KOMPONENTE SUBP

(1) ***Baza podataka*** – podaci, metapodaci, baza indeksa

(2) ***Sistem za upravljanjem skladištenjem podataka*** – upravljanje datotekama i upravljanje baferima

(3) ***Ulazi u BP*** – upiti, aplikacije, održavanje šeme BP. Preko jezika BP (DL) koji se sastoji od DDL i od DML.

(4) ***Upravljanje transakcijama i oporavkom*** – da obezbedi da BP ostane u konzistentnom stanju u konkurentnoj obradi podataka (rešavanje konflikata).

TRANSAKCIJA I ACID OSOBINE

-Niz operacija nad bazom – izvršenje jedne logičke jedinice posla u realnom sistemu.

(1) ***Atomnost*** – sve operacije uspešne ili nijedna – ostvaruje se preko commit i rollback.

(2) ***Konzistentnost*** – transakcija je jedinica konzistentnosti. Za vreme obavljanja transakcije može da se naruši ali ne i pre i posle.

(3) ***Izolacija*** – kada se 2 transakcije istovremeno izvršavaju njihovi efekti moraju biti izolovani

(4) ***Trajnost*** – efekti transakcije ne mogu biti izgubljeni.

ANSI/SPARC ARHITEKTURA

-Tronivovska arhitektura čiji nivoi imaju za cilj da odvoje logičku od fizičke i aplikacione strukture BP.

(1) ***Interni nivo*** (*fizički*) – kako su podaci fizički organizovani – IDDL (Internal DDL)

(2) ***Konceptualni nivo*** (*šema BP*) – definiše opštu logičku strukturu, sve podatke u sistemu i njihove logičke odnose i koji treba da omogući upravljanje podacima kao zajedničkim resursom u celom sistemu.

(3) ***Eksterni nivo*** (*podšema*) – definiše specifičnu logičku strukturu, za specifične zahteve.

MODELI PODATAKA I 4 OSNOVNE KOMPONENTE

-Intelektualni alati pomoću kojih se modeluje sistem kao skup objekata, njihovih atributa i veza.

(1) ***Struktura modela*** – skup koncepata za opis objekata, atributa i veza

(2) ***Ograničenja*** – na vrednosti podataka, moraju uvek da budu zadovoljena.

(3) ***Operacije*** – nad konceptima strukture, preko kojih se prikazuku i menjaju elementi BP

(4) ***Dinamička pravila integriteta*** – definiše se osnovno dinamičko ponašanje. Din.pravilo se može predstaviti trojkom <operacija, ograničenje, akcija>

OPERACIJE U MOV-U

-Insert, Delete, Update, Connect (pojavljivanja O1 klase A sa pojavljivanjem O2 klase B), Disconnect, Reconnect.

-Postoje i 2 operacije pretraživanja – navigacione operacije i upitni jezik MOV-a.

OPERACIJE NAD RELACIJAMA – RELACIONA ALGEBRA

-Konvencionalne skupovne operacije:

(1) ***Unija*** ($R_3 := R_1 \cup R_2$) – daje sve elemente obe relacije, bez dupliranja

(2) ***Presek*** ($R_3 := R_1 \cap R_2$) – daje sve elemente koji se pojavljuju u obe relacije (zajedničke)

(3) ***Razlika*** ($R_3 := R_1 - R_2$) – daje sve elemente prve relacije koji nisu elementi druge

(4) ***Dekartov proizvod*** ($R_3 := R_1 \times R_2$) – daje parove koje čine jedna n-torka prve i jedna n-torka druge relacije (npr. prva ima 1 2 3, druga A B pa C D, rezultujuća će imati 1 2 3 A B pa 1 2 3 C D).

-Specijalne relacione operacije

(1) ***Projekcija*** – $\pi_x(R) = \{x|\exists y, \langle x, y\rangle \in R\}$ – unarna operacija koja vadi vertikalni podskup iz neke relacije (npr. ID, ime, starost; odatle radimo $\pi_{ime,starost}(tabela)$ i dobijamo listu u kojoj su sva prva pojavljivanja nekog imena i njihova starost)

(2) ***Selekcija*** - $\sigma_\theta(R) = \{x|x \in R\ AND\ \theta(x)\}$ – unarna operacija koja selektuje n-torke koje zadovoljavaju neki uslov (vadi horizontalni podskup – npr. $\sigma_{starost>20,mesto="Beograd"}(R)$).

(3) ***Spajanje*** (JOIN) - $R_1[\ _x\theta]R_2 = \{\langle x, y\rangle|x \in R_1\ AND\ y \in R_2\ AND\ \theta(x, y)\}$ – binarna operacija koja spaja dve relacije na taj način da se u rezultatu pojavljuju oni parovi n-torki jedne i druge relacije koje zadovoljavaju neki zadati uslov (npr. $R_1[R_1.ID = R_2.ID]R_2$ će spojiti n-torke sa istim ID-jevima).

(4) ***Deljenje*** - $A(Y, Z)[Y \div Z]B(Z) = \pi_x A - \pi_x\big((\pi_x A \times B) - A\big) = R(X)$ – operacija pogodna za upite sa rečima svi, sve, sva. Rezultat je n-torka x koja uzima vrednosti iz A.X, a par $\langle x, y\rangle$ postoji u A za sve vrednosti y koje se pojavljuju u B(Z) (npr. prva sa P1 P2 P3, druga sa brojevima indexa 1 P1, 1 P2, 2 P1, 3 P3, 1 P3, a $R_2[R_2.predmet \div R_1.predmet]R_1$ daje rezultat tabelu „brInd 1“ jer se deli redom i uzimaju se samo one kolone koje imaju za sve tri iz prve isti brInd i ta kolona se vraća).

-Dodatne operacije rel.algebre koje se definišu zbog postojanja nula vrednosti:

A	B
a	b
a	?
?	b
?	?

A	B
x	y
?	b
?	?

A	B
a	b
a	?
?	b
?	?
x	y

(1) ***MAYBE_SELECT*** ($?\sigma$) – selektuju se one n-torke za koje se predikat sračunava u nula vrednost – $R :=?\sigma(R_3)$ će dati u dve kolone (A i B) ? B, ? ?.

(2) ***MAYBE_JOIN*** – u rezultatu se pojavljuju one n-torke za koje se predikat sračunava u nula vrednost $R_a = R_b[?A =?D]R_c$ – daje rezultat sa kolonama A B C D E, i redom: a1 b1 c1 ? e1, a1 b1 c3 ? e3, a2 b2 c1 ? e1, a2 b3 c3 ? e3, ? b3 c1 ? e1, ? b3 c2 d2 e2, ? b3 c3 ? e3.

A	B
a1	b1
a2	b2
?	b3

C	D	E
c1	?	e1
c2	d2	e2
c3	?	e3

(3) ***OUTER_JOIN*** – ako hoćemo da uradimo operaciju ekvispajanja relacija R1(A,B) i R2(C,D) po A=C a $\pi_A(R1) \neq \pi_C(R2)$ tada će se u rezultatu gubiti podaci. U tom slučaju ih spoljno spajanje može sačuvati. Postoje 3 vrste spoljnog spajanja koje prikazujemo preko sledećih tabela:

A	B
1	2
2	1
4	?

C	D
3	4
2	2

a) ***Levo spoljno spajanje*** ($\boldsymbol{\Lambda\Sigma\Sigma}$) – navode se sve n-torke relacije R1 a za one koje se ne spajaju sa n-torkama R2, atributi iz R2 dobijaju nula vrednost (npr $R3 := R1[\Lambda\Sigma\Sigma\ R1.A = R2.C]R2$ bi dalo tabelu sa kolonama A B C D i to redom: 1 2 ? ?, 2 1 2 2, 4 ? ? ?).

b) ***Desno spoljno spajanje*** ($\boldsymbol{\Delta\Sigma\Sigma}$) – navode se sve n-torke relacije R2 a za one koje se ne spajaju sa n-torkama R1, atributi iz R1 dobijaju nula vrednost (npr $R4 := R1[R1.A = \Delta\Sigma\Sigma\ R2.C]R2$ bi dalo tabelu sa kolonama A B C D i to redom: ? ? 3 4, 2 1 2 2).

c) ***Centralno spoljno spajanje*** ($\boldsymbol{T\Sigma\Sigma}$) – pojavljuju se sve n-torke obe relacije a one koje nedostaju dopunjavaju se nula vrednostima. $R5 := R1[\mathrm{T}\Sigma\Sigma\ R1.A = \mathrm{T}\Sigma\Sigma\ R2.C]R2$

(4) ***OUTER_UNION*** – operacija unije se može izvesti samo nad relacijama koje su kompatibilne. Dodavanjem novih atributa i postavljanjem na nula vrednost dve nekompatibilne relacije se mogu učiniti kompatibilnim. $R5 \coloneqq R1\ OUTER\_UNION\ R2$ (ako je R1 A B C sa 2 reda a R2 A D sa tri reda onda ce R5 biti A B C D sa 5 unosa.

RELACIONI RAČUN N-TORKI

-Predikatski račun prvog reda u kome promenljive uzimaju vrednosti n-torke relacija date BP.
-**Sud** – afirmativna rečenica koja je ili istinita ili neistinita. **Predikat** je af.reč. koja ima smisla i sadrži promenljive, a postaje sud kada te promenljive dobiju konkretnu vrednost. **Predikatski račun** je mat.teorija čiji su objekti predikati. **Atomi** (atomske formule) su simboli suda.
-Formule se formiraju preko **pravila**:

(1) Atom je formula
(2) P1 je formula ⇒ i NOT P1 je (P1) su formule
(3) P1 i P2 su formule ⇒ P1 AND(OR) P2 su isto formule
(4) P1(S) – formula koja sadrži slobodnu promenljivu S ⇒ i $\exists P1(S)$ i $\forall P1(S)$ su isto formule.

-**Kvantifikatori**: (1) ***egzistencijalni*** (∃) – daje true ako postoji bar jedna vrednost X sa koju se predikatska formula F sračunava u true i (2) ***univerzalni*** (∀) daje true ako je za svako X F true.
-**Opšti iskaz relacionog računa**: Neka su R1..Rn relacije u nekoj BP, A..:C atributi ovih relacija respektivno i F formula.: $\frac{t: R1, u: R2, \dots, v: Rn}{t.A, u.B, \dots v.C \text{ WHERE } F}$

-Primer – Prikaži brInd i imena studenata starijih od 20 god: $\frac{t: Student}{t.brInd, t.ime \text{ WHERE } t.starost > 20}$

RELACIONI RAČUN DOMENA

-Promenljive uzimaju vrednosti iz nekih domena atributa relacija posmatrane relacione BP.
-**Uslov članstva**: R(izraz, izraz...), gde je izraz oblika A:v, gde je A atribut relacije a v ili neka promenljiva ili konstanta. Uslov članstva se sračunava u true ako postoji n-torka u relaciji R koja ima zadate vrednosti navedenih atributa (npr. Student(brind='12', šSmera=01) će biti true ako postoji n-torka u relaciji student sa navedenim vrednostima).
-**Opšti iskaz relacionog računa domena**: $x, y, \dots z$ WHERE F.
-Primer – Prikaži brInd i imena studenata starijih od 20 god:
x, y WHERE $\exists z > 20\ and\ Student(brind: x, ime: z, starost: z)$

QBE (QUERY BY EXAMPLE)

-implementacija relacionog računa domena. Ima dvodimenzionalnu sintaksu:
-**Postupak rada**:

(1) Ako se unese u rubriku naziv relacije, u rubrukama tog reda će se ispisati imena atributa.
(2) U kolonama ispod se navode operacije koje se odnose na celu n-torku. Koriste se sledeće operacije: print(P), insert(I), delete(D), update(U).
(3) Uslovi u istom redu se spajaju sa AND
(4) Ako se želi OR – mora se pisati u različitim redovima.

PRAVILA INTEGRITETA MODELA

(1) ***Integritet entiteta*** (ključa) – Nijedan atribut koji je primarni ključ ili njegov deo ne može da bude null.

(2) ***Referencijalni integritet*** – Ako R2 ima spoljni ključ koji je vezuje sa R1 preko primarnog ključa, tada vrednost FK mora da bude jednaka PK ili null. R1 i R2 ne moraju biti različite. Primer: Student (brInd, šSmera), Smer(šSmera), Prijava (šPrijave, brInd) - $\pi_{\check{s}Smera}(Student) \subseteq \pi_{\check{s}Smera}(Smer) \cup \{Null\}$.

-Ova pravila se navode zajedno sa definicijom spoljnjeg ključa u opisu relacije:

$FOREIGN\ KEY\ (lista\ agributa\ koji\ ga\ \check{c}ine)$
$REFERENCES\ naziv\ relacije\ primarnog\ ključa$
$DELETE\ opcija$
$UPDATE\ opcija$
$opcija :: RESTRICT | CASCADE | SET\ NULL\ |\ SET\ DEFAULT$

POSLOVNA PRAVILA INTEGRITETA

-Pravila integriteta stanja

(1) ***Pravila integriteta za domene*** (domen je skup vrednosti atributa i može biti predefinisani ili semantički).

(2) ***Pravila integriteta za atrubute*** – definišu se preko sledeće četvorke ⟨naziv atributa, domen, ograničenje, akcija⟩. Ograničenje je predikat nad atrubutom, a akcija može biti samo odbijanje operacije koja je prouzrokovala narušavanje uslova, te se zbog toga ona obično ne navodi.

(3) ***Pravila integriteta za relacije*** – omogućava vezivanje vrednosti jednog za vrednost drugog atributa u jednoj relaciji (CREATE INTEGRITY RULE naziv_ograničenja predikat). Promenljive u predikatu mogu biti samo atributi jedne relacije. Atributi se označavaju dot notacijom.

(4) ***Pravila integriteta za bazu*** – preko njih je moguće iskazati bilo kakvo složeno ograničenje na vrednosti atributa u BP, ograničenje koje povezuje vrednosti atributa iz više relacija.

-Pravila integriteta prelaza iz stanja u stanje – ako se uvede i konvencija da se oznakom " ' " označavaju promenljive koje uzimaju vrednosti iz relacija pre operacije, a bez te one koje uzimaju vrednosti iz relacije posle posmatrane operacije, može se iskazati i ograničenje prelaza iz stanja u stanje preko istog relacionog računa n-torki.

SPOLJNO SPAJANJE U SQL-U

-OUTER JOIN se koristi da bi se u rezultat spajanja uključili i oni redovi koji ne zadovoljavaju uslov.

(1) ***LEFT*** – uključuje sve sa leve strane JOIN-a tako što se praznim redom proširuje tabela sa desne strane.

(2) ***RIGHT*** – uključuje sve sa desne strane JOIN-a tako što se praznim redom proširuje tabela sa leve strane.

(3) ***FULL*** – uključuje sve redove iz obe i proširuje obe praznim redom

-Primer:
$SELECT *$
$FROM\ A\ LEFT[OUTER]JOIN\ B$
$ON\ A.ID = B.ID$
$(USING\ ID)$

POGLED U SQL-U (VIEW)

-Ako dodelimo ime tabeli koja je rezultat nekog upita:

$$CREATE\ VIEW\ APROSEK(A1, prosekA2, sumaA3)$$
$$SELECT\ A1, AVG(A2), SUM(A3)$$
$$FROM\ A$$
$$GROUP\ BY\ A1$$

-Osnovne prednosti u korišćenju pogleda:

(1) ***Jednostavnost korišćenja*** – definisane tabele pogleda mogu da se stave jednostavno na uvid korisnicima.

(2) ***Sigurnost podataka*** – pogledi su moćan mehanizam kontrole pristupa

(3) ***Performanse*** – pogled se čuva u kompajliranom obliku – brz rad

(4) ***Nezavisnost podataka*** – za ostvarivanje nezavisnosti programa od baze podataka.

-Uslovi koje treba da isputni pogled da bi poslužio za ažuriranje: u FROM samo jedna tabela; nema HAVING, GROUP BY ni DISTINCT; WHERE ne može da sadrži podupit nad tabelom iz FROM-a; sve kolone u upitu moraju biti bazne kolone i moraju biti uključene i sve not null kolone tabele nad kojom je pogled definisan.

OGRANIČENJA U SQL-U

(1) ***Ograničenja domena*** – CHECK – na sve kolone nad jednim domenom

(2) ***Ograničenja tabele*** – UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK

(3) ***Opšte ograničenje*** – CHECK – uslov nad podacima više tabela; ako je zadovoljeno – TRUE ili UNKNOWN.

-**UNIQUE** – true akko ne postoje 2 reda sa istim NOT NULL vrednostima u kolonama gde je sprecifirano ograničenje.

$$CREATE\ TABLE\ A($$
$$A1\ CHAR(13)\ UNIQUE,$$
$$\vdots$$
$$A2\ CHAR(13)\ UNIQUE\ NOT\ NULL)$$

-**PRIMARY KEY** – true ako ne postoje 2 ista i nema null... $A3\ INTEGER\ PRIMARYKEY$

-**FOREIGN KEY** – referencirana i referencirajuća kolona i tabela.

$$A4\ INTEGER\ NOT\ NULL\ REFERENCES\ B(A4)$$
$$ili$$
$$CONSTRAINT\ ODELJENJE\ FOREIGN\ KEY(ODELJENJEID)$$
$$REFERENCES\ ODELJENJE\ (ODELJENJEID)$$

-**CHECK** – definiše uslov ograničavanja dozvoljenih vrednosti kolona, međuzavisnosti kolona i redova ili opšte ograničenje:

$$IME\ VARCHAR(20)\ NOT\ NULL\ CHECK\ (IME = UPPER(IME))$$
$$ili$$
$$CONSTRAINT\ ODELJENJE\ CHECK(SELECT\ COUNT(*)$$
$$FROM\ UCENIK) <= 100)$$
$$CREATE\ ASSERTION\ name$$
$$CHECK\ (uslov)$$

-Provera ograničenja – da bi se provera odložila za kraj transakcije stavlja se DEFERRABLE. Suprotno sa NOT. Može biti INITIALY DEFERRED, kada se na početku odloži i INITIALY IMEDIATE, kojim se može odložiti provera do kraja transakcije korišćenjem SET CONSTRAINTS.

ARHITEKTURA OBJETNIH SUBP

(1) ***Objektni model*** – ODMG model, izveden iz OMG koji je objekti model definisan kao zajednička osnovna za objektne PJ, komunikaciju objekata u kl-server arh. i objektne baze podataka.

(2) ***Objektni specifikacioni jezici*** – ODL i OIF (Object Interchange Format).

(3) ***Objektni upitni jezik*** – OQL – ima znatno veće mogučnosti od SQL-a zbog prednosti objektnog modela.

(4) ***Language binding*** – C++, Java i Smalltalk podrška za manipulaciju perzistentnim objektima, operisanje preko OQLa, upravljanje transakcijama i sl.

-Slika – levo – specifikacija ODL -> preprocesor (koji pokazuje na prevodilac sa desne) i baza na dnu (povezana pristupom podacima sa aplikacijom). Sa desne – izvorni kod -> prevodilac -> binarni kod (u koji ulazi i runtime) -> linker-> aplikacija.

NASLEĐIVANJA U ODMG-U

(1) ***Nasleđivanje ponašanja*** – koristi se veza nadtip-podtip (ISA veza). Podtipovi nasleđuju operacije nadtipova Npr: INTERFACE A{…}, INTERFACE B:A{…}. Podržano je višestruko nasleđivanje ponašanja ali ne i overloading.

(2) ***Nasleđivanje stanja*** – extends – nasleđuje se celpkupno stanje i ponašanje. Veza extends je tranzitivna. Slika – UML dijagram na vrhu Interface A – vezan za njega preko ISA Interface B – pa preko ISA C, pa preko extends D.

OQL - ULAZNE TAČKE, DEFINICIJA PROMENLJIVIH I OSNOVNI UPISNI BLOK

-**Ulazne tačke** – bilo koji opseg neke klase, bilo koji perzistentni objekat čije je ime definisano preko BIND

-OQL **ima isto SELECT, FROM, WHERE**, a može i preko imena objekta da vrati taj objekat.

-Ako se referencira kolekcija mora se staviti i **iterator** u FROM, npr: STUDENTI X (X IN STUDENTI, STUDENTI AS X), prvi deo vraća set<string>, ili će sledeća stvar u FROM vratiti bag<string>:

$$SELECT\ X.Ime$$
$$FROM\ STUDENTI\ X$$
$$WHERE\ X.Pol = "M"$$

-**Izraz putanje u OQL-u** – Koristi se za konstrukciju uputa. Omogućava da se od ulazne tačke dođe do nekog objekta. Definiše se dot notacijom: Student.ImenaSt – ovo je literal tipa string

-**Struktura kao rezultat OQL upita** – može da se vrati i kompleksna struktura koja se definiše u samom upitu.

OBJEKTNO-RELACIONI MODEL

-**Karakteristike**:

(1) ***Mogućnost definisanja novih tipova***. Atribut relacije može biti definisan nad osnovnim predefinisanim tipom. Mogućnost definisanja novog tipa kao seta. Relacija sa atributom čija je vrednost nova relacija – ugnježdena relacija.

(2) ***Mogućnost definisanja metoda*** – njima se definišu operacije nad korisničkim tipovima.

(3) ***N-torke imaju ulogu objekata*** – svaka n-torka može da ima identifikator. Postojanje ovog identifikatora omogućuje dva načina povezivanja:

a) *Konvencijalni* – na osnovu vrednosti primarnih i spoljnih ključeva

b) *Objektni* – referenciranjem identifikatora povezanih n-torki.

KORISNIČKI DEFINISANI TIPOVI O-R MODELA

(1) **DISTINCT tip** – jednostavni, perzistentni korisnički def. tip čijim je uvođenjem podržalo strogo tipiziranje. Nije podržano nasleđivanje – mora da bude FINAL. npr:

$$CREATE\ TABLE\ SOBA(SOBAID\ CHAR\ (10),\ DUŽINA\ INTEGER,\ ŠIRINA\ INTEGER,\ POVRŠINA\ INTEGER)$$

-U ovom slučaju bi se definisalo CREATE TYPE METAR AS INTEGER FINAL; i CREATE TYPE KVMETAR AS INTEGER FINAL čime bi se omogućilo postojanje ograničenja da se ne može pri nekom update-u reći ŠIRINA = POVRŠINA. Ako nam je namera da manipulišemo dalje međusobno ovim podacima, to se može uraditi i npr. preko CAST(DUŽINA AS INTEGER)* CAST(POVRSINA AS INTEGER).

(2) **Struktuirani tip** – perzistentni imenovani tip sa jednim ili više atributa. Za definisanje se koriste metode preko kojih je omogućeno učaurenje. Deklaracija metode se navodi nakon deklaracije tipa. Metode mogu biti originalne i redefinisane, moraju biti navedene u definiciji tipa. Pozivaju se preko dot notacije. Ključne reči:

-UNDER ime_nadtipa – omogućava da str.tip bude podtip nekog drugog str.tipa

-NOT INSTANTIABLE – ne dozvoljava generisanje konstruktor metode i ne može se instancirati, ali njegovi podtipovi mogu (ako nemaju isto ovo ograničenje).

-NOT FINAL – uvek stoji, jer str.tip nije konačan i može da ima podtipove.

KONSTRUISANI TIPOVI O-R MODELA

(1) ***Referentni tip*** – tabele nad struktuiranim tipovima mogu imati i referentnu kolonu koja služi kao identifikator. Navodi se kao REF IS naziv_atributa način generisanja. Način generisanja može biti SYSTEM GENERATED ili DERIVED.

-Ako je X vrednost tipa REF(T) tada ntorku t ili neku njenu komponentu možemo dobiti na 2 načina – preko operatora -> (x->A je vrednost atributa u ntorki t); ili korišćenjem operatora DEREF koji se primenjuje na referencu i vraća referenciranu n-torku.

(2) ***Tip vrsta*** – niz polja koje čine parovi $\langle naziv\ podatka, tip\ podatka\rangle$. Oblika: ROW (ULICA CHAR (30), BROJ INTEGER, GRAD CHAR(30)).

(3) ***Kolekcija*** – grupa koja se sastoji od nula ili više elemenata istog tipa. Broj elemenata – kardinalnost kolekcije. Tip kolekcije je određen vrstom i tipom elemenata kolekcije. Nisu podržani višedimenzioni nizovi u SQL:1999 standardu. Dva niza su uporediva samo ako su tipovi njihovih elemenata uporedivi. Npr. CLAN CHAR (20) ARRAY[20].

SEMANTIKA ECA PRAVILA

-**ECA pravila** su pravila u aktivnim bazama podataka, produkciona pravila, ON događaj IF uslov DO akcija. Sastoje se od trojki $\langle event, condition, action\rangle$. Razlikujemo primitivne i složene događaje:

-Primitivni događaji:

(1) ***Ažuriranje podataka*** – u onim aktivnim sistemima koji su nadogradnja relacionih to su insert, uprade i delete, a u onim koji su nadogradnja objektnih – kreiranje brisanje ili izmena nekog objekta kao i poziv metode koja modifikuje objekte.

(2) ***Prikaz podataka*** – od aktivnih – SELECT, od objektnih – poziv metode za prikaz objekta.

(3) ***Vremenski događaj*** – apsolutan ili periodičan.

(4) ***Aplikativno definisan događaj*** – određuje ga sama aplikacija.

-Složeni događaji

(1) ***Logički operatori*** – kombinacija AND, OR ili NOT

(2) ***Sekvenca*** – ECA pravilo se može pokrenuti nakon određene sekvence događaja.

(3) ***Vremenska kompozicija*** – kombinacija vremenskih i događaja koji nisu vremenski, npr. 5s nakon događaja D1.

OBRADA ECA PRAVILA (VRSTE POVEZIVANJA)

-Osnovni načini povezivanja:

(1) ***Trenutno*** – istog trenutka kada se desi događaj.

(2) ***Odloženo*** – čeka da se završi transakcija pa se onda odrađuje akcija.

(3) ***Razdvojeno*** – pri događaju otvara se nova transakcija.

-Složeniji načini povezivanja:

(1) ***Trenutan/odložen*** – evaluacija uslova se vrši odmah pri događaju ali se izvršavanje akcije odlaže.

(2) ***Razdvojen/razdvojen*** – evaluacija uslova se ne vrši odmah nego kasnije i to kao odvojena transakcija što je slučaj i sa izvršavanjem akcije.

SQL:1999 TRIGERI

-Sprecifična vrsta ECA pravila – događaj je ažuriranje, uslov je proizvoljni predikat, a akcija niz SQL naredbi. **Osnovne karakteristike**:

(1) Trigeri su objekti šeme baze podataka koji su vezani tačno za jednu tabelu i koji se pozivaju svaki put kada se obavi INSERT, UPDATE i/ili DELETE.

(2) Primenjuju se za vođenje loga (žurnala) o izvršenim operacijama nad BP.

-Klasifikacija je kompleksna, postoji više podela:

(1) ***BEFORE*** i ***AFTER*** trigeri

(2) ***INSERT***, ***UPDATE*** i ***DELETE*** trigeri

(3) ***Statement-level*** i ***row-level*** trigeri

-Primer trigera: Ograničiti povišicu plate za najviše 20%

```
CREATE TRIGGER KontrolaPovećanja
AFTER UPDATE OF Plata ON Radnik
REFERENCING
   OLD ROW AS StariRed
   NEW ROW AS NoviRed
FOR EACH ROW
WHEN (NoviRed.Plata>StariRed.Plata*1.2)
   UPDATE RADNIK
   SET Plata = StariRed.Plata*1.2
   WHERE ŠRadnika = NoviRed.ŠRadnika
```

DOBRO OFORMLJEN XML DOKUMENT I VALIDAN DOKUMENT

-Uslovi za dobro oformljen dokument – postoji deklaracija, samo jedan koreni element, svi atributi i elementi su sintaksno ispravni.

-Validan dokument – onaj koji poštuje strukturu definisanu u opisu dokumenta (šemi).

OPERACIJE SA XML DOKUMENTIMA

(1) ***Navigacione – XPATH*** – omogućava odresiranje delova i navigaciju do njih. Dokument posmatra kao stablo čvorova. Definiše koren dokumenta koji je fiktivni čvor čije je dete koreni element. Izraz putanje se zadaje preko / i //. Preko * se referenciraju sva deca nekog elementa. F-je: čvorova, tekstualne, logičke i numeričke.

(2) ***Specifikacione – XQUERY*** – upitni jezik. Pretpostavlja postojanje XML šeme. Upiti se mogu podeliti na:

a) *Upiti nad jednim dokumentom – prost neizmenjen sadržaj*. Za identifikovanje se koristi sintaksa document(uri), npr document("ex.xml")//Otpremnica[broj='10'].

b) *Upiti nad jednim dokumentom – izmenjeni sadržaj*. Definiše se preko FLWR upita (for, let, where, return), npr:

FOR $*iterator* in *xpath*
LET $*prom*:=*xpath*
WHERE *uslov*
RETURN *xml*

c) ***Upiti nad više dokumenata*** – isto FLWR, samo sa više iteratora u FOR-u. Može se koristiti i distinct-values f-ja.

TRANSAKCIJE – UPOREDNA OBRADA

-**Menadžer transakcija** – upravlja celokupnim izvršenjem transakcija. Nakon zahteva za w ili r ide prosleđuje se planer-u.
-**Planer** – vodi računa o redosledu izvršavanja akcija u skupu transakcija. Ako se narušava integritet BP može da odloži izvršavanje neke akcije. Iniciranjem akcije čitanja ili pisanja dolazi se do bafera.
-**Serijabilnost** – kada uporedno izvršavanje transakcija proizvodi isti rezultat kao i neko serijsko izvršenje istog skupa.
-Skup transakcija je izvršen **korektno** akko je taj skup serijabilan.
-**Oblik** jednog skupa transakcija $r_1(A), w_1(A)$

KONFLIKT-SERIJABILNOST

-**Konflikt** – situacija u kojoj izmena redosleda dve operacije u izvršenju dovodi do izmene efekata na bazu bar jedne od transakcija iz posmatranog izvršenja. **Nije konflikt**: $r_i(X), r_j(Y)$ ili $r_i(X), w_j(Y)$ čak i kada je X=Y, $w_i(X), r_j(Y)$, $w_i(X), w_j(Y)$.
-Dve susedne operacije različitih transakcija **mogu da zamene mesta** ako se operacije obavljaju nad istim elementom i ako je bar jedna od njih upisivanje.
-**Konflikt-ekvivalencija** – ako se jedan skup transakcija može transformisati nekonfliktnim izmenama mesta susednih operacija.
-**Konflikt-serijabilnost** – ako je izvršenje konflikt-ekv. sa nekim serijskim izvršenjem.
-**Graf predhođena transakcija** – transakcije su čvorovi a grane predhođenje transakcija. Kaže se da Ti predhodi Tj u izvršenju S ako postoje operacije Oi i Oj tako da Oi predhodi Oj u S, i Oi i Oj se odnose na isti element BP, barem jedna operacija je upisivanje.
-Npr. $S_1: r_2(A), r_1(B), w_2(A), r_3(A), w_1(B), w_3(A), r_2(B), w_2(B)$
$S_2: r_2(A), r_1(B), w_2(A), r_2(B), r_3(A), w_1(B), w_3(A), w_2(B)$. Prva je serijabilna, druga nije.

PROTOKOLI ZAKLJUČAVANJA

-Planer teško može da obavi proveru serijabilnosti i preduzima akcije u realnom vremenu. Zato se serijabilnost najčešće ostvaruje forsirano kroz **mehanizam zaključavanja** – omogućava se transakciji da postavi lokot (***XL*** – ekskluzivno, ne može niko da postavi bilo koji drugi lokot, i ***SL*** – deljeno, može druga da postavi samo SL).

-Protokol zaključavanja:

(1) Transakcija koja čita postavi SL

(2) Transakcija koja hoće da menja postavlja XL. Ako postoji SL treba da se transformiše u XL.

(3) Ako postoji nekompatibilni lokot – transakcija ide u stanje čekanja.

(4) XL se oslobađa obavezno, a po pravilu i SL na kraju, sa COMMIT i ROLLBACK

-Može doći do pojave dva XL na istom mestu. Zato se koristi **dvofazni protokol zaključavanja**:

(1) pre operisanja sa objektom mora da se postavi lokot na njega (faza širenja, a oslobađanje lokota je faza skupljanja)

(2) nakon oslobađanja lokota transakcija više ne može da postavi lokote ni na jedan objekat baze.

-Ako u nekom skupu sve transakcije poštuju dvofazni prot.zaklj. taj skup je uvek serijabilan.

VREMENSKO OZNAČAVANJE (TIMESTAMPING)

-Jedan od protokola za ostvarivanje serijabilnosti. Svakoj transakciji se dodeljuje ***univerzalni identifikator*** u redosledu kako pristižu, a sva fizička ažuriranja se odvijaju tek sa COMMIT.

-***Konflikti nastaju*** u izvršenju kada neka starija transakcija zahteva da vidi neki rekord koga je mlađa ažurirala, ili ako hoće da ažurira taj isti rekord. ***Konflikti se rešavaju*** resetovanjem stare.

-Svakom objektu se ***pridružuje dve oznake***:

(1) ***RMAX*** – najveći identifikator transakcije koja je uspešno izvršila čitanje

(2) ***UMAX*** – najveći identifikator transakcije koja je uspešno izvršila update.

-Algoritam timestamping-a:

```
Read(R)
  if t>=UMAX
    then [accept] RMAX:=MAX(t,RMAX)    else [konflikt] restart t
Write(R) [zajedno sa commit]
  if t>=UMAX and t>=RMAX
    then UMAX:= t      else restart T;
```

ŽIVI I MRTVI LOKOTI

-**Živi lokot** – neka transakcija je u stalnom čekanju na neki objekat zato što druge uvek pre nje postavljaju lokot. Rešava se jednosavno, uvođenjem nekog redosleda zaključavanja (npr. FIFO).

-**Mrtvi lokot** – kada lokoti koji su postavljeni na objektima od strane 2 transakcije dovode obe u stanje čekanja. **Načini razrešavanja**:

(1) ***Prekidanje posle isteka nekog intervala vremena*** – poništava se transakcija

(2) ***Prevencija lokota*** – uvode se protokoli; jedan se zasniva na uređenju elemenata BP, a drugi na dodeljivanju spec.vrem.oznake transakciji. Ako je transakcija koja traži lokot starija od one koji ga drži dozvoljava se čekanje, u suprotnom se prekida.

(3) ***Detekcija mrtvog čvora*** – dozvoljava da dođe do mrtvog lokota, pa se ona koja ga je izazvala ubije, ide rollback, a ona sama eventualno restartuje sa nadom da neće ponovo da dovede do lokota. Za detekciju se koristi *graf čekanja* – čvorovi su transakcije, a grana Ti – Tj postoji ako Ti zahteva neki objekat koji je Tj zaključala. Svodi se na nalaženje ciklusa.

OPORAVAK BP

-Vraćanje u korektno stanje nakon nekog otkaza. Da bi oporavak mogao da se izvrši mora SUBP da obezbedi **redundantnost podataka**. Jedan skup ovih podataka se čuva u **logu** a drugi u **arhivskoj memoriji** gde se povremeno pravi dump cele baze.

-Strategija oporavka:

(1) Ako su oštećene memorijske jedinice – koristi se arhivska kopija, za to vreme ne sme nijedna transakcija da bude aktivna.

(2) Ako nije fizički oštećena – preko loga.

-Jedan protokol oporatva se zasniva na **checkpoint**-ovima. U pojedinim predefinisanim tačkama vremena na log se upisuje rekord tačke ispitivanja koji sadrži listu trenutno izvršenih transakcija. **Postupak**:

(1) formiraju se 2 liste transakcija: UNDO i REDO

(2) pretražuje se log unapred, od checkpoint-a

(3) ako se nađe BEGIN sa T, T se dodaje u UNDO

(4) ako se nađe COMMIT za T, T se prebacuje iz UNDO u REDO

(5) koristeći vrednosti pre i posle iz loga poništavaju se sve one T u UNDO, a restartuju one u REDO.

OPORAVAK U DISTRIBUIRANIM BAZAMA PODATAKA

-Jedna transakcija može da ažurira neke podatke u bazi pod ORACLE i u bazi sa SQL serverom. Da bi se obezbedila atomnost definiše se **dvonivovski mehanizam oporavka**:

(1) Na prvom nivou ***mehanizmi oporavka lokalnih SUBP***

(2) Na drugom nivou – ***koordinator*** koordinira mehanizme sa prvog nivoa, implementira se ***dvofazni protokol potvrđivanja*** (kada svi SUPB na prvom pošalju signal da je deo transakcije kojom upravljaju završen koordinator odgovara sa "pripremi se", ovi pišu u log i šalju OK ili NOT OK, a ako istekne vreme čekanja podrazumeva se NOT OK. U drugoj fazi, ako su svi poslali OK vrši se COMMIT, u suprotnom ide ROLLBACK svih delova transakcije).