Constraintler ilişkisel veri tabanıyla ilişkisel olmayan veri tabanı arasındaki en büyük farkı yaratan konu.

**VİEW**

Fiziksel olarak diskte yer tutmaz.

Bir tabloymuş gibi, obje olarak var ve onun üzerinden sql run edilebiliyor.

Tablo gibi yer alır, diskte yer işgal etmez.

SELECT dışında DML operasyonları yapılamaz.

**SİMPLE VİEW vs COMPLEX VİEW**

Tek tablo kullanır. Birden çok tablo kullanılır

Fınction kullanılmaz. Functionslar Kullanılır.

Gruplama yok. Gruplama var.

DML çalıştırılır. DML çalıştırılamaz.

**İNLİNE VİEWS**

SELECT e.last name, e.salary, c.department\_id, d.maxsal

FROM employees e,

(SELECT department\_id, max (salary) maxsal

FROM employees

GROUP BY department\_id) d

WHERE department\_id = d.depar tment\_id

AND e.salary = d. maxsal;

Herhangi tablo olmadan bir tablo oluşturuyoruz. Her departmanın maxımum ucretıne esıt alan ucretı alacak.

**ROWNUM**

Rowların sırasını belirtiriz. ORDER BY dan sonra ROWNUM gelse daha mantıklı olacaktı. Yanlış bu.

En kıdemlı ilk 5 çalışanı getir;

SELECT ROWNUM AS "Longest employed", last \_name, hire\_date

FROM employees

WERE ROWNUM <=5

ORDER BY hire \_date;

Doğrusu;

SELECT ROWNUM AS "Longest employed", 1ast \_name, hire\_date

FROM (SELECT last\_name, hire \_date

FROM employees

ORDER BY hire date)

WHERE ROWNUM <=5:

**SEQUENCES**

Tablolarımızda PK kullanmayacaksak,

Otomatik sıralama yapmak istediğimiz zaman yararlanırız.

**\*\*\*INDEXES\*\*\***

1. Belirli bir parçaya ulaşmamızı sağlar.
2. Biricikliği UNIQLIĞI sağlar.

INDEX ne zaman yapılmalı?

Herhangi sprguda kullanacağımız parça %2 ve %4 arasındaysa

Sütunları birlikte kullanıyorsam

Sütunların çoğu NULLsa

**USER ACCESS**

Veri tabanı yönetim kısmı ile alakalıdır.

En önemli komutu GRANT ile erişim hakkı veririz. REVOKE…

DCL(GRANT AND REVOKE)

**DATABASE TRANSACTİONS**

COMMIT: Yaptığımız işlem veri tabanına kaydedilir ve kalıcı hale gelir.

ROLLBACK: Transaction geri alınır. Kalıcıdır. Hesaptan para düşüp diğer hesaba yazmak bir transaction örneği.

SAVEPOINT: Biyerde saklarız. Geri alabilmek için bu save noktasına geri alabiliriz. Bir INSERT ve UPDATE sonrası geri almak için faydalıdır. ROLLBACK yaparken SAVEPOINT belirtiriz.

**LOCKSDATA**

\*\*Developar a LOCK mekanizmasını devreye sokan, isolationı sağlayan SQL cümleciği? - FOR UPDATE

FOR UPDATE kullandı. Onun işi bitecek sonra diğer kullanıcılar gelecek. LOCK etti ve isoloation sağladı.

**DATA WAREHOUSİNG**

İşi bitmiş verileri, işe yarayacak şekilde yapılandırılmış olarak yeni katman oluşturmadır.

Ortada ürün yok, yol yordam bir yapıdır. Bir yöneticinin karar vermesıne destek olacak şekilde verinin toplanma şeklidir.

Bir tarafta normalizasyonu esas alan yapı ve diğer tarafta dimensional yapı (star ve snow şema)vardır.

**\*\*\*TERİMLER\*\*\***

Konu Odaklı (Subject Oriented) : Veri ambarını konulara göre oluşturmadır.

Entegre (Integrated) : Data warehouse, veri ambarı içinde entegre olması gerek.

Kalıcı (Nonvolatile) : Değişmeyecek veriler.

Zaman Boyutlu (Time Variant) : Bir veriyi tarihi ile tutmak

**Ad hoc Querie:** anında sonuç

**Canned Querie**: Önceden bilinen sorular

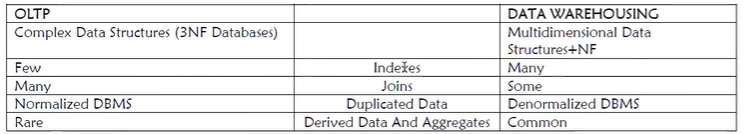
**OLTP :** Transaction’ın kaydedildiği katman. DATA W temelinde OLTP vardır. Normalizsyn yapısı var.

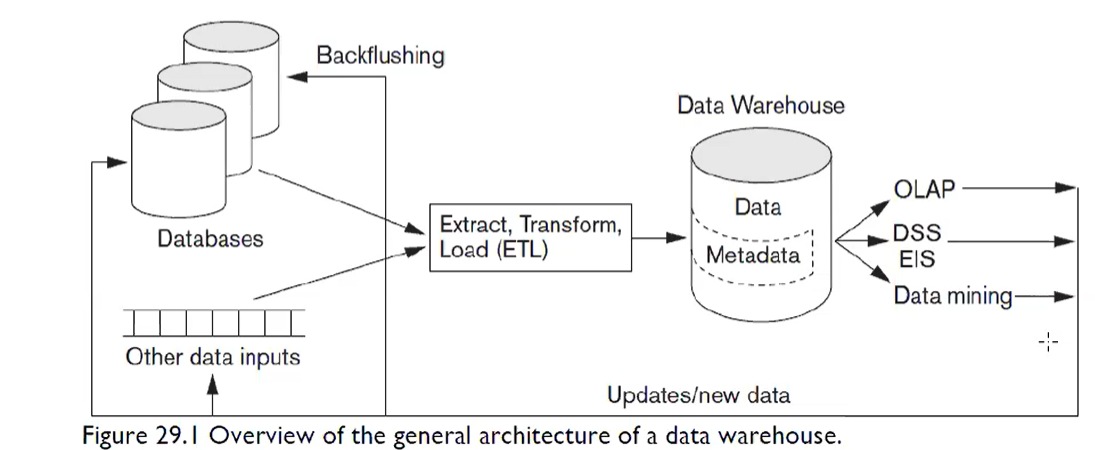
Karar verici Developar a bağlı. Bu sebeble farklı katman ihtiyacı doğar.

**Karar Destek Sistemleri:** Karar verme aşamasında Verilerl farklı katmanlarda oluşturur.

**OLAP:**

**ETL(extract, transform, load):** Katman oluşturmak için kullanır.

****



Uzun zaman boyutuna yayılmış veriyi tutmayı sağlar.

Immon: Normalizasyonu dikkate alır.

Kimball:

**DATA WAREHOUSES MİMARİSİ**

Basic

Staging Area: Ara katman

**OLTP SYSTEMS DATA WAREHOUSİNG SYSTEMS**

Cari data tutar Tarihsel datayı tutar

Detaylı yapılı veri Daha özetlenmiş veri

Dinamik data Statik data

Tekrarlı işlemler çok Yapılandırılmamıs süreç

Application orianted Subject orianted

**Şema Modelleme Teknikleri**

Normalize Yapı

Fact table vs Dimension Table (Yıldız Şeması)

Snowflake Şema

Fact Constellation

**OLAP**

Çok boyutluluğunun işlevleri:

-Veri kümesinin çok boyutlu tanımlanabilmesi

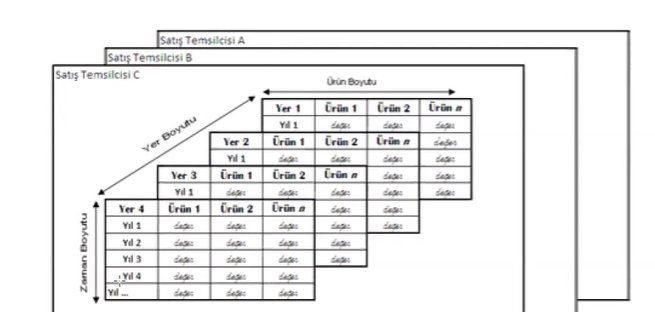
-Çok boyutlu sunum

-Boyutun doğal hiyerarşisini kurabilme

-Boyut üyelerine formül iliştirebilme

Çözümleme esaslı karara dönük işleme

**Dört Boyutlu Veri Yapısı**

Zaman, yer, ürün ve satış temsilcisi boyutu

Multi dimensional table

**ÇOK BOYUTLU TEKNİKLER**

**Dilimleme:** Bir boyutu ele alarak ona denk gelen değeri çekip almak

**Kesme:** 2 boyutun ilişkilendirilmesi

**Döndürme:** marka zaman sınıf- sınıf zaman marka

**Dallan ve Toparlan:** Kendi boyutunda dallanma

**Çapraz Dallanma:** Bir boyuttan başka boyuta dallanma

**FİNAL HAKKINDA**

ER diyagramı verilir, bu diyagramı bir dokuman tabanlı verı tabanına JSON formatında yerleştiriniz. İlikşkilere iyi bak, bire bir çoka çok reference embed collection …

DML DDL … farkları ve komutları, özellikleri…

Bilgi soruları ağırlıklı. Örneğin SAVEPOINT SQL cümleciği yerine işlevi.

Veri ambarı nedir ve özellikleri

JSON yapısını bil.

Bunu alın embed yapısına dönüştürün veya ikiye iki collection yapısına çevirin.

Pattern lardan soru yok.

Tanımlamalardan yapı ve mekanızma