**Tranzakciók**

Részletesebb infók a doksiban -> Oracle Database Concepts 9-10. fejezet (12.1 doku)

A tranzakcióktól általában elvárt tulajdonságok:

A - atomosság (az adatbázis-kezelő garantálja)

C - a konzisztencia megőrzése (a programozó feladata)

I - elkülönítés (több szintjét is támogatja általában egy rendszer)

D - tartósság (a rendszer garantálja a helyreállító módszereivel)

S - sorolhatóság (a gyakorlatban sokszor túl szigorúnak tűnik)

Három olyan probléma, amelyek a gyakorlatban előfordulhatnak:

1. Piszkos adatok olvasása

Még nem commitált adatokat olvas egy tranzakció.

2. Nem ismételhető lekérdezés

Egy lekérdezés újabb futásakor időközben módosított (és commitált) sorokat olvas.

3. Fantom sorok olvasása

Egy lekérdezés ismételt futtatása olyan sorokat is talál, amelyek előzőleg még nem   
 léteztek.

A fenti problémáknak megfelelően az SQL szabvány 4 elkülönítési szintet definiál, amelyek a fenti esetek közül egyre kevesebbnek az előfordulását engedik meg. Ezek az alábbiak:

Read uncommitted (1,2,3)

Read committed (2,3)

Repeatable read (3)

Serializable (egyik fenti probléma sem fordulhat elő)

Az Oracle a fentiek közül a Read committed és a Serializable elkülönítési szinteket támogatja, továbbá megmondható egy tranzakcióra, hogy az Read Only legyen.

Egy példa a fenti 2. illetve 3. problémára:

T1 T2

1- read(X)

2. write(X)

3. commit

4. read(X)

Kérdés: T1 ugyanazt látja-e a 4. lépésben, mint az elsőben?

Válasz: NEM feltétlenül (attól függ, hogy milyen elkülönítési szint van beállítva)

Ennek oka, hogy az adatbázis-kezelő alapértelmezés szerint az utasítás szintű READ konzisztenciát garantálja, (ez a fentiek közül a második elkülönítési szint), vagyis, hogy az utasítás végrehajtása közben történt változások láthatatlanok az utasítás számára.

Kérdés: Hogyan garantálja az adatbázis-kezelő az utasítás szintű READ konzisztenciát?

Válasz: A ROLLBACK szegmensek segítségével. (Vagy újabb verzióban UNDO táblatérrel.) Mindkettő lényege, hogy a blokknak több példányát (régit és újat) is tárolja a rendszer.

A rollback szegmenseket a rendszer véletlenszerűen rendeli a tranzakciókhoz, úgy, hogy nagyjából egyenletes legyen az eloszlás. Az első DML utasítás kiadásakor rendel csak rollback-et a tranzakcióhoz, így aki nem módosít, ahhoz nem rendel. (Létezik persze explicit hozzárendelés is -> SET TRANSACTION USE ROLLBACK SEGMENT …) Az extenseket (az extens egy adatfájlon belüli szomszédos adatblokkokat jelent) ciklikus sorrendben allokálja a rendszer, ha az előző már megtelt. Egy extensbe egyszerre több tranzakció is írhat, de minden blokk csak egy tranzakció adatait tartalmazza. Ha a tranzakció commit-ált, akkor az extens felszabadul, de a tartalma még nem törlődik, mert még lehet, hogy szükség lesz rá.

Egy lekérdezés elindulásakor a rendszer egy úgynevezett **System Change Numbert** (SCN) rendel a lekérdezéshez. A módosító utasítások is kapnak ilyet, és az általuk módosított blokkokba is bekerül ez a szám. Így az olvasó utasítás csak azokat a blokkokat olvassa be, amelyeknek az SCN-je régebbi, mint az utasításé. A közben módosult blokkok helyett azok régi verzióját olvassa be a rollback szegmensről.

Így persze az is előfordulhat (ha a lekérdezés túl lassú), hogy időközben már a rollback szegmensen sincs meg minden szükséges blokk. Ilyenkor a lekérdezés a következő hibaüzenettel elhal:

„ORA-1555: snapshot too old (rollback segment too small)”

UNDO táblatér esetén úgy kerülhetjük ezt el, hogy egy INIT paraméterrel megmondjuk, hogy mennyi ideig őrizze meg a rendszer a régi blokkokat -> UNDO\_RETENTION

(Megjegyzés: Az INIT vagy inicializációs paraméterek határozzák meg az adatbázis-szerver működésének részleteit. Sokat közülük az instancia elindulásakor lehet csak megadni, de vannak olyanok is, amiket működés közben is meg lehet változtatni.)

A tranzakció szintű READ konzisztenciát is kérhetjük a rendszertől a megfelelő elkülönítési szint beállításával. (READ ONLY illetve SERIALIZABLE) Ezt a beállítást megtehetjük rendszer (system), session, vagy tranzakció szinten. READ ONLY csak ez utóbbi szinten állítható be.

Csak olvasó tranzakció csak SELECT, LOCK TABLE, SET ROLE, COMMIT, ROLLBACK, ALTER SYSTEM utasításokat adhat ki.

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE | READ ONLY| READ COMMITTED

ALTER SESSION SET ISOLATION LEVEL SERIALIZABLE | READ COMMITTED

ALTER SYSTEM …

Egy szerializálható tranzakció a rendszernek azt az időpontbeli állapotát látja, amikor a tranzakció elindult. (plusz természetesen a saját maga által végzett változtatásokat) Ha közben mások módosítanak és commitálnak, azt nem látja. Ha ő is megpróbál olyan adatot módosítani, amit közben egy későbbi tranzakció módosított, (vagyis a szerializálható tranzakció elindulása után történt a commit) akkor a módosító utasítás a következő hibaüzenettel hal el:

„ORA-08177: Cannot serialize access for this transaction”

Ilyenkor a tranzakció eldöntheti, hogy az eddigi munkáját commit-álja vagy rollback-eli, és esetleg újraindítja önmagát. Vagyis a tranzakció korábbi műveletei azért nem vesznek el!!!

Tranzakciók kezelésére szolgáló utasítások: COMMIT, ROLLBACK, SAVEPOINT

**Példa:**

--CREATE TABLE tr\_proba(sorsz NUMBER(4), szam NUMBER, szoveg VARCHAR2(40));

-- INSERT INTO tr\_proba VALUES(1, 10, 'Elso sor');

-- INSERT INTO tr\_proba VALUES(2, 20, 'Masodik sor');

1. ablak 2. ablak

SET AUTOCOMMIT OFF

------------------------> SET AUTOCOMMIT OFF

SELECT \* FROM tr\_proba;

------------------------> SELECT \* FROM tr\_proba;

UPDATE tr\_proba

SET szam=szam+1

WHERE sorsz=1;

------------------------> SELECT \* FROM tr\_proba;

COMMIT;

------------------------> SELECT \* FROM tr\_proba; **(látja az újat)**

----------------------------------------------------------

COMMIT;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

SELECT \* FROM tr\_proba;

------------------------> SELECT \* FROM tr\_proba;

UPDATE tr\_proba

SET szam=szam+1

WHERE sorsz=1;

------------------------> SELECT \* FROM tr\_proba;

COMMIT;

------------------------> SELECT \* FROM tr\_proba; **(nem látja az újat)**

### Zárolások (Lock) az Oracle rendszerben

A zárolásokkal kapcsolatos fontos fogalmak a következők:

Mi a zárolt objektum (sor vagy tábla)?

A zárolás módja (kizárólagos vagy megosztott, esetleg további módok)

Minél kisebb objektumot képes zárolni egy rendszer, annál nagyobb konkurenciát enged meg, viszont annál több adminisztrációra van szüksége a zárolások nyilvántartásához.

Problémák a zárolással kapcsolatban:

**Holtpont** (amikor két vagy több tranzakció egymásra vár)

A holtponttal kapcsolatban kétféle szokásos stratégia létezik:

1. A holtpont megelőzése (ez költséges és nehézkes)
2. A holtpont felismerése és megszüntetése. (Ezt alkalmazza az oracle is)

**Zárak tovagyűrűzése** (lock escalation)

Amikor pl. egy tábla több sorára is sor szintű zárolás érvényes, és a rendszer egy határ után az egész táblára vonatkozó zárolássá alakítja át a sor szintű zárakat.

Ez nagyban megnöveli a holtpont esélyét.

Az oracle soha nem alkalmazza ezt a módszert. (Más adatbázis-kezelők igen)

A zárolással kapcsolatos legfontosabb szempontok az Oracle esetében:

* Automatikus sor szintű zárolást alkalmaz a rendszer.
* Lehetőség van manuális zárolásra is.
* Az olvasások nem zárolják az adatokat semmilyen módon.
* Az olvasásoknak soha nem kell várakozniuk az író utasítások miatt.
* Az író utasításoknak soha nem kell várakozniuk az olvasó utasítások miatt.
* Az író utasítások csak akkor várakoznak más író utasításra, ha ugyanazt a sort érinti a két utasítás.
* A kialakított zárolások a tranzakció végéig élnek. (Rollback vagy Commit)
* Egy SAVEPOINT után megszerzett zárolások a savepointig való rollback-elés esetén felszabadulnak.

Zárolási típusok az oracle-ben:

DML lock (sor vagy tábla szintű, az adatokat védi)

DDL lock (az objektum szerkezetét és definícióját védi, részletesebben lásd később)

Belső lock és latch (a rendszer belső adatstruktúráit védi, pl. SGA-t, adatszótárat)

Elosztott zárak (párhuzamos szerver esetén az instanciák közötti védelmet biztosítja)

Az Oracle nagyon sokféle zártípust használ, számunkra csak TM és TX típusok lesznek érdekesek. A lehetséges zárak típusai fel vannak sorolva a V$LOCK\_TYPE táblában.

Zárak szintje

Ez azt határozza meg, hogy mekkora adatmennyiséget érint a zár, például egy sort vagy egy egész táblát.

Az oracle automatikusan sor szintű zárat helyez el az érintett sorokra a következő utasítások esetén: **INSERT, DELETE, UPDATE, SELECT … FOR UPDATE**

Ezek a zárak a tranzakció végéig élnek.

A sor szintű zárakon kívül az oracle automatikusan tábla szintű zárat is elhelyez az érintett táblákra a fenti utasítások esetén. Az első 3 utasítás esetén RX, az utolsó utasítás esetén RS módú zárat (újabb verziók már itt is RX-et. lásd alább) Tábla szintű zárat a fentieken kívül még a **LOCK TABLE** utasítás hatására helyez el az oracle.

Zárolási módok

Ez azt határozza meg, hogy mennyire erős a zár, vagyis milyen párhuzamos műveleteket enged meg a rendszer a többi tranzakciónak.

Összesen ötféle tábla szintű zárolási módot támogat a rendszer, amelyeket a LOCK TABLE utasítás megfelelő paraméterezésével manuálisan lehet előidézni. A zárójelben megadtuk azt a számértéket is, ahogyan a rendszerkatalógusban (V$-al kezdődő nézetek) a zárolási mód tárolva van.

(V$LOCK tábla lmode oszlopa)

(RS -> 2) LOCK TABLE <tábla> IN ROW SHARE MODE

(RX -> 3) LOCK TABLE <tábla> IN ROW EXCLUSIVE MODE

(S -> 4) LOCK TABLE <tábla> IN SHARE MODE

(SRX -> 5) LOCK TABLE <tábla> IN SHARE ROW EXCLUSIVE MODE

(X ->6) LOCK TABLE <tábla> IN EXCLUSIVE MODE

A zárolási módok jelentése nagyjából a következő:

#### RS Későbbi módosítási szándék jelzése, vagyis ne sajátítsa ki senki a táblát.

RX Néhány sort módosítok éppen.

S Az egész táblát olvasom, ne módosítsa senki, de olvashatja más is.

SRX Ne módosítsa senki a táblát, én viszont épp azt teszem néhány sorral.

X Az egész táblát módosítom.

A LOCK TABLE és a SELECT … FOR UPDATE utasítás esetén megadható a **NOWAIT**, ez esetben nem vár a rendszer arra, hogy a zárolás sikerüljön. Ha nem sikerült azonnal, akkor az ORA-00054: hibával hal el az utasítás.

**Lekérdezések a rendszer által fenntartott aktuális zárakról:**

A rendszerbe éppen bejelentkezett felhasználók munkameneteit a v$session tábla tartalmazza.

Minden munkamenetnek egyedi azonosítója van: SID (session id). A saját felhasználónk munkameneteit a következő lekérdezéssel kapjuk meg:

SELECT \* FROM v$session WHERE username=USER;

Az aktuális munkamenetünk azonosítóját pedig a következővel:

SELECT sys\_context('userenv', 'sid') FROM dual;

Melyik session milyen zárolást tart fenn jelen pillanatban és mióta (CTIME)?

SELECT se.sid, se.username, lo.type, lo.lmode, lo.ctime

FROM v$lock lo, v$session se

WHERE se.sid = lo.sid AND username = 'NIKOVITS';

SID USERNAME TYPE LMODE CTIME

-------------------------------------------

305 NIKOVITS TM 3 18

305 NIKOVITS TX 6 18

A TM táblaszintű zárolást jelent. Ha több táblát zárolok, akkor több ilyen zár is látható lesz a fenti eredményben.

A TX tranzakciónként csak egyszer fordulhat elő, és azt jelzi, hogy valamely sorokra sor szintű zárat helyezett el a tranzakció.

(Összes zártípus -> lásd V$LOCK\_TYPE táblában.)

Melyik session vár éppen egy zárolásra (REQUEST > 0), melyik session-re várnak éppen (BLOCK = 1), és milyen régen birtokolja a zárat, illetve várnak rá (CTIME)?

SELECT se.sid, se.username, lo.type, lo.lmode, lo.request, lo.ctime, block

FROM v$lock lo, v$session se

WHERE se.sid = lo.sid AND username = 'NIKOVITS';

SID USERNAME TY LMODE REQUEST CTIME BLOCK

--------- ---------- -- --------- --------- --------- ---------

16 NIKOVITS TM 3 0 4499 0

16 NIKOVITS TX 6 0 4499 **1**

29 NIKOVITS TX 0 **6** 186 0

29 NIKOVITS TM 3 0 186 0

Mely objektumokat tartja zárolás alatt valamelyik session jelen pillanatban?

SELECT lo.oracle\_username, lo.session\_id, lo.locked\_mode,

db.object\_name, db.object\_type

FROM v$locked\_object lo, dba\_objects db

WHERE lo.object\_id = db.object\_id and oracle\_username = 'NIKOVITS';

ORACLE\_USERNAME SESSION\_ID LOCKED\_MODE OBJECT\_NAME OBJECT\_TYPE

----------------- ---------- ----------- ------------ -------------

NIKOVITS 16 3 TR\_PROBA TABLE