**4. VIDEO (SQL - 2)**

Veritabanini kullanan ve onlari yoneten insanlar vardir. Biz simdi yonetenlerin oldugu kisma biraz girecegiz programlamayla falan pek alakali olmayacak.

Transactions

Commit – Onaylamak = Siz verilerle ugrasirken veriler havada ucusuyorlar veritabaninin bu verileri gercekten islemesi icin diske yazmasi gerekir veritabani tum verileri diske yazmaz veriler RAM’de durur.

Veritabani olasi hatalara onlem olarak olusturulan yani kaydi yeni veriyi hemen veritabanina yazmiyor.

Siz commit kullanmadan bu veriler diske yazilmaz. Diske yazmiyorsaniz eger o veri bilgisayarinizi kapattiginizda ucuyor gidiyor demektir.

Commit’in cagrildigi her yere veritabani bir nokta bir isaret koyuyor ve rollback ile bu noktalara geri donebiliyoruz. Sistem geri yuklemesi gibi dusun bir tarih verip kaydediyorsun herhangi bir aksilikte o tarihteki bilgisayarina geri donuyorsun ;)

Rollback’I anlik olarak da kullanabiliyorsunuz mesela hesaptan hesaba para transferi yaparken hata olustu para ucmasin diye hemen rollback yapiliyor.

Mesela veritabani uzerinden birden cok kisi calisiyor ve ben bir veriyle ugrasirken atiyorum baska biri de o veriyle ugrasyor ve ne ben ne de o commit etmedi ve birbirimizden habersiziz iste bu gibi durumlarda herhangi bir sikinti olmasin diye veritabani commit islemini kendi islemini kendisi gerceklestirip kim en son nerede kalmissa ona ulasmasini saglayabiliyor. (Two-face commit, Three-face commit daha sonra)

Integrity Constratints

Atiyorum bir veritabanindan gittiniz banka subelerinden birini sildiniz mesela. Bu durumda o subede bulunan yuzlerce hesabin subesi bosa (null’a) dusecektir. Demekki sube silmek o kadar kolay olmamali.

Dusunerek silmeye tesvik eder integrity constraints.

- not null

- primary key (daha cok veriye daha hizli ulasmak icin kullanilir)

- unique

- check (bir verinin bir veri kumesi icinden kontrol edilmesini saglar – checkbox gibi)

Chech (semester in (‘Fall’, ‘Winter’, ‘Spring’, ‘Summer’)) semester isimli attribute bu 4 optiondan farkli bisey olamaz olursa hata aliriz.

Referential Integrity

Primary key ve foreing key arasindaki iliskidir.

**Create table course(**

**course id char(5) primary key,**

**title varchar(20),**

**dept\_name varchar(20) references department )**

**create table course(**

**…**

**dept\_name varchar(20)**

**foreign key(dept\_name) references department**

**on delete cascade**

**on update cascade,**

**…)**

Complex Check Clauses

**Check (time\_slot\_id in (select time\_slot\_id from time\_slot))** 🡪 Daha once girilen verilerden bir deger girmeye zorluyoruz kullaniciyi bu sayede ;)

Built-in Data Types

- date, time, timestamp (date and time), interval

Index Creation

Tree’ler ve Hash’ler indexlerdir. Verilere daha hizli erismeyi saglar indexler ;)

Primary keyler uzerinde index otomatik olarak olusturulur. Her erismeye calistiginizda bir oncekinden daha hizli bir sekilde erisirsiniz.

User-Defined Types

**Create type Dollars as numeric(12,2) final**

(Dollars adinda bir veri tipi olusturuyoruz)

Domains

**Create domain person\_name char(20) not null**

Su domainin icindeki veriler su ozellikleri icerecek diye belirtiyoruz. Type’ler ile benzerdir.

Large-Object Types

Blob = binary large object (videolar, resimler)

Clob = character large object

Mesela videolar, resimler veya cok buyuk metinler veritabaninda tutulmak isteniyor bu durumda bu type’ler kullaniliyor.

Verileri dosyalarda tutup dosya yollarini veritabanlarinda tutmak gibi yaklasimlar da var.

Authorization

Biz veritabani yoneticisiyiz kim ne yapabilir ne yapamaz buna karar veriyoruz.

Privilige list = Read, Insert, Update, Delete, Index, Resources, Alteration, Drop

Yetkiyi grant ile tanimliyoruz.

Grant <privilege list> on <relation name or view name> to <user list>

**Grant select on instructor to U1, U2, U3**

All privileges diyerek butun bu yetkileri verebiliriz. (select yerine yani)

Verdigimiz yetkiyi geriye revoke ile aliyoruz.

Revoke <privilege list> on <relation name or view name> from <user list>

**Revoke select on branch from U1, U2, U3**

All privileges diyerek butun bu yetkileri geri alabiliriz.

Roles

Veritabani uzerinde binlerce kullanici olabiliyor ve yonetici bunlarin hepsini bilmeyebilir bilse bile hepsine tek tek iste yetki ver yetki al bunlarla ugrasamaz.

Bir role tanimliyor mesela programci diyor ve benim veritabanim uzerinde programlama yapan herkesi programciya ekliyor. (Bir tane tanimliyor iste veri ambari, rapor alan kisiler, yedek alan kisiler vs…)

**Create role instructor;**

**Grant instructor to Amit**

**- grant select on takes to instructor**

\*\*\* all privilages yazmayip bos birakirsak da yine tum yetkileri vermis oluruz.

\*\*\* zincir seklinde gidebiliyor iste instructor Amit’e Amit bir baskasina bir baskasi oteki birine falan diye)

Authorization on Views

**Create view geo\_instructor as (select \* from instructor where dept\_name = ‘Geology’);**

**Grant select on geo\_instructor to geo\_staff**

**5. VIDEO (SQL - 3)**

JDBC and ODBC

Programlama dilinden veritabanina erismeyi saglayan teknolojilerdir.

Programlama dili veritabanindan soyutlanmis olmasi gerekir. Atiyorum bugun degisik bir veritabani kullandik sonra baska bir teknolojiyi kullandik tarzi seylere imkan saglamak icin ikisinin ayri olmasi gerekir. (Abstraction)

ODBC (Open Database Connectivity) – C, C++, C#, Visual Basic

JDBC – Java

SQL Injection

Mesela bir uygulamaya girerken kullanici adi ve sifre kismina iste adiyorum bir hoca not girecek sisteme ve kullanici adi ve parolasini girerken yanlislikla kullanici adinin oraya tirnak isaretini koydu. (“select \* from instrcutor where name = ‘” + name + “’ “) Bu sorgudaki tirnak adamin yanlislikla koydugu tirnakla kapatilacagi icin adam bundan sonra istedigi gibi girip query’ler yazabilir hic istenmeyen sekilde veritabanini silebilir veya farkli seyler yapabilir. Buna sql injection deniyor.

Kullanici adini yazarken bu tirnak isaretleri gibi guvenligi engelleyici karakterlerin girilmemesi kontrolu yapilabilir bunu engellemek icin veya java’da mesela bunu engelleyici classlar var bunlari kullanarak bu durumu engelleyebiliriz. (herseyde gecerli 🡪 ADO.Net, jdbc, odbc vs..)

Metadata Features

Alinan bilginin ust bilgisinin alinmasidir. Mesela kullanici adi parolayi falan aldik ama iste kolon isimlerinin yani attribute’lerin isimlerinin alinmasi, tiplerinin alinmasi. Bu tip bilgiler Metadata’dir. Ust bilgidir yani.

Transaction Control in JDBC

Yani commit ve rollback’in programlama dili icinden nasil yazilacagini gosterir.

Conn.setAutoCommit(false) 🡪 AutoCommit’I kapatiyoruz bunu kapatirsak commit’I veya rollback’I elle cagirmamiz gereklidir. Bunu true da yapabiliriz hic ugrasmayarak ama elle yapmak daha faydalidir.

- Conn.commit();

- Conn.rollback();

Blob and Clob

Atiyorum elimizde 10 gb video var ve video arsivi yapiyoruz iste arka plan muzik bilgisi video bilgisi konum bilgisi vs.. giriyoruz veritabanina zaten 2 video yuklesek veritabanina RAM’I hasat edip bilgisayari kullanilmaz hale getirecektir. Bu yuzden bu gibi buyuk dosyalari streamlerle gecici bufferlarla alabiliriz blob ve clob kullanarak.

( Blob.setBlob(int parameterIndex, InputStream inputStream) )

SQLJ

JDBC’ye alternatif olarak kullanilan bir teknolojidir.

ODBC

Butun veritabanlari hatta butun hersey destekliyor. Baglanti teknolojisidir. Ortak bir protokol gibidir.

En yavas ve en kotu calisan teknolojilerdendir. Ama bazen kullanilabilir.

Limitleri ve bazi problemleri var. (Kullanmak bu yuzden cok iyi degil)

ADO.NET

Microsoftun teknolojisidir. (.Net, Asp.net, C#, Visual Basic)

OLE-DB (tavsiye ediliyor, 7-8 kat daha hizli calisiyor), XML data, Entity framework

Embedded SQL

SQL cumlelerini bir programalama dili icerisinden yazmaya yariyor. (C, Java, Cobol)

Procedural Extensions and Stored Procedures – Functions and Procedures

Table-valued functions – tablo dondurebilen fonksiyonlardir.

**Create function dept\_count (dept\_name varchar(20))**

**Returns integer**

**Begin**

**Declare d\_count integer;**

**Select count(\*) into d\_count from instructor where instructor.dept\_name = dept\_name**

**Return d\_count;**

**End**

(Bu fonksiyon bolumde kac tane hoca oldugunu dondurur mesela)

Bunu yaziyoruz bu veritabanimizda duruyor ve her seferinde sorguyla ugrasmadan direkt olarak bu fonksiyonlari cagiriyoruz. (ya da stored procedure leri)

**Select dept\_name, budger from department where dept\_count(dept\_name) > 1** (Fonksiyonumuzu burada kullaniyoruz mesela)

**Table Functions**

**Create function instructors\_of(dept\_name char(20)**

**Returns table (**

**ID varchar(5)**

**Name varchar(20)**

**Dept\_name varchar(20),**

**Salary numeric(8,2))**

**Return table**

**(select ID, name, dept\_name, salary from instructor where instructor.dept\_name = instructors\_of.dept\_name)**

**Select \* from table (instructors\_of(‘Music’))** (Fonksiyonumuzu burda kullaniyoruz)

SQL Procedures

**Create procedure dept\_count\_proc(in dept\_name varchar(20), out d\_count integer)**

**Begin**

**Select count(\*) into d\_count from instructor where instructor.dept\_name = dept\_count\_proc.dept\_name**

**End**

**Declare d\_count integer;**

**Call dept\_count\_proc(‘Physics’, d\_count); (Burada cagiriyoruz procedure mizi)**

- Procedure geriye deger dondermez ama islevsel olarak fonksiyonla ayni seydir.

- Overloading burdada gecerli. (Ayni isim farkli parametreli fonksiyonlar, procedure’ler)

- While, repeat, for looplari

External Language

Diger islemin tam tersi olarak SQL islemlerinden programlama dillerine erisimi saglar.

- sandbox techniques

- SQL ve programlama dilleri o kadar gelistiki birbirlerinin yerini alabiliyorlar.

- Teknolojiyi bilmek cok onemli. (Performans, hiz, guven, kalite durumuna gore avantajlarini belirleyip nasil daha iyiyse onu kullanirsiniz)

- Bazen SQL de hesaplama avantajli oluyor bazen programlama dilinde yapmak daha avantajli oluyor.

Triggers

Bir is yapilmadan once veya sonra otomatik olarak bazi islemlerin calismasini saglar. (Tetikler)

**Create trigger timeslot\_check1 after insert on section referencing new row as nrow**

**For each row**

**When (nrow.time\_slot\_id not in**

**( select time\_slot\_id from time\_slot)) I\* time\_slot\_id not present in time\_slot \*I**

**Begin**

**Rollback**

**End;**

Yukaridaki kodda sayet time\_slot duzenini begenmediysek trigger hemen rollback yapiyor yani son eklenen veriyi siliyor son adima (commit’e) donuyor.

Mesela diyelim ki bir tabloya insert ediyoruz iste bir trigger yazmisiz insert ederken kontrol et diye mesela insert edilirken hemen otomatik olarak trigger calisir ve kontrol eder siz bisey yapmazsiniz.

Iste Mesela Ali isminde birisi gelirse onu ekleme veritabanina tarzi bisey olabilir.

Recursion in SQL

Mesela iste bir dersin on kosulu baska bir ders o dersin kosulu bir baska ders falan tarzi olan durumlarda recursive kullanabiliriz.

**With recursive rec\_prereq(course\_id, prereq\_id) as (**

**Select course\_id, prereq\_id from prereq**

**Union**

**Select rec\_prereq.course\_id, prereq.prereq\_id, from rec\_rereq, prereq where rec\_prereq.prereq\_id = prereq.course\_id )**

**Select \* from rec\_prereq;**

Aggragation

- Ranking (birinci, ikincinin dondurulmesi gibi durumlarda kullanilan fonksiyondur)

**Select ID, rank() over (order by GPA desc) as s\_rank from student\_grades**

Yukaridaki kodda eger atiyorum 5 tane ayni nota sahip kisi olursa bunlarin hepsi birinci olacak ve hemen ardindan gelen kayit 6. Olmus olacak eger bunu istemiyorsak yani 2. Olan kisi niye 6. Gozuksun derseniz rank yerine dense\_rank fonksiyonunu kullanabilirsiniz.

**select ID, dept\_name, rank() over (partition by dept\_name order by GPA desc) as dept\_rank**

Bu kodda yer alan partititon ile atiyorum iste bilgisayar muhendisligi bolumu ile, endustri muhendisligini ayir onlarin kendi arasindaki ranking’ini bana getir demek olur.

Limit – Bu kelime iste sonuclari 20’li 20’li limetlemeye veya iste 30’ar sekilde limitlemeye yarar.

Percent\_rank = yuzdesel olarak rank veriyor.s

Row\_number = basina bir satir numarasi koymaya yariyor. (Genelde cok anlamli degil)

Windowing

Iste mesela gunluk satislariniz, borsa hareketleri, doviz durumlari falan gibi durumlarda iste bir pencere koyuyorsunuz ve atiyorum 3 gunluk pencere alacam diyorsunuz sonra bir kaydiriyorsunuz 3 gun sonra bir daha kaydiriyorsunuz 3 gun boyl boyle gidiyor.

Moving average – Bu ucer gunluk kaydirmalarin ortalamasi (Borsada falan gorebilirsin)

**Select date, sum(value) over (order by date between rows 1 preceding and 1 following) from sales**

2 following desek 1 gun onceki ve 2 gun sonraki verileri alacak bu durumda 1 preceding (1 gun once) ve 1 following (1 gun sonra) verileri alir. Veya gun olarak degilde iste saat olarak alacak her neyse.