VERİ TABANLARINDA VERİ TİPLERİ

Veri tabanları, verilerin düzenli ve erişilebilir bir biçimde saklanmasını sağlayan sistemlerdir. Bu sistemlerde kullanılan veri tipleri, bilgilerin ne şekilde işleneceğini, saklanacağını ve karşılaştırılacağını belirler. Veri tipi seçimi, hem veri bütünlüğü hem de sistem performansı açısından büyük önem taşır. Bu metinde, NoSQL ve SQL veri tabanlarında kullanılan veri tipleri detaylı bir şekilde açıklanmıştır.

1. NoSQL Veri Tabanlarında Veri Tipleri

NoSQL (Not Only SQL) veri tabanları, ilişkisel olmayan yapıdaki verileri depolamak için kullanılan sistemlerdir. Geleneksel tablolardan ziyade belge (document), anahtar-değer (key-value), grafik ve sütun tabanlı yapılara dayanır. Aşağıda NoSQL sistemlerinde yaygın olarak karşılaşılan veri tipleri açıklanmıştır:

- **String (Dize):** Metin değerlerini ifade eder. Kullanıcı adı, açıklama gibi verileri tutmak için kullanılır.
- **Number (Sayı):** Tamsayılar ve ondalıklı sayılar gibi sayısal verileri saklar.
- **Boolean:** Yalnızca iki değere sahip olan bu tür, mantıksal işlemlerde kullanılır (true/false).
- **Array (Dizi):** Birden fazla değeri tek bir alanda tutar. Örneğin bir kullanıcının ilgi alanları.
- **Object (Belge / Document):** JSON veya BSON formatında iç içe geçmiş anahtar-değer çiftleri içerir.
- **Null:** Değerin tanımsız ya da boş olduğunu belirtir.
- **Binary (İkili):** Görsel, ses veya diğer dosya türlerini saklamak için kullanılır.
- **Timestamp:** Belirli bir zamanı ifade eder; veri girişi ya da güncelleme zamanı gibi bilgiler için kullanılır.

2. SQL Veri Tabanlarında Veri Tipleri

SQL (Structured Query Language) tabanlı veri tabanlarında veri türleri daha yapılandırılmış ve katı kurallara sahiptir. SQL sistemlerinde veri türleri üç ana kategoride incelenir: **sayısal**, **karakter**, ve **zaman (tarih/saat)** veri tipleri. Ayrıca bazı özel ve çeşitli veri türleri de bulunmaktadır.

3. Sayısal (Numeric) Veri Tipleri

Bu veri tipleri, matematiksel işlemler ve sayısal analizler için kullanılır:

- **INT / INTEGER:** Tamsayıları saklamak için kullanılır. Genellikle 4 byte yer kaplar.
- **SMALLINT:** Daha küçük aralıktaki tamsayıları saklar. 2 byte yer kaplar.
- **BIGINT:** Büyük boyutlu tamsayılar için uygundur. 8 byte yer kaplar.
- **DECIMAL(p, s) / NUMERIC(p, s):** Ondalık sayıları yüksek kesinlikte tutar. "p" toplam basamak sayısını, "s" ondalık kısmı belirtir.
- **FLOAT / REAL / DOUBLE PRECISION:** Yaklaşık ondalıklı sayı tutar. Bilimsel hesaplamalarda tercih edilir.

4. Karakter (Character) Veri Tipleri

Metin biçimindeki veriler bu türle saklanır:

- **CHAR(n):** Sabit uzunluklu metinler için kullanılır. Eksik karakterler boşlukla tamamlanır.
- **VARCHAR(n):** Değişken uzunluklu metin verileri için kullanılır.
- **TEXT:** Uzun metin içeriklerini (örneğin blog yazıları) tutmak için kullanılır.

5. Unicode Karakter Veri Tipleri

Çok dilli uygulamalarda kullanılan bu veri tipleri, tüm dillerin karakterlerini destekler:

- **NCHAR(n):** Unicode destekli sabit uzunluklu karakter dizisidir.
- **NVARCHAR(n):** Unicode destekli değişken uzunluklu karakter dizisidir.
- **NTEXT:** Uzun Unicode metinler için kullanılır; ancak günümüzde yerini `NVARCHAR(MAX)` almıştır.

6. Date ve Time (Tarih ve Zaman) Veri Tipleri

Tarih ve saat bilgilerini saklamak için kullanılır:

- **DATE:** Yıl, ay ve gün bilgisi tutar. Örn: `2025-07-31`
- **TIME:** Saat, dakika, saniye verilerini içerir. Örn: `15:45:00`
- **DATETIME:** Hem tarih hem saat bilgisini birlikte saklar. Örn: `2025-07-31 15:45:00`
- **TIMESTAMP:** Verinin oluşturulma ya da güncellenme zamanını belirtir.
- **YEAR:** Yalnızca yıl bilgisini içerir (örn: `2025`).

7. Çeşitli Veri Tipleri

SQL sistemlerinde ve bazı NoSQL veri tabanlarında özel durumlar için kullanılan veri türleri şunlardır:

- **BOOLEAN:** Mantıksal (true/false) değerler için kullanılır.
- **ENUM:** Sadece belirli seçeneklerin seçilebildiği sınırlı bir değer kümesidir. Örn: (‘Erkek’, ‘Kadın’, ‘Diğer’).
- **SET:** Çoklu değerlerin seçilebildiği sabit bir değer kümesidir.
- **BLOB (Binary Large Object):** İkili verileri (resim, video, PDF) saklamak için kullanılır.
- **JSON / JSONB:** Yapısal verilerin saklanması için uygundur. Özellikle PostgreSQL gibi veri tabanlarında yaygındır.
- **UUID:** Evrensel benzersiz tanımlayıcıdır; her kayıt için tekil bir kimlik sağlar.
- **GEOMETRY / GEOGRAPHY:** Coğrafi konum bilgileri için kullanılır (GIS sistemlerinde).

SONUÇ

Veri tabanlarında kullanılan veri tipleri, verinin türüne ve kullanılacağı alana göre dikkatle seçilmelidir. Hem SQL hem de NoSQL veri tabanları, farklı ihtiyaçlara göre veri türleri sunar. Veri türlerinin doğru seçilmesi, sistemin verimliliğini ve güvenilirliğini doğrudan etkiler. Veri tabanı tasarımında veri tiplerine gereken özen gösterilmelidir.

VERİ TABANI NESNELERİ ve BİLEŞENLERİ

Veri tabanı yönetim sistemleri, yalnızca veri depolamakla kalmaz; aynı zamanda kullanıcıları, izinleri, yapıları ve erişim kontrollerini yöneten çok sayıda bileşenden oluşur. Bu bileşenler, verinin doğru, güvenli ve hızlı işlenmesini sağlamak üzere tasarlanmıştır. Aşağıda bu kavramların detaylı açıklamaları ve örnekleri verilmiştir.

1. Kullanıcı (User)

Veri tabanına erişim sağlayan birey ya da uygulamadır. Her kullanıcının bir kullanıcı adı ve genellikle bir şifresi vardır.

Örnek:

- Kullanıcı adı: `arda_karadag`
- Yetkileri: yalnızca veri görüntüleme

2. Şema (Schema)

Veri tabanında nesnelerin (tablolar, görünümler, prosedürler vb.) mantıksal gruplandırmasını sağlar. Aynı veri tabanı içinde birden fazla şema bulunabilir.

Örnek:

- `ogrenci_schema` içinde: `ogrenciler`, `notlar`, `dersler` tabloları yer alabilir.

3. Profil (Profile), Yetki (Privilege), Rol (Role)

- **Profil (Profile):** Kullanıcıların oturum parametrelerini belirler (örn. bağlantı süresi, kaynak kullanımı).
- **Yetki (Privilege):** Bir kullanıcıya özel izinler tanımlar (örn. SELECT, INSERT).
- **Rol (Role):** Birden fazla yetkinin gruplandırılmasıdır; birden fazla kullanıcıya aynı anda atanabilir.

Örnek:

- `readonly_role`: Sadece SELECT yetkisi içerir.
- `admin_profile`: 24 saat bağlantı izni verir.

4. Collection, Tablo (Table), Kolon (Column), Satır (Row), Hücre (Cell/Field)

- **Collection:** NoSQL veri tabanlarında, belgelerin tutulduğu yapıdır. SQL'deki tabloya benzer.
- **Tablo (Table):** Verilerin satır ve sütun şeklinde tutulduğu yapıdır.
- **Kolon (Column):** Her sütun, bir veri tipine göre tanımlanır.
- **Satır (Row):** Tabloya eklenen her kayıt bir satır oluşturur.
- **Hücre (Cell/Field):** Bir satır ile bir sütunun kesişimindeki veri noktasıdır.

Örnek:

Tablo: `ogrenciler`

Kolonlar: `id`, `ad`, `soyad`, `dogum_tarihi`

Satır: `1, Arda, Karadağ, 2003-02-15`

Hücre: `Arda` (ad kolonunda, 1. satır)

5. Sanal Tablo (View)

Gerçek bir veri içermez, ancak bir veya daha fazla tablodan sorgu sonucu oluşan sanal yapıdır. Kullanıcıya veriyi filtrelenmiş biçimde sunar.

Örnek:

```
CREATE VIEW aktif_ogrenciler AS
SELECT ad, soyad FROM ogrenciler WHERE aktif = 1;
```

6. Takma İsim (Synonym)

Veri tabanı nesnesine başka bir adla erişimi sağlar. Özellikle farklı şemalardaki nesnelere kolay erişim için kullanılır.

Örnek:

`ders_notu` adlı tabloya `dn` takma adı verilebilir:

```
CREATE SYNONYM dn FOR ders_notu;
```

7. Sayı Üreteci (Sequence)

Artan ya da azalan sayılar üretir. Genellikle birincil anahtar (ID) üretmek için kullanılır.

Örnek:

```
CREATE SEQUENCE ogrenci_id_seq START WITH 1 INCREMENT BY 1;
```

8. Dizin (Index), Anahtar (Key)

- **Dizin (Index):** Veri arama işlemlerini hızlandıran yapılardır.
- **Anahtar (Key):** Verinin benzersizliğini veya ilişkilendirilmesini sağlayan sütun(lar).

Örnek:

- `id` sütunu PRIMARY KEY olabilir.
- `INDEX` ile `ad` sütununa dizin eklenerek sorgular hızlandırılabilir.

9. Tablespace, FileGroup

- **Tablespace:** Fiziksel veri dosyalarının saklandığı alandır (özellikle Oracle'da kullanılır).
- **FileGroup:** SQL Server'da benzer işlevi görür; veri dosyalarını gruplar.

Örnek:

- `students_data` adlı tablo, `userspace1` adlı tablespace'e atanabilir.

10. Function, Procedure, Package, Trigger

- **Function:** Girdi alıp çıktı döndüren yapılardır.
- **Procedure:** Belirli işlemleri gerçekleştiren komut bloklarıdır; genellikle çıktı döndürmez.
- **Package:** Birden fazla fonksiyon ve prosedürün bir araya geldiği yapıdır.
- **Trigger:** Tablodaki bir işlem gerçekleştiğinde otomatik çalışan yapıdır.

Örnek:

```
CREATE FUNCTION tam_ad(ad TEXT, soyad TEXT)
RETURNS TEXT AS $$ SELECT ad || ' ' || soyad; $$ LANGUAGE SQL;

CREATE TRIGGER not_girildiginde_logla
AFTER INSERT ON notlar
FOR EACH ROW EXECUTE FUNCTION logla();
```

SONUÇ

Veri tabanı sistemleri, yalnızca veri değil; kullanıcılar, roller, nesneler ve işlevsellik gibi birçok kavramdan oluşur. Bu kavramların bilinmesi, hem sistem yönetimi hem de güvenli veri işlemleri açısından oldukça önemlidir. Yukarıda açıklanan nesne türleri, hem SQL hem de NoSQL sistemlerde karşımıza çıkabilir ve profesyonel veri yönetimi için temel bir yapı sunar.

Bu çalışma, BTK Akademi kaynakları doğrultusunda hazırlanmış olup Arda Karadağ tarafından yazılmıştır.