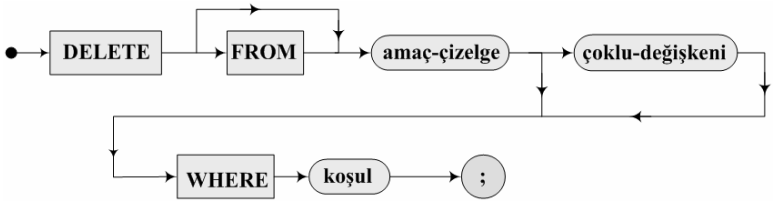
➢ Amaç-çizelge yerine THE (altsorgu-1) yazılırsa, altsorgu-1 bir görünüm tanımı gibi alınır ve ekleme işlemi görünüme ekleme gibi yapılır.

➢ Örnekler:

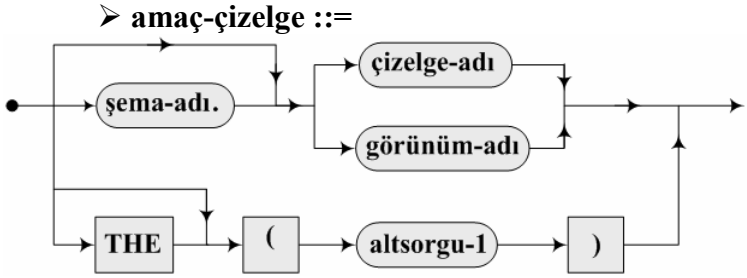
1. INSERT INTO Bölüm VALUES (756, 'Mekanik', '54381');
2. INSERT INTO Personel (sicNo, adı, bölNo)
VALUES (23415, 'Ali Özdalga', 756);
3. INSERT INTO Personel
(SELECT sicNo+1000, adı, ücreti*1.25, bölNo+100
FROM Personel WHERE bölNo = 321);

7.2.4. DELETE Deyimi

➢ Bir temel çizelge ya da görünümden bir ya da birkaç satır silmek için kullanılır.



➢ **amaç-çizelge ::=**



➢ Amaç-çizelge için THE (altsogu-1) gösterimi kullanılırsa, altsorgu-1 bir görünüm tanımı gibi alınır ve silme işlemi görünüm üzerinden silme gibi gerçekleştirilir.

➢ Deyimin sonuna "WHERE koşul" yazılırsa, amaç çizelgenin yalnız koşulu sağlayan

satırları silinir. Bu seçim yazılmazsa amaç çizelgenin tüm satırları silinir.

- Örnekler:

```
1. DELETE Alım;
2. DELETE FROM Tüketim WHERE prjNo = 3;
3. DELETE FROM THE (SELECT * FROM Firma
                    WHERE fNo NOT IN (SELECT fNo FROM Alım));
```

7.3. SQL Sorgu Örnekleri

7.3.1. Yalın Sorgular

- SÖ.7.1. "Tüm bölümlerin numaralarını, adlarını ve yöneticilerinin sicil numaralarını (başka bir deyişle BÖLÜM çizelgesinin tamamını) bul".

```
SELECT * FROM Bölüm;
```

- SÖ.7.2. "Kurumda üretilen malzemelerin kodlarını bul".

```
SELECT DISTINCT mKodu FROM Üretim;
```

- SÖ.7.3. "Ücreti 500 YTL'den büyük olan personelin tüm niteliklerini bul".

```
SELECT * FROM Personel WHERE ücreti > 500;
```

- SÖ.7.4. "Ücreti 500 YTL'den büyük olan personelin sicil numarasını ve adını bul".

```
SELECT sicNo, adı FROM Personel
WHERE ücreti > 500;
```

- SÖ.7.5. "3 numaralı bölümde çalışan ve ücreti 500 YTL'den büyük olan personelin sicil numarasını ve adını bul".

```
SELECT sicNo, adı FROM Personel
WHERE bölNo = 3 AND ücreti > 500;
```

- SÖ.7.6. "Ücreti 500 YTL'den büyük olan personelin sicil numarasını, adını ve çalıştığı bölümün adını bul".

```
SELECT Personel.sicNo, Personel.adı, Bölüm.adı
FROM Personel, Bölüm
WHERE ((Personel.bölNo = Bölüm.bölNo) AND
            (Personel.ücreti > 500));
```

- SÖ.7.7. "SA3842 kodlu malzemeyi üreten bölümlerin numaralarını bul".

```
SELECT DISTINCT bölNo FROM Üretim
WHERE mKodu = 'SA3842';
```

7.3.2. Çok Düzeyli Sorgular

- SQL'de sorgular iç içe SELECT deyimleri ile anlatılabilir; SELECT deyimleri arasında küme üyeliği, niceleme ve varolma koşulları kullanılabilir. Bu olanaklar dilin anlatım gücünü arttırdığı gibi aynı sorgunun farklı biçimlerde anlatılmasını sağlayarak dile anlatım zenginliği de kazandırır.

- SÖ.7.8. "SA3842 kodlu malzemeyi üreten bölümlerin numaralarını ve adlarını bul".

Bu sorgu SQL'de üç farklı yapıda anlatılabilir.

a)
```
SELECT DISTINCT B.bölNo, B.bölAdı FROM Bölüm B, Üretim Ü
WHERE B.bölNo = Ü.bölNo AND Ü.mKodu = 'SA3842';
```

b)
```
SELECT bölNo, bölAdı FROM Bölüm WHERE bölNo IN
    (SELECT bölNo FROM Üretim WHERE mKodu = 'SA3842');
```

c)
```
SELECT B.bölNo, B.bölAdı FROM Bölüm B WHERE EXISTS
    (SELECT * FROM Üretim Ü
     WHERE Ü.mKodu = 'SA3842' AND Ü.bölNo = B.bölNo);
```

- SÖ.7.9. "Kaya Mert adlı kişinin çalıştığı bölümün numarasını ve adını bul".

(Diğer birçok sorgu gibi bu sorgu da SQL'de farklı biçimlerde anlatılabilir. Aşağıda bu sorgunun anlatım biçimlerinden biri verilmektedir.)

```
SELECT bölNo, bölAdı FROM Bölüm WHERE BölNo IN
    (SELECT BölNo FROM Personel WHERE adı = 'Kaya Mert');
```

- SÖ.7.10. "Kaya Mert adlı kişinin çalıştığı bölümün yöneticisinin sicil numarasını ve adını bul".

```
SELECT sicNo, adı FROM Personel WHERE sicNo IN
        (SELECT yönSicNo FROM Bölüm WHERE bölNo IN
                (SELECT bölNo FROM Personel WHERE adı = 'Kaya Mert'));
```

- SÖ.7.11. "38 numaralı firmadan satın alınan malzemeri tüketen projelerin numaralarını ve adlarını bul".

```
SELECT prjNo, prjAdı FROM Proje WHERE prjNo = ANY
        (SELECT prjNo FROM Tüketim WHERE mKodu IN
                (SELECT mKodu FROM Alım WHERE fNo = 38));
```

- SÖ.7.12. "3 numaralı projede tüketilen malzemelerden en az birini tüketen projelerin numaralarını ve adlarını bul".

```
SELECT PrjNo, prjAdı FROM Proje WHERE prjNo IN
        (SELECT prjNo FROM Tüketim WHERE mKodu IN
                (SELECT mKodu FROM Tüketim WHERE prjNo = 3));
```

- SÖ.7.13. "SA3842 kodlu malzemeyi tüketmeyen projelerin numaralarını ve adlarını bul".

```
SELECT prjNo, prjAdı FROM Proje  WHERE prjNo NOT IN
        (SELECT prjNo FROM Tüketim  WHERE mKodu = 'SA3842');
```

Bu sorgu, NOT EXISTS koşulu kullanılarak aşağıdaki gibi de yazılabilir.

```
SELECT P.prjNo, P.prjAdı FROM Proje P WHERE NOT EXISTS
        (SELECT * FROM Tüketim T
         WHERE T.prjNo = P.prjNo AND T.mKodu = 'SA3842');
```

- SÖ.7.14. "Mevcut miktarı 0 olan ve kurumda üretimeyen malzemelerin kod ve adlarını bul".

```
SELECT mKodu, mAdı FROM Malzeme
WHERE mevMik = 0 AND mKodu NOT IN  (SELECT mKodu FROM Üretim);
```

- SÖ.7.15. "Kurumda en yüksek ücreti alan personelin sicil numarasını ve adını bul".

```
SELECT sicNo, adı FROM Personel WHERE ücreti >= ALL
                                        (SELECT ücreti FROM Personel);
```

- SÖ.7.16. "Bölüm yöneticisi olmayan, ancak bölüm yöneticilerinin en az birinden daha yüksek ücret alan personelin sicil numarasını, adını ve ücretini bul".

```
SELECT P.sicNo, P.adı, P.ücreti FROM Personel P
WHERE P.sicNo NOT IN (SELECT yönsicNO FROM Personel)
        AND P.ücreti > SOME  (SELECT DISTINCT Y.ücreti
                              FROM Personel Y, Bölüm B
                              WHERE Y.sicNo = B.yönSicNo);
```

- SÖ.7.17. "Çalışanlardan en az birinin ücreti bölüm yöneticisinin ücretinden daha büyük olan bölümlerin numarasını ve adını bul".

```
SELECT B.bölNo, B.bölAdı FROM Bölüm B
WHERE (SELECT ücreti FROM Personel Y
        WHERE Y.sicNo = B.yönSicNo) < SOME
                (SELECT ücreti FROM Personel P
                WHERE P.bölNo = B.bölNo);
```

- SÖ.7.18. "Satın alınan tüm malzemeleri tüketen (tükettiği malzemeler arasında satın alınan tüm malzemeler yer alan) projelerin numaralarını ve adlarını bul".

```
SELECT P.prjNo, P.prjAdı FROM Proje P WHERE NOT EXISTS
        ((SELECT DISTINCT mKodu FROM Alım)  MINUS
         (SELECT mKodu FROM Tüketim T WHERE T.prjNo = P.prjNo));
```

- SÖ.7.19. "BZ018, BZ172 ve BZ424 kodlu malzemelerin her üçünü de üreten bölümlerin

numaralarını ve adlarını bul".

```
SELECT bölNo, bölAdı FROM Bölüm WHERE bölNo IN
      (SELECT DISTINCT Ü1.BölNo FROM Üretim Ü1, Üretim Ü2, Üretim Ü3
       WHERE (Ü1.bölNo = Ü2.bölNo) AND  (Ü2.bölNo = Ü3.BölNo) AND
             (Ü1. mKodu = 'BZ018') AND  (Ü2.mKodu = 'BZ172') AND
             (Ü3.mKodu = 'BZ424'));
```

- SÖ.7.20. "Tükettiği malzemelerin tümü kurumda üretilen malzemeler olan projelerin numaralarını ve adlarını bul".

```
SELECT  prjNo, prjAdı FROM Proje WHERE prjNo NOT IN
      (SELECT DISTINCT prjNo FROM Tüketim  WHERE mKodu NOT IN
            (SELECT DISTINCT mKodu FROM Üretim));
```

- SÖ.7.21. "7 numaralı projede tüketilen tüm malzemeleri üreten bölümlerin numaralarını ve adlarını bul".

```
SELECT B.bölNo, B.bölAdı FROM Bölüm B WHERE NOT EXISTS
      ((SELECT T.mKodu FROM Tüketim T WHERE T.prjNo = 7) MINUS
       (SELECT Ü.mKodu FROM Üretim Ü WHERE Ü.bölNo = B.bölNo));
```

- SÖ.7.22. "Tükettiği malzemelerin hiçbiri yapıldığı bölümde üretilmeyen projelerin numaralarını ve adlarını bul".

```
SELECT P.prjNo, P.prjAdı FROM Proje P WHERE NOT EXISTS
      (SELECT * FROM Tüketim T, Üretim Ü
       WHERE ((T.prjNo = P.prjNo) AND (T.mKodu = Ü.mKodu) AND
             (Ü.bölNo = P.bölNo)));
```

7.3.3. Seçme Listesinde Deyim ve İşlev Kullanılması

- Amaç çizelgenin nitelikleri, kaynak çizelgelerdeki niteliklerden elde edilen yeni değerler de olabilir. Diğer taraftan sorgunun yanıtı amaç çizegeden oluşabileceği gibi, amaç çizelgeye uygulanacak işlev ya da işlevlerin değerlerinden de oluşabilir. Seçme listesinde deyim ve işlev kullanımı ile ilgili örnek sorgular aşağıda yer almaktadır.
- SÖ.7.23. "Her personelin sicil numarasını, adını ve %25 artırılmış ücretini bul".

```
SELECT SicNo, adı, ücreti*1.25 AS yeniÜcreti FROM Personel;
```

- SÖ.7.24. "15 ve 36 numaralı bölümlerin her ikisinde de üretilen malzemelerin kodlarını ve bu iki bölümündeki üretim miktarlarının toplamını bul".

```
SELECT DISTINCT P1.mKodu, (P1.miktar + P2.miktar) AS tMiktar
FROM Üretim P1, Üretim P2
WHERE ((P1.bölNo = 15) AND (P2.bölNo = 36) AND
        (P1.mKodu = P2.mKodu));
```

- SÖ.7.25. "Kurumda üretilen toplam 'konnektör' miktarını bul".

```
SELECT SUM(ALL miktar) AS konMik
FROM Üretim WHERE mKodu =
      (SELECT mKodu FROM Malzeme WHERE mAdı = 'Konnektör');
```

- SÖ.7.26. “17 numaralı bölümde çalışan personelin sayısını ve ücretlerinin toplamını bul”.

```
SELECT COUNT(*) AS perSay, SUM(ALL ücreti) AS ücretTop
FROM Personel WHERE bölNo = 17;
```

- SÖ.7.27. "Kurumda en yüksek ücreti alan personelin sicil numarasını ve adını bul".

```
SELECT siNo, adı FROM Personel WHERE ücreti =
                        (SELECT MAX(ücreti) FROM Personel;
```

- SÖ.7.28. "Kurumda en az ücret alan kişi ya da kişilerin sicil numarasını ve adını bul".

a)
```
SELECT sicNo, adı FROM Personel WHERE ücreti =
      (SELECT MIN(ücreti) FROM Personel);
```

b)
```
SELECT K.sicNo, K.adı FROM Personel K WHERE NOT EXISTS
      (SELECT * FROM Personel P WHERE P.ücreti < K.ücreti);
```

➢ SÖ.7.29. "Kurumda üretilen malzeme sayısını bul".

```
SELECT COUNT(DISTINCT mKodu) AS mSayısı FROM Üretim;
```

➢ SÖ.7.30. "Ücreti, 17 numaralı bölümde çalışan personelin tümünün ücretinden büyük olan personelin sicil numarasını ve adını bul".

a)
```
SELECT sicNo, adı FROM Personel WHERE ücreti > ALL
    (SELECT ücreti FROM Personel WHERE bölNo = 17);
```

b)
```
SELECT sicNo, adı FROM Personel WHERE ücreti >
    (SELECT MAX(ücreti) FROM Personel WHERE bölNo = 7);
```

7.3.3. Gruplandırma Gerektiren Sorgular

➢ SÖ.7.31. "Bölüm numaralarını ve çalışan personel sayılarını bul".

```
SELECT bölNo AS bölümNo, COUNT(*) AS perSay
FROM Personel GROUP BY bölNo;
```

➢ SÖ.7.32. "Çalışan personel sayısı en az 3 olan bölümlerin numaralarını bul".

```
SELECT bölNo FROM Personel
GROUP BY bölNo HAVING COUNT(*) > 2;
```

➢ SÖ.7.33. "En az 3 bölüm tarafından üretilen malzemelerin kodlarını, adlarını ve bu malzemeleri üreten bölüm sayılarını bul".

```
SELECT M.mKodu, M.mAdı, COUNT(Ü.bölNo) AS bölSay
FROM Malzeme M, Üretim Ü WHERE M.mKodu = Ü.mKodu
GROUP BY M.mKodu HAVING COUNT(*) > 3;
```

➢ SÖ.7.34. "Bölümlerin numaralarını, adlarını, personel sayılarını ve çalışan personelin ücret toplamlarını bul".

```
SELECT B.bölNo AS bölümNo, B.bölAdı AS bölümAdı,
        COUNT(*) AS perSay, SUM (ALL ücreti) AS ücrTop
FROM Bölüm B, Personel P  WHERE B.bölNo = P.bölNo GROUP BY B.bölNo;
```

➢ SÖ.7.35. "Çalışan personelin tümünün ücreti 500 YTL'den ya da daha büyük olan bölümlerin numaralarını ve adlarını bul".

```
SELECT BNO, ADI FROM BÖLÜM WHERE BNO IN
    (SELECT BNO FROM PERSONEL
    GROUP BY BNO HAVING MIN(ÜCRETİ) >= 500);
```

➢ SÖ.7.36. "Çalışan personelden en az birinin ücreti 500 YTL'den küçük olan bölümlerin numaralarını ve adlarını bul".

```
SELECT bölNo, bölAdı FROM Bölüm WHERE bölNo IN
    (SELECT bölNo FROM Personel
     GROUP BY bölNo HAVING MIN(ücreti) < 500);
```

➢ SÖ.7.37. "Personel sayısı 10'dan küçük olup, en az 5 malzeme üreten bölümlerin numaralarını ve adlarını bul".

```
SELECT bölNo, bölAdı FROM Bölüm WHERE bölNo IN
    ((SELECT bölNo FROM Personel
      GROUP BY bölNo HAVING COUNT(*) < 10) INTERSECT
    (SELECT bölNo FROM Üretim
     GROUP BY BNO HAVING COUNT(*) >= 5));
```

➢ SÖ.7.38. "Çalışan personelin tümü birbirinden farklı ücret alan bölümlerin numaralarını ve adlarını bul".

a)
```
SELECT bölNo, bölAdı FROM Bölüm WHERE bölNo IN
    (SELECT bölNo FROM Personel GROUP BY bölNo
     HAVING COUNT(ALL ücreti) = COUNT(DISTINCT ücreti);
```

b)
```
SELECT B.bölNo, B.bölAdı FROM Bölüm B WHERE NOT EXISTS
    (SELECT * FROM Personel P1, Personel P2
     WHERE (P1.bölNo = B.bölNo) AND (P2.bölNo= P1.bölNo) AND
                                        (P1.ücreti = P2.ücreti));
```