

ACID – proprietati la tranzactii

Atomic. Daca la o tranzactie facem mai multe update, insert etc. modificarie la date vor fi aplicate toate sau nici una. Fiecare tranzactie este socotitica ca o unitate care fie e executata, fie nu. Tot ce face tranzactia, o face asupra la toate datele deodata, nu doar la unele.

Consisty – Toate datele respecta toate cerintele impuse, adica domeniul de valori.Tranzaction manager ne ofera garantia de a avea date consistente pana si dupa la tranzactie, adica nicio data nu se pierde nicaieri.

Isolation – izolarea la tranzactii. O tranzactie nu poate vedea ce face alta.

Durability – odata efectuata, modificarile unei tranzactii sunt puse pe memoria permanenta, adica interna, si chiar de sistemul cade, ele mereu vor fi pastrate acolo

* Fenomene la tranzaction izolation level:
* Dirty Read – situatie in care o tranzactie citeste date care inca nu sunt committed. Daca T1 modifica o linie, dar nu da commit, T2 va putea citit aceste valori modificate, desi T1 nu a dat commit, dar daca T1 nu va da commit,ci esueaza si da rollback, T2 a lucrat cu date inexistente.
* Non repeatable read – situatie cand o tranzactie citeste aceeasi linie de 2 ori, dar primeste rezultate difertie. De ex, T1 citeste o linie, si face ce are de facut. Acum T2 modifica linia data. Daca T1 o citeste din nou, va oferi rezultate diferite, dar commited in schimb. Tousi, T1 inca nu a fost inchisa, dar T2 a modificat datele si T1 deja vede modificarile. Asa problema apare din cauza ca T1 preia write lock la liniile cu care lucreaza un query din el, apoi la final de query scoate lockul, dar alte query desi nu pot scrie, ca e luat write lock deja, ele pot citit datele. Deci, fie query1 care modifica o linie de 2 ori, si fie query 2 care citeste de 2 ori linia data. Cand query1 modifica linia, el ia write lock, dar query2 tot poate inca citi linia. El o citeste dupa ce query1 a modificat-o, ne arata rezultatul, insa pana ca el sa o afiseze a 2 oara, deja query1 face a doua modificare, si inca nu a lasat write lock,desi nu conteaza asta, dar oricum query 2 citeste liber a 2 oara si deja de data asta vede cu totul alt rezultat la aceeasi linie.
* phantom read – apare cand o tranzactie extrage mai multe rows de 2 ori, dar rows returnate sunt diferite. De ex, T1 a returnat 5 rows. T1 inca nu e inchisa. Acum, T2 adauga niste linii noi in tabel si se inchide. T1 ia liniile iar, dar deja sunt mai multe. T1 a da lock la liniile cu care lucra, dar asta nu inseamna ca T2 nu poate adauga altele.
* Sunt 4 nivele de izolation:
* READ UNCOMIMITED – o tranzactie poate citi modificarile uncommited facute de alta tranzactie si permite dirty reads
* READ COMMITED – o tranzactie poate citi doar modificarile commited ale altei tranzactii. Tranzactia tine erite lock asupra la liniile cu care lucreza la momentul dat si nimeni nu o poate modifica, doar citit, pana ea nu elibereaza row. Totusi, poate aparea non repeatable read, caci liniile pot fi citite de de alte tranzactii chiar cat lockul e tinut, desi nu s-a terminat tranzactia. De ex, T1 face de 2 ori acelasi select, in 2 queries difertie. Cand ea a facut primul select, inainte de asta T2 poate deja sa modifice datele si sa ia el lockul, insa deocamdata e ok, T1 ne afiseaza un rezultat, caci are acces la read, nu la write. Dar, pana sa faca al 2 select, T2 modifica iar datele la aceleasi linii, si T2 de data asta cand face acelasi select, vede alte date la acelasi query Cand se face doar select, nu se ia niciun lock. Nu e necesar.Poate aparea non repeatable read
* REPEATABLE READ – o tranzactie tine write si read lock asupra la **toate** coloanele si liniile cu care lucreaza,indiferent ca face ceva cu ele la moment sau nu, si nicio alta tranzactie nu le poate modifica sau accesa pana prima nu elibereaza lockul la sfarsit de tranzactie.Adica, lockul prelat de un query e tinut pana la final de tranzactie. Poate aparea phantom read, caci oricum in tabel pot fi adaugate noi linii, si deci acelasi query executat de T1 a doua ora ar putea returna mai multe linii daca T2 adauga.
* SERIALIAZABLE – tranzactiile sunt executate in mod secvential, adica cate una rand pe rand.
* Pentru a modifica nivelul de izolation in Oracle, se foloseste:

SET TRANSACTION ISOLATION LEVEL <LEVEL>

ALTER SESSION SET ISOLATION LEVEL <LEVEL>

* Cel default e READ COMMITTED

**DML STATEMENTS**

* Insert
* Update
* Delete
* Merge – insert si update, adica daca sunt deja prezente, sunt doar updated, daca nu, sunt inserate
* **Atentie! In Oracle nu putem insera mai multe valori folosind Insert!!!**
* **E bine sa scrie commit dupa fiecare tranzactie,sau operatii ce nu depind de altele, desi oricum daca tranzactia odata reuseste, se va da autocommit**

**SELECT**

* ROWNUM – numarul coloanei selectate. Alternativa la limit

Putem folosi ROWNUM pentru a limita cu conditii numarul de tuples, gen WHERE ROWNUM < 5

SELECT \* FROM test WHERE RAWNUM <= 4;

* OFFFSET Nr ROWS – Ignora primele Nr rows

SELECT \* FROM test OFFSET 4 ROWS;

* Oracle are o tabela numita dual, si ea nu are valori sau coloane, e doar penru a testa functii
* Putem pune NULL FIRST sau NULL LAST ca ASC sau DESC
* In Orale SQL server nu putem combina RAWNUM cu OFFSET ROWS, si se pune ROWS FETCH NEXT X ROWs ONLY

SELECT \* FROM test OFFSET 2 ROWS FETCH NEXT 2 ROWS ONLY;

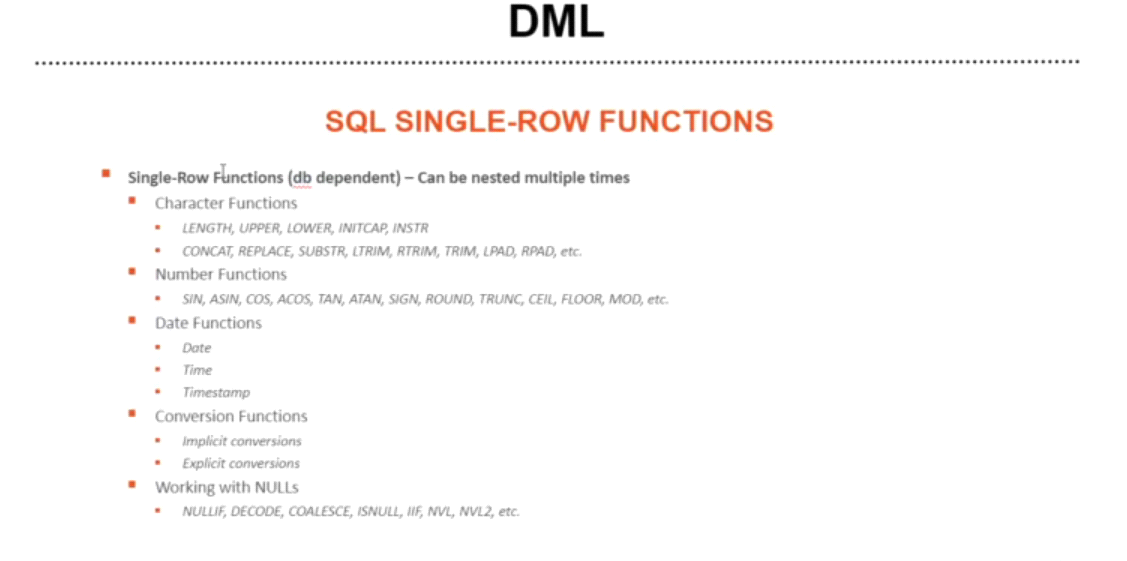
* Nu punem AS dupa numele la tabel:

FROM employees ~~AS~~ E1

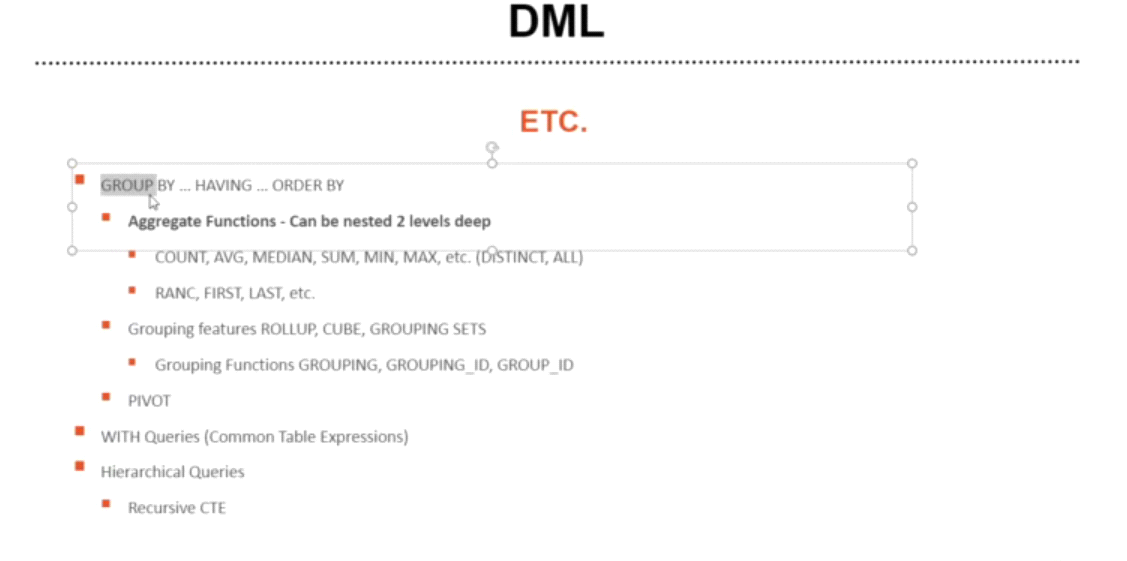
* Putem sa Punem AS la numele la coloana

SELECT E1.first\_name,E2.first\_name AS "Manager\_First\_name"

**Function**



* trim(string) – elimina spatiile libere in plus de la inceput si final
* ltrim(string) – elimina spatiile libere de la inceput
* rtrim(string) – elimina spatiil libere de la final
* lpad(string, size, stringToAdd) – adauga in stanga la string stringul stringToAdd de atatea ori ca stringul final sa aiba size stabilit
* rpad() – ca lpad, doar ca la dreapta
* initcap(string) – adauga litera mare la ineput de fiecare cuvant
* instr(stringInitial, stringDeCautat) – returneaza pozitia la stringul gasit
* substr(string, pozitia, nrCaractere) – extrage nr de caractere de la pozitia indicata,
* round(nr, Pozitie) – va aproxima de la pozitie, gen: round(1.996,2) = 1.997 sau round(1.9994,3) = 1.9990
* ceil(numar) – taie cifrele de dupa virgula si aduna cu 1
* floor(numar) – taie cifrele de dupa virgula si scade cu 1



**Variables**

DEFINE variabila = 10;

SELECT &variabila FROM dual;

**Function**

CREATE FUNCTION Functie

(param1 number, param2 number)

return number AS

BEGIN

return param1+param2;

END;

Dupa parametri, pot fi puse urmatoarele cuvinte cheie:

In – parametrul poaet fi citit, dar nu poate fi modificat

OUT – parametrul nu poate fi citit, dar poate sa i se modifice valoarea

IN OUT – parametrul poate fi si ctit si modificat

Procedura

CREATE PROCEDURE multiplica(ident LONG, valoare varchar)

IS

BEGIN

INSERT INTO test VALUES(ident, valoare);

END;

begin

multiplica(50,'test50');

end;

Atentie! Procedura o executam intre

begin

....

end;

**Procedura vs Function**

* O functie mereu returneaza o valoare cu RETURN, dar o procedura poate returna sau poate sa nu returne ceva
* O funtie poate fi folosita cu SEELCT, INSERT etc. dar o procedura nu poate fi invocata in SQL statements

**Join**

* Dupa Join putem sa folosim using

SELECT first\_name, last\_name, salary, job\_title  
FROM employees  
 LEFT JOIN jobs USING (job\_id);

* Dar putem folosi si on, nu e nicio problema

**Key auto increment**

cardNr NUMBER(6) GENERATED ALWAYS AS IDENTITY (START WITH 10000 INCREMENT BY 1) PRIMARY KEY,

**Triggers**

* Pentru a accesa valorile sterse, folosim :old si cele noi :new
* Pentru a afla daca vreo valoare a fost stearsa sau adaugata folosim:

if inserting, if updating sau if deleting