QUERY OPTIMIZATION AND DENORMALIZATION

Veri Tabanı Yönetim Sistemleri HW2, Sevda Çimen *152120131020*¡-this

I. QUERY OPTİMİZASYON

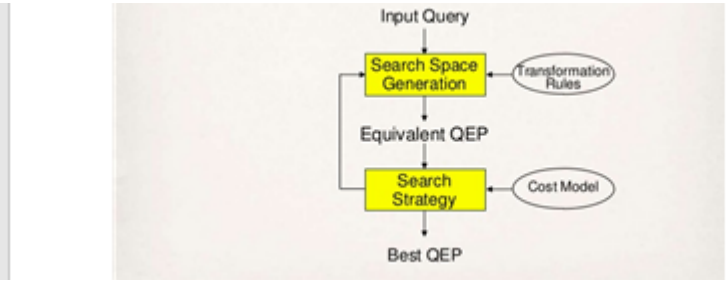
OPTİMİZASYON NEDİR? : Optimizer ya da diğer isimleri ile query optimizer veya cost based optimizer bir veritabanı parçasıdır ve SQL ifadelerinin nasıl koşulması gerektiğinin yolunun çıkartılmasını sağlar. Bütün SQL ifadeleri optimizer'ı kullanmaktadır. "Cost" adı verilen bir miktar hesaplanmakta ve SQL ifadeleri bu yönden koşturulmaktadır.Sorgunun çalıştırılması planını hazırlar. Sorgulama planlayıcısı ve optimize edicisi zor olan düşünme işini yapar. Önce basit optimizasyonlar uygular (5*10'u 50 olarak sadeleştirmek gibi daha iyi performansla sonuçlanan iyileştirmeler). Sonra farklı optimizasyonları olan farklı "sorgulama planlarına" bakar, her bir sorgulama planının ilgili tablolardaki satırlara göre maliyetini (CPU ve süre) tahmin eder, sonra en iyi planı alır ve sonraki adıma geçirir. Basit bir SELECT sorgusu için sadece bir plan vardır ve maliyetsiz (zero cost) olarak tanımlanır. Karmaşık sorgular için 3 aşamalı bir optimizasyon yapılır:

1) Optimizer nested loop join, diğer adıyla nested iteration olup olmadığına bakar. Nested Loop Join: Bir tablodaki her satırı (outer table), diğer tablodaki (inner table) her satırla karşılaştırarak join'in sağlayacağı satırlara karar verir. Algoritmayı pseudo-code olarak yazarsak:

for each row R1 in the outer table
for each row R2 in the inner table
if R1 joins with R2
return (R1, R2)

Eğer burada maliyet 0,2'nin altında ise optimizer bu aşamada durur.
2) Bu aşamada bir plan olup olmadığını anlamak için olası optimizasyonlara ve genel desenlere bakar. Eğer maliyet 1'in altında ise optimizer bu aşamada durur.
3) Son olarak ise bütün optimizasyon kurallarına bakar. Bir DML ifadesinin çalıştırılması için Oracle veritabanı birden fazla aşama devreye sokmaktadır. Mesela bir tablo ya da indekse hangi yollardan ve sırayla erişileceği işleri değiştirebilir. Veritabanının, verilen SQL sorgusunu hangi yollardan çalıştıracağı ve istenilen veriyi getireceği, sorgu performansını oldukça etkilemekte ve sorgunun ne kadar hızlı cevap vereceğini değiştirmektedir. Bu sebeplerden dolayı optimizer'ın hesaplayacağı cost (maliyet) ve sizin sorgunuzun algortiması eşittir ne kadar hızlı sorgu yanıtıdır.

Cost Based Optimizer birkaç yol ile sorgunun nasıl



Şekil 1. Optimizasyon Süreci

çalıştırılacağına karar vermektedir. Herhangi bir SQL sorgusu için Oracle Optimizer aşağıdaki işlemleri gerçekleştirmektedir;
- Predicate (yüklem - WHERE koşulu) yapılandırması ve sorgunun bütün koşulları.
- Integrity constraint'lerinin (sınırlayıcı) gözden geçirilmesi.
- Optimizer hedefleri ve seçimleri.
- Erişim yollarının saptanması.
- Join sıralamasının hesaplanması.

Optimizer bu işlemleri yaparken her bir erişim yolu için bir cost, yani maliyet hesaplar. Bu maliyetlerin tamamına ve erişim yollarının sıralanmış haline "Execution plan" denmektedir. Çalıştırma planı bir Optimizer için yol haritasıdır ve SQL sorgularını nasıl ve hangi yollardan çalıştırması gerektiğini algılamaktadır. Alternatif erişim yolları ile oluşan çalıştırma planları arasından maliyeti en düşük olan plan geçerli kılınır ve ilgili SQL sorgusu çalıştırıldığı zaman o plan üzerinden çalıştırılır.

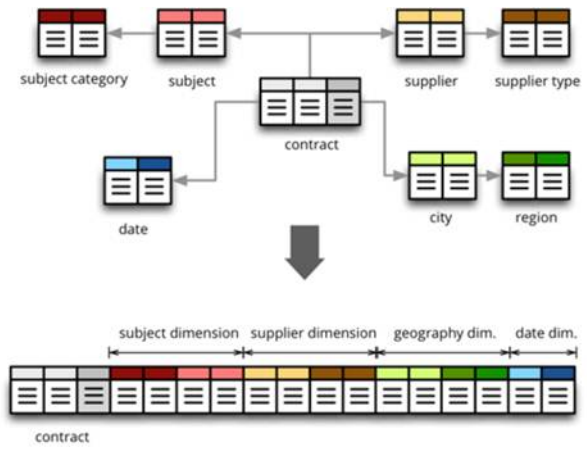
A. QUERY OPTİMİZASYON İLE İNDEX İLİŞKİSİ

SQL Server'da Query Optimizer, bir sorgunun nasıl çalıştırılacağıyla ilgili alternatifleri inceleyip karar veren bir yapıdır. Bu kararı verirken veri dağılım şeklinin çok önemli olduğu durumlar da olur. Mesela bir koşul acaba 100 bin satırdan 3ünü mü döndürüyor yoksa 30 bini mi, bunu bilmesi gerekir indexlemeye ihtiyaç olup olmadığına karar verilsin. İndexlemenin amacı veritabanının başarısını arttırmaktır.Bu yüzden indexkullanılan ver tabanlarında ne kadar çok veri barındırılırsa barındırılsın kolay ve hızlı bir şekilde ulaşım sağlanabilmektir.

Query optimizer ise sorgunun nasıl çalıştırılacağı ile ilgilendiği için indexleme ile daha kolay çalıştıma işlemi sağlanacağından bu iki yapı birbiriyle ilişkilidir.İndexlenmiş tabloların optimize edilmesi sorgu performansını arttırmaktadır. Veriyi daha hızlı bir şekilde sorgulamak için indeksleme ve OLAP gibi teknikler de kullanılır.

II. DENORMALIZATİON

Önce normalize et, eğer performans problemi var ise denormalize işlemi uygula" dır. Denormalizasyon işleminin kesin kuralları yoktur ve veritabanı tasarımcısının sorguların yavaşlamasına karşın bulduğu çözümler olarak düşünülebilir. Denormalizasyon işlemi performans kazanımları düşülüp sadece gerekli tablolara uygulanması yeterli bir işlemdir.



Şekil 2. Denormalizasyon

Denormalization, veri yapısı normalleştirildikten sonra, veritabanı yöneticisinin seçime bağlı yedek veri örneklerini seçici olarak geri eklediği bir ilişkisel veritabanında okuma odaklı veri alım performansını hızlandırmaya yönelik bir yaklaşımdır.

Normalizasyon bir tablodaki veri satır kayıtlarını diğer tablodakilere bağlayarak ilişkilendirmek için anahtar ya da kimlik (id) alanları kullanılması suretiyle tekrarlayan verinin azaltılması anlamına gelmektedir. Basit ancak akılda kalır örnek olarak müşteri ve sipariş tablosu örnek verilebilir. Burada bir tablo ile bir müşterinin siparişlerini depolarken müşteri bilgilerini her sipariş satırında tekrarlamak (denormalizasyon) yerine müşteri bilgilerini tablodaki müşteri no alanını sipariş tablosundaki müşteri no alanına bağlayarak elde edebiliriz.

A. DENORMALİZASYONUN SORGU HIZINA

Denormalizasyon işleminden sonra veri tek satırda tutulduğu için sorgulanmasının daha hızlı olması beklenmektedir Denormalizasyon ile verinin okunması ve sorgulanmasında veri motorunun bilgiyi sağlamak için erişmesi ve işlem yapması gereken tablo sayısını azaltması ile performans sağlamaktadır. ERP gibi yoğun işlemsel hareket (transaction) gören veritabanlarında normalizasyon kullanımı gerekirken, DSS gibi işlemsel hareketlerin minimum olduğu ancak yoğun sorgulama ve raporlama yapılan veritabanlarında ise denormalizasyon kullanımı gerekmektedir.