- IS ZA UPRAVLJANJE ZNANJEM -

Modeli znanja i skladištenje znanja

prof. dr Zoran Marjanović
Srđa Bjeladinović

Agenda

- Pasivni vs Aktivni SUBP
- Model znanja
  - Događaj
  - Uslov
  - Aktivnost
- Model izvršavanja
- Primeri aktivnih SUBP
- Trigeri i aktivna pravila
- Skladištenje znanja

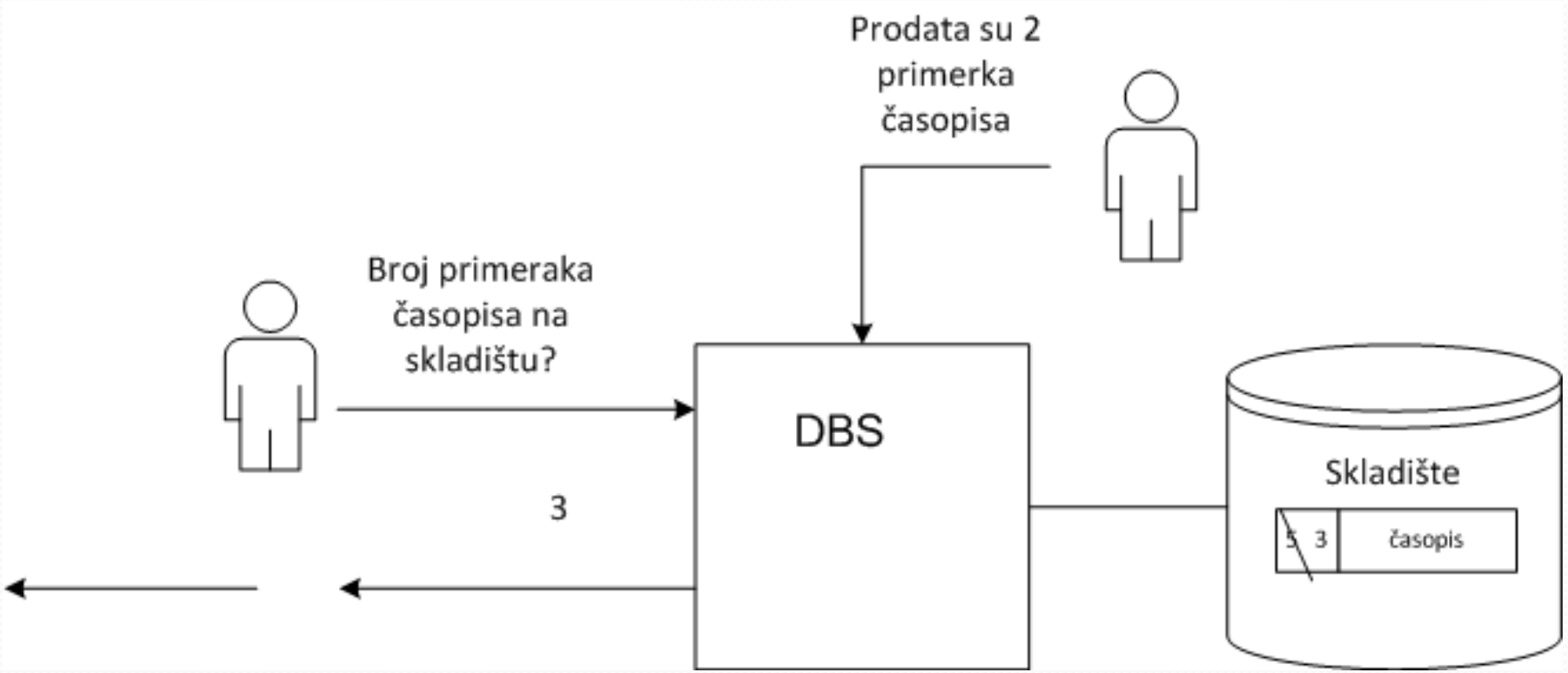
Pasivni vs Aktivni SUBP

- Konvencionalni SUBP su pasivni!
- Operacije nad podacima (CRUD) u konvencijalnom SUBP:
  - Pozivaju se kao odgovor na operacije koje dirketno pokreću korisnici ili
  - Pozivaju se u toku izvršavanja programa
- Aktivni sistemi – mogućnost automatskog izvršavanja određenih operacija kao odogovor na događaj i/ili uslov

Pasivni (vs Aktivni) SUBP - Primer

- Pojednostavljeni primer skladišnog poslovanja
- Početne pretpostavke:
  - Na skladištu se nalazi određeni broj primeraka nekog časopisa
  - Na osnovu informacija o broju prodatih časopisa smanjuje se stanje
  - Kada broj primeraka postane manji od 5, vrši se naručivanje 100 primeraka

Pasivni (vs Aktivni) SUBP - Primer



Pasivni (vs Aktivni) SUBP - Primer

- U bazi podataka čuvaju se samo podaci o trenutnom boju primeraka
- Ostala semantika u programu
- Dva tipa programa:
  - Program za ažuriranje stanja
  - Program za periodično ispitivanje stanja

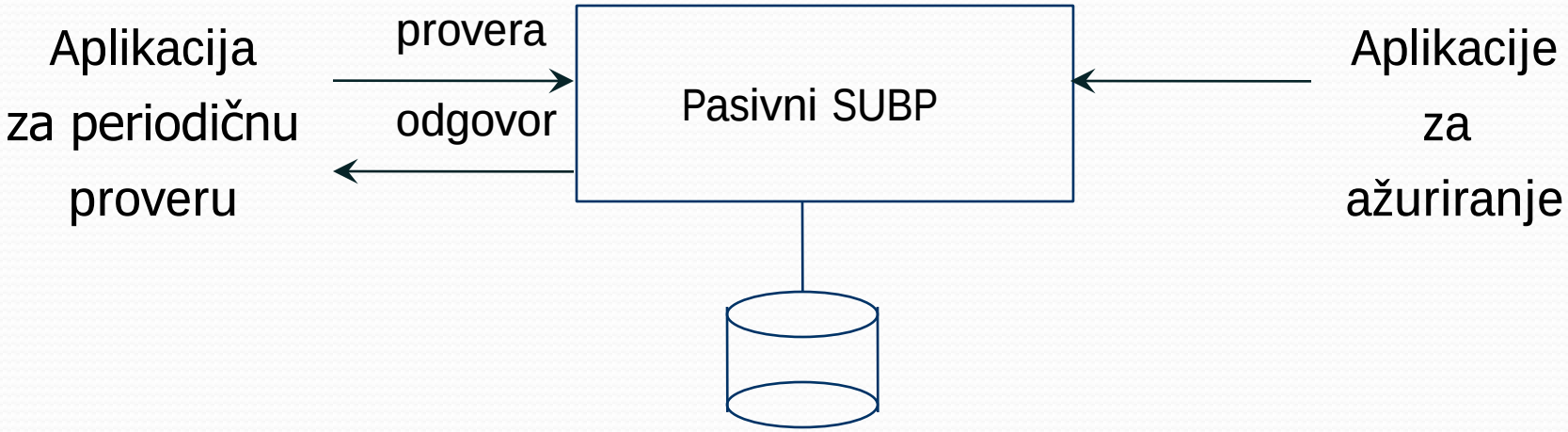
Pasivni (vs Aktivni) SUBP - Primer

- Problem razumevanja semantike i frekventnosti izvršavanja programa

- Rešenja dostupna u konvencionalnom SUBP:
  - Učestalo izvršavanje programa -> „skupo“
  - Ređe izvršavanje programa -> Kako odrediti pravi trenutak?

Pasivni (vs Aktivni) SUBP - Arhitektura

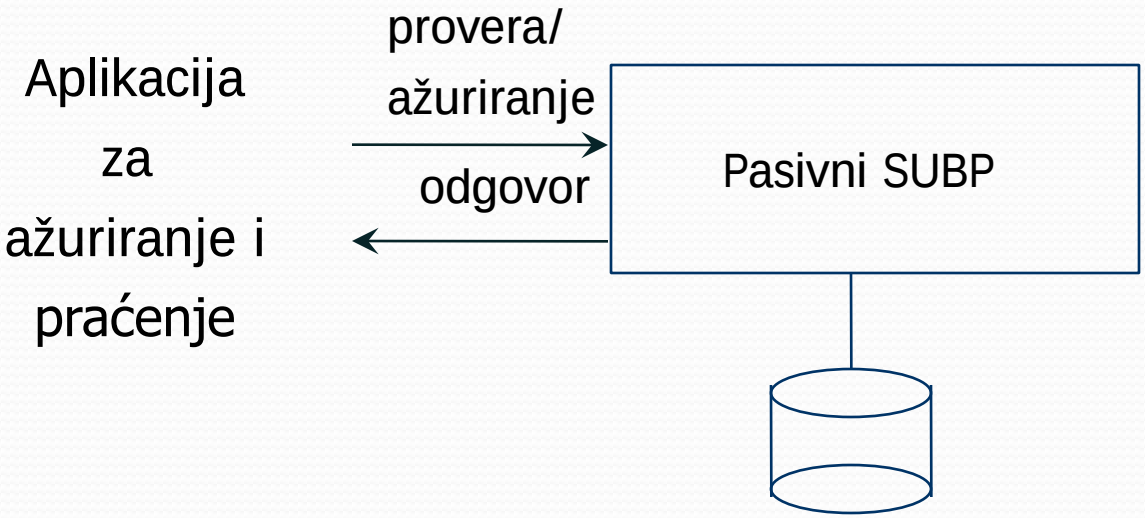
- Pristup I - Pasivni SUBP sa 2 tipa programa



- Problem: Odrediti optimalnu frekvenciju izvršavanja
  - Previše učestalo: neefikasno („skupo“)
  - Previše proređeno: potencijalni gubitak

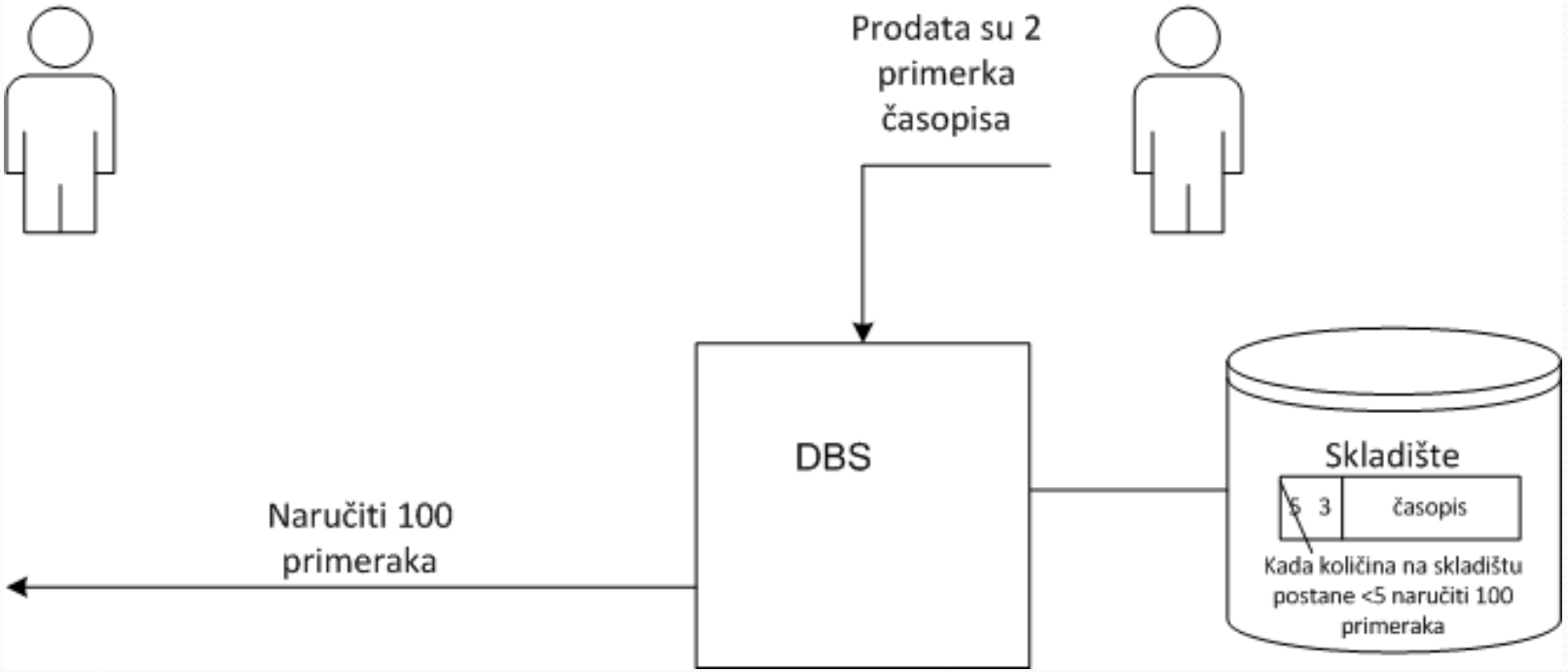
Pasivni (vs Aktivni) SUBP - Arhitektura

- Pristup II - Pasivni SUBP sa 1 tipom programa



- Problem: Odrediti optimalnu frekvenciju izvršavanja
  - Odsustvo modularnosti (promena uslova u logici reagovanja utiče na aplikaciju za praćenje)
  - Logika reagovanja izvan baze podataka

(Pasivni vs) Aktivni SUBP - Primer

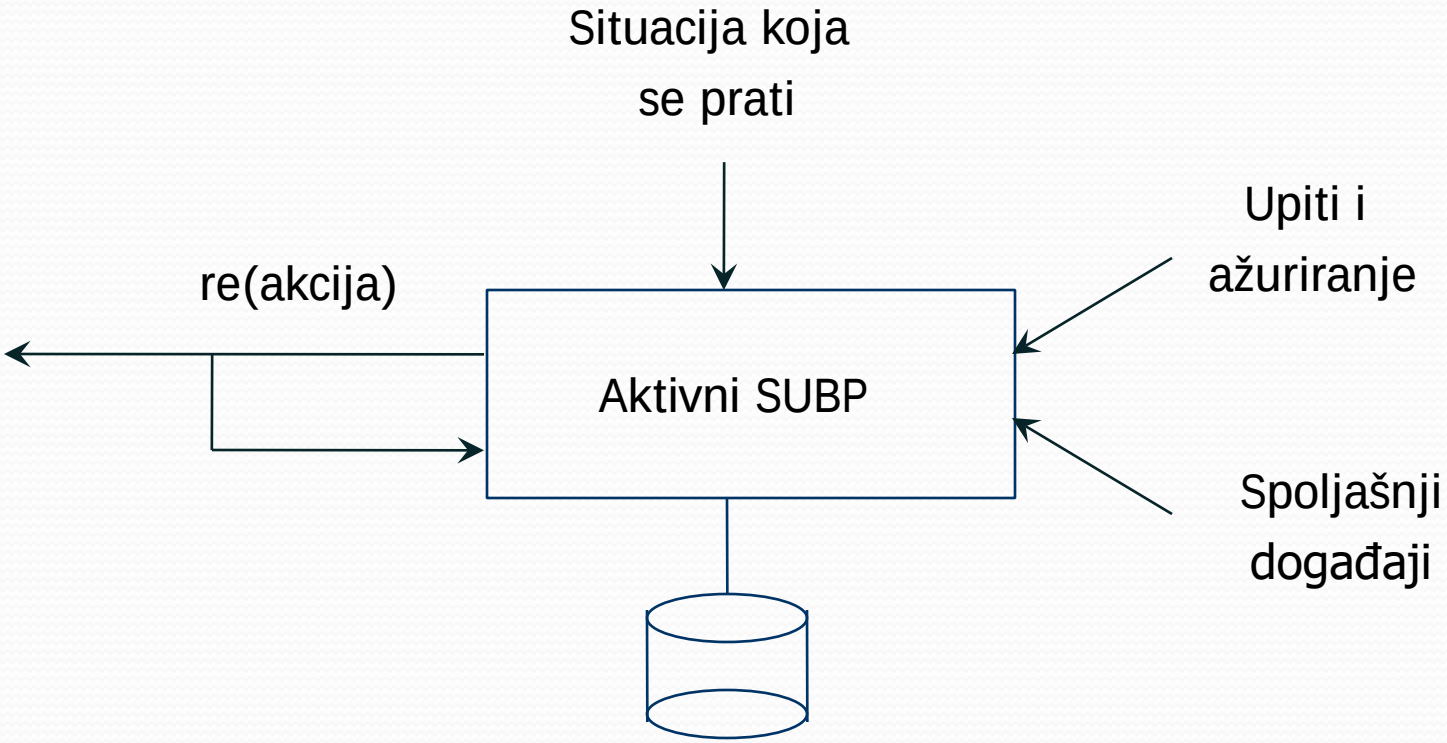


(Pasivni vs) Aktivni SUBP - Primer

- Aktivni SUBP u primeru eliminiše potrebu za programom za periodično proveravanje stanja
- Navedeni program će zameniti pravilo u jedinstvenoj bazi pravila i podataka
- Automatsko pokretanje pravila i generisanje narudžbenice

(Pasivni vs) Aktivni SUBP - Arhitektura

- Pristup III - Aktivni SUBP



Aktivni SUBP

- **Aktivni SUBP** je SUBP u kojem se odgovarajuće operacije automatski izvršavaju, kao reakcija na identifikovanu situaciju.
- Aktivni SUBP premeštaju dinamičko ponašanje iz aplikacije u sam SUBP
- Dve osnovne ideje aktivnih SUBP-ova:
  - Prepoznati **situacije** (događaje koji su nastali i/ili uslove koji su zadovoljeni) u bazi, aplikacijama i okruženju
  - Pokrenuti definisane **reakcije** (operacije nad bazom podataka ili programe)

Aktivni SUBP

- Da bi podržao reaktivno ponašanje, aktivni SUBP mora da pruži:
  - **Model znanja** (mehanizam opisivanja). Indikuje šta može biti rečeno o aktivnim pravilima sistema
  - **Model izvršavanja** (runtime strategiju). Određuje kako se pravila realizuju tokom izvršavanja
- Pored logičke šeme i činjenica, aktivni sistemi omogućuju obuhvatanje dodatne dinamike sistema iskazane kroz složena pravila integriteta
- Sva specificirana pravila čine **model znanja**

Komponente aktivnih SUBP

- Pored softvera za upravljanje bazom, aktivni SUBP se sastoji od još dve komponente:
  - Detektor događaja
  - Mehanizam za izvršavanje pravila

Event – Condition – Action (ECA)

- ECA pravila:

Event (Događaj) – Condition (Uslov) – Action (Akcija)

- Kada se događaj pojavi, ukoliko je zadovoljen uslov (događaj i uslov određuju situaciju) izvršiti akciju

ON događaj IF uslov		„situacija“
DO akcija		„(re)akcija“

Prednost korišćenja pravila

- Provera uslova je „skupa“, sa aspekta efikasnosti izvršavanja, dok je detekcija događaja „jeftinija“
- Navedeni problem je posebno izražen kod baza podataka sa velikom količinom podataka
- Mogućnost definisanja različitih akcija za različite događaje i isti uslov

Događaji

- **Događaj** – komponenta pravila koja opisuje dešavanje na koje se može odgovoriti pravilom
- Definisanje događaja prethodi formiranju pravila
- Sa aspekta trenutka izvršavanja pravila, događaji se mogu podeliti na:
  - Događaji za koje se izvršava pravilo nakon detekcije (AFTER)
  - Događaji za koje se izvršava pravilo pre detekcije(BEFORE)

Događaji

- Granularnost događaja se može definisati na nivou:
  - set-a (sve instance određene klase)
  - subset-a (određene instance klase)
  - pojedinačnih članova (members)

- Po složenosti događaja razlikuju se **primitivni** i **složeni**.

Primitivni događaji

- U primitivne događaje spadaju:
  - **Ažuriranje podataka** – U aktivnim sistemima (nadgradnja RSUBP) to su INSERT, UPDATE i DELETE. Aktivni sistemi (nadgradnja OSUBP) to su kreiranje, brisanje, izmena određenog objekta ili poziv metode za modifikaciju objekta
  - **Prikaz podataka** – U aktivnim sistemima (nadgradnja RSUBP) to je SELECT. Aktivni sistemi (nadgradnja OSUBP) to je poziv metode za prikaz objekta
  - **Vreme** – Apsolutni vremenski događaj (dostizanje određene tačke u vremenu) ili periodičan (ponavlja se periodično)
  - **Aplikativno definisan događaj** – Događaj definisan u samoj aplikaciji, za koji se definišu i pravila.

Složeni događaji

- Složeni događaji se kmbinuju od prostih i drugih složenih događaja. Operatori su:
  - **Logički operatori** – Događaji kombinovai logičkim operatorima AND, OR ili NOT
  - **Sekvenca** – Pravilo se pokreće kada se dva ili više događaja pojave u određenom redosledu
  - **Vremenska kompozicija** – Kombinovanje vremenskih i događaja koji nisu vremenski (npr. „5 sec. Nakon događaja D1“)

Događaji - razmatranja

- Događaj može biti detektovan:
  - U bazi podataka
  - U aplikaciji
  - Određen vremenskim trenutkom
- Parametrizacija događaja kod određenih aktivnih SUBP-ova
- Model znanja bi trebalo da uključe i:
  - Implicitne događaje (ALWAYS; FIRST)
  - Selektivnu aktivaciju/deaktivaciju pravila
  - Kontekstna pravila kroz grupisanje
  - Itd.

Uslovi

- **Uslov** – komponenta pravila kojom se proverava kontekst nastanka događaja (predikat nad stanjem baze, upit nad bazom ili izvršavanje aplikativne procedure)
- U najopštijem smislu, uslov unutar ECA pravila je opcioni (tzv. Event-Action pravilo). Ukoliko izostane tretira se kao da je uvek true.
- Uslov se može odnositi i na prethodno stanje atributa ili sistemske varijable („new“ i „old“ vrednosti)
- Između uslova i akcije, ponekad, se može proslediti parametar

Akcije

- **Akcija** – zadaci koje je potrebno da izvrši pravilo, ukoliko se relevantni događaj desio i ukoliko je zadovoljen usov

- Tipovi akcija:
- **Ažuriranje podataka** (INSERT, UPDATE, DELETE)
- **Pristup podacima** (SELECT)
- **Druge komande**: Definisanje podataka, upravljanje transakcijom, dodela i oduzimanje privilegija
- **Do-instead** naredba
- **Poziv aplikacijske procedure**

Model izvršavanja

- Iako specifičan za konkretni SUBP, postoje zajedničke faze za svaki od modela izvršavanja:
  - Faza signalizacije – nastanak događaja
  - Faza okidanja (trigerovanja) – za nastale događaje okidaju se odgovarajuća pravila
  - Faza evaluacije – evaluacija uslova okidnutnog pravila
  - Faza planiranja – definiše kako i kada će biti obrađeno pravilo
  - Faza izvršenja – izvršava se akcija odabranog pravila. U ovoj fazi mže se signalizirati drugim događajima da okidnu „kaskadna“ pravila

Redosled izvršavanja aktivnih pravila

- Arbitraža
- Po prioritetu
- Statistički podaci
- Dinamička pravila (npr. najnovija pravila)
- Alternativa pojadinačnom izvršavanju - Evaluacija uslova između više prikaldnih pravila i konkuretno izvršiti aktivnosti tih pravila

Prioritet pravila

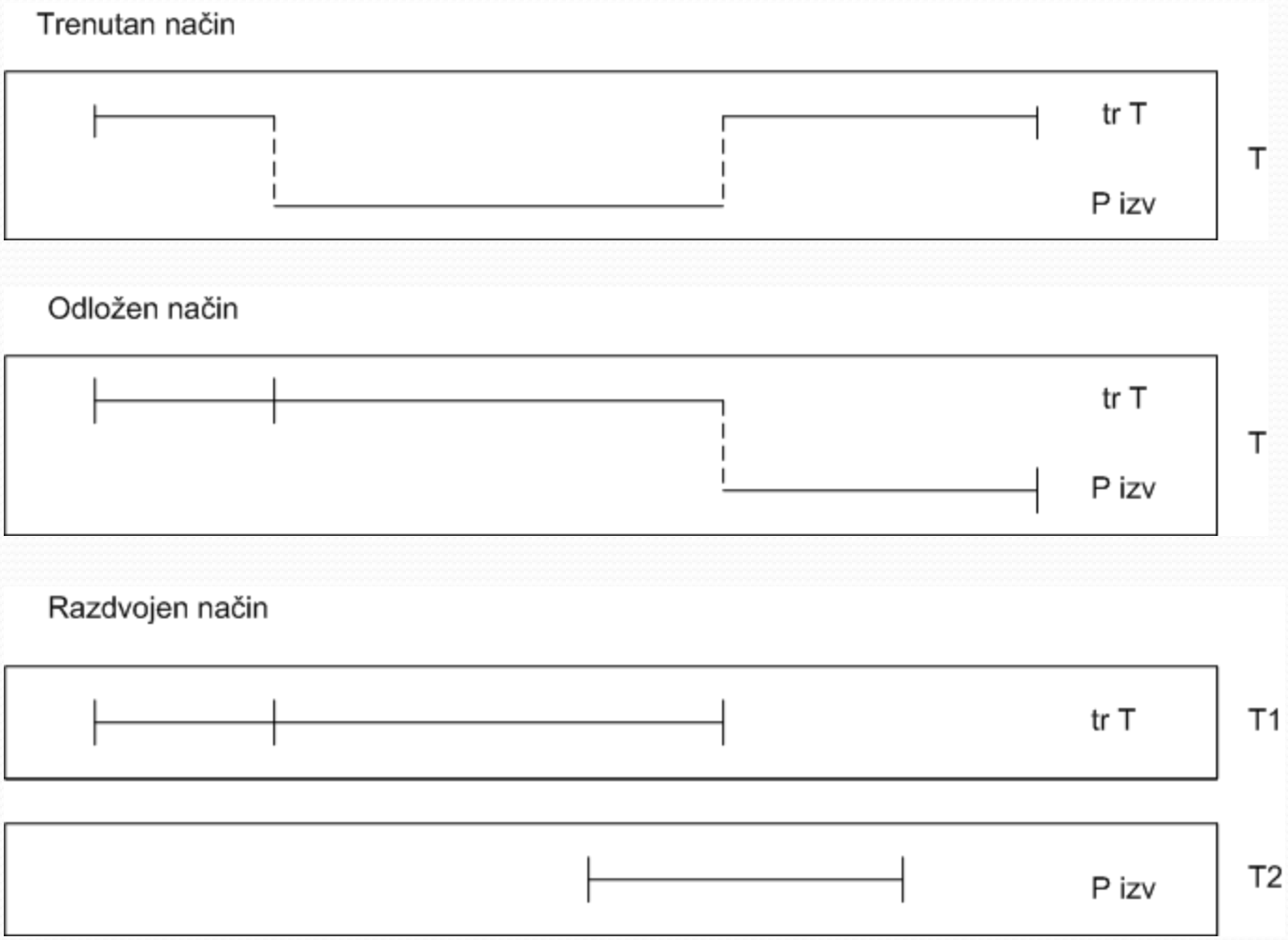
- **Prioritet pravila**: U situacijama kada je više pravila aktivirano usled istog događaja, potrebno je definisati polisu izvršavanja.

- Izbor pravila se može zasnivati na:
  - Relativnom prioritetu (prednost između para pravila). Fleksibilniji
  - Apsolutnom prioritetu (numerički prioritet). Zahteva ažuriranje prioriteta postojećih pravila sa svakim dodavanjem novog pravila

Hijerarhija izvršavanja aktivnih pravila

- Hijerarhija izvršavanja aktivnih pravila:
  - Momentalno, nakon što događaj nastane
  - Nakon svake pojedinačne operacije nad bazom podataka (npr. unos n-torke)
  - Nakon zvaršene transakcije, tj. svih naredbi definisanih unutar nje (npr. nakon eksplicitnog commit-a)
  - Aktiviranje specifikovano od strane aplikacije

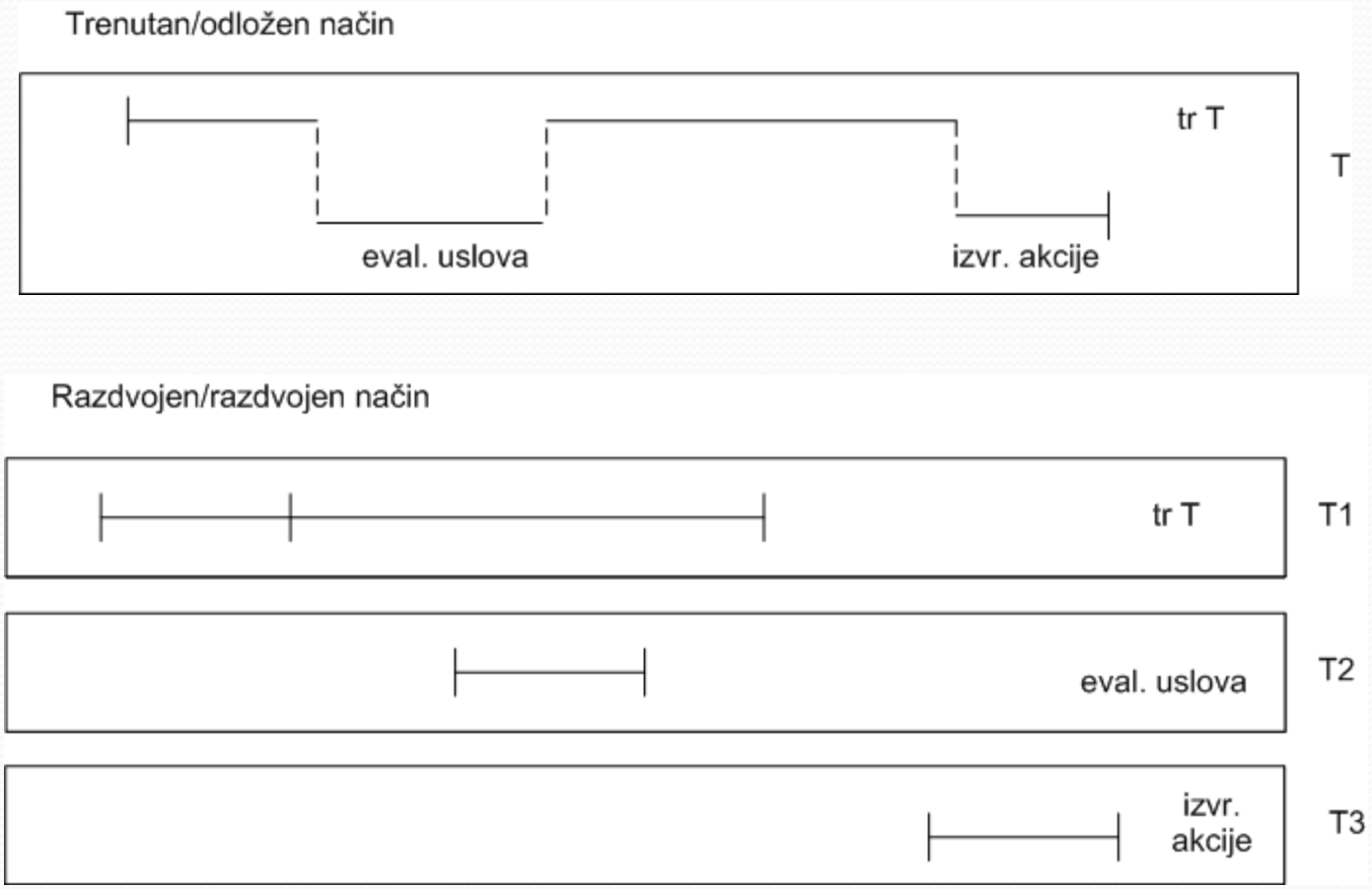
Model izvršavanja – osnovni načini



Model izvršavanja – osnovni načini

- Načini povezivanja transakcije (događaja koji pokreće pravilo i izvršavanja ECA pravila):
  - **Trenutan** (Immediate) – Prekid izvršavanja transakcije odmah po detekciji događaja sa definisanim ECA pravilom. Nastavka transakcije po završetku pravila
  - **Odložen** (Deferred) – Obrada prvila nakon završetka transakcije
  - **Razdvojen** (Decoupled) – Izvršavanje pravila je potpuno nova transakcija, nezavisna od polazne

Model izvršavanja – složeni načini



Model izvršavanja – složeni načini

- **Trenutan/odložen** - Ovim načinom vrši se evaluacija uslova prilikom nastanka događaja, dok se izvršavanje akcije odlaže nakon završetka početne transakcije

- **Razdvojen/razdvojen** – Niti se evaluacija uslova vrši trenutno, niti izvršavanje akcije. Ne dolazi do prekida početne transkacije, već se pokreću dve nove (za evaluaciju uslova i izvršavanje akcije)

Relacioni aktivni SUBP – STARBUST –

- Razvijen 1985. u IBM-ovom istraživačkom centru sa ciljem pružanja mogućnosti proširenja sistema, odnosno uključivanja novih tehnologija u okviru komponenti sistema
- Sastoji se iz dve glavne komponente: procesor upitnog jezika i jezgro upravljanja podacima
- Sintaksna pravila su zasnovana na SQL-u, skupovno orijentisana –pokreću se skupovima promena nad bazom podataka
- Priroda promena prilikom izvršavanja operacija prati se i čuva u tranzicionim tabelama

Relacioni aktivni SUBP – STARBUST –

- Pravila su zasnvana na pojmu prelaza stanja – uzimajući u obzir „neto“ efekte prelaza stanja

- Sintaksa naredbe za kreiranje pravila:

create rule <naziv_pravila> **on** <tabela>
when <predikat_prelaza>
[if <uslov>**]**
then <lista_akcija>
[precedes <lista_pravila>**]**
[follows <lista_pravila>**]**

Relacioni aktivni SUBP – STARBUST –

- Nakon pokretanja pravila proverava se uslov, ukoliko je zadovoljen (true) izvršavaju se akcije

- Akcije, odnosno Starburst operacije nad bazom podataka:

  - manipulacija podacima: insert, delete, update, select
  - kontrolne funkcije: rollback, commit
  - definisanje podataka: create

Relacioni aktivni SUBP – POSTGRESQL–

- Objektno relaciono orjentisan sistem za upravljanje bazom podataka (*eng. Object - Relational Database Management System – ORDBMS*)
- Open source, nije u vlasništvu nijedne kompanije
- Dostupan na mnogim platformama uključujući Linux-u, FreeBSD-u, Solaris-u, MC Windows-u i Mac OS.
- Podržava veći deo SQL standarda
- Nudi sve uobičajene funkcionalnosti relacionih baza podataka uz nekoliko jedinstvenih karakteristika - mogu se dodavati sopstveni tipovi podataka u PostgreSQL.

Relacioni aktivni SUBP – POSTGRESQL–

- PostgreSQL funkcionalnosti uključuju sledeće:
  - Transakcije
  - Podizbore
  - Poglede
  - Referencijalni integritet spoljnog ključa
  - Sofisticirano zaključavanje
  - Korisnički definisane tipove
  - Nasleđivanje
  - Pravila
  - Više-verzijsku kontrolu konkurentnosti

- Verzijom 8.0 uvedene su neke nove funkcionalnosti:
  - Podrazumevana Microsoft Windows verzija
  - Tablične prostore
  - Sposobnost da se menjaju tipovi kolona
  - Oporavak do tačke u vremenu

Relacioni aktivni SUBP – POSTGRESQL–

- PRS I sistem pravila zasnovan na Postgrel-u: naredba se izvršava ili uvek (ALWAYS) ili nikad (REFUSE) ili tačno jedanput (ONE-TIME), događaji si implicitni

- PRS II sistem pravila nastao da dopuni nedostatke PRS I sistema pravila, u okviru kojih su događaju eksplicitni.

- PRS II ima sledeću sintaksu:

define [tuple | rewrite] rule <naziv>
on <dogadjaj> **to** <objekat>
[**where** <uslov>]
do [instead] <lista akcija>

- Događaji predstavljaju standardne Postgrel naredbe za manipulaciju podacima: append, delete, replace/retrieve, new, old....

Relacioni aktivni SUBP – POSTGRESQL–

- Pravila se mogu implementirati na dva načina:
  - trigger implementacija (tuple-level) – obrađuje se triger za svaku pojedinačnu n-torku relacije
  - query rewrite implementacija – upiti se kombinuju sa skupom pravila generišući modifikovane upite

- Pravila se definišu izborom ključne reči tuple ili rewrite

Objektni aktivni SUBP – HiPAC –

- HIPAC (High Performance Active Database System)
- Najpoznatiji aktivni sistem, zasnovan na objektno orijentisanom modelu podataka
- Generalizovan jezik pravila, okrenut ka prevazilaženju vremenskih ograničenja prilikom upravljanja podacima
- Pravila i njihove komponente su objekti baze podataka pa se ponašaju kao drugi objekti, mogu se grupisati ili imati atribute i biti povezani sa drugim objektima
- Svako pravilo ima definisan događaj (Event), uslov (Condition), akciju (Action) i način povezivanja (Coupling Mode)

Objektni aktivni SUBP – HiPAC –

- Događaj predstavlja operaciju nada bazom podataka, vremensku karakteristiku ili njihovu kombinaciju
- Uslov predstavlja skup upita nad bazom podataka
- Akcija je sekvenca operacija nad bazom podataka
- Parametri povezuju događaj, uslov i akciju
- Način povezivanja može biti: trenutan, odložen i razdvojen (razdvojen način povezivanja može biti uzročno-povezan razdvojen i nepovezan razdvjen
- Parametri iz događaja prenose se u uslov, a parametri iz događaja i uslova se prenose u akciju

Objektni aktivni SUBP – HiPAC –

- HiPAC klasa Rule je podklasa klase Entity
- Atributi klase Rule: Event, Condition, Action, ostali atributi
- Operacije klase Rule: Create, Delete, Modify, Enable, Disable, Fire
- Izvršavanje pravila može biti:
  - konkurentno izvršavanje pravila: jedan događaj može da pokrene više pravila pa ne dolazi do rešavanja konflikta;
  - rekurzivno - kao rezultat izvršenja jednog pravila, može biti pokrenuto isto ili drug pravilo
- Fleksibilnost se ostvaruje izborom načina povezivanja

SQL:1999 Trigeri

- Triger se može definisati kao proceduralni kod koji se automatski izvršava svaki put kada se desi definisani događaj nad određenom tabelom ili pogledom.
- Trigeri su specifična vrsta ECA (Event – condition – action) pravila
- Događaj je operacija ažuriranja baze podataka, uslov je proizvoljni SQL predikat, a akcija je sekvenca SQL naredbi

SQL:1999 Trigeri

- Pod ažuriranjem se podrazumevaju DML operacije (INSERT, UPDATE ili DELETE)
- Neki sistemi takođe podržavaju i drugu vrstu trigera koji se okidaju kao posledica izvršenja DDL naredbi ili kontrolnih naredbi (npr. commit i rollback transakcije)
- Ovakvi DDL trigeri se mogu koristiti u bazi podataka za potrebe revizije
- SQL:1999 ne podržava definisanje trigera nad SELECT naredbom

Namena trigera

- Trigeri se pre svega koriste za očuvanje integriteta baze podataka

- Mogu implementirati pravila integriteta, koja se dele na dve grupe:
  - pravila integriteta modela
  - poslovna pravila integriteta

Izvršenje trigera

- Vreme izvršenja trigera (Trigger timing) u odnosu na posmatrani događaj:
  - BEFORE
  - AFTER
  - INSTEAD OF (ORACLE)
- Koliko puta se telo trigera izvršava
  - STATEMENT – trigeri koji se pokreću na nivou naredbe
  - ROW – trigeri koji se pokreću na nivou reda
- Primer: Suma stavki

Specifikacija trigera

CREATE TRIGGER <naziv trigera>
{BEFORE | AFTER|INSTEAD OF}
{INSERT | UPDATE | DELETE [OF <naziv kolone1>, <naziv kolone2>,…]}
ON <naziv tabele>
[REFERENCING {OLD [ROW] [AS] <sinonim za red pre ažuriranja>
| NEW [ROW] [AS] <sinonim za red posle ažuriranja>
| OLD TABLE [AS] <sinonim za tabelu pre ažuriranja>
| NEW TABLE [AS] <sinonim za tabelu posle ažuriranja>},…]
[FOR EACH {ROW | STATEMENT}]
[WHEN (<uslov>)]
{<SQL naredba>
| BEGIN
{<SQL naredba>;}…
END}

Primer 1 (Oracle)– Identifikovati efekat

```
CREATE OR REPLACE TRIGGER TRG_BD_StavkaPrijemnice
BEFORE DELETE ON StavkaPrijemnice
FOR EACH STATEMENT
BEGIN
    RAISE_APPLICATION_ERROR (-20000, 'Zabranjeno brisanje')
END;
```

Primer 1 – Efekat

- Napisati triger kojim se zabranjuje je bilo kakvo brisanje iz tabele stvaka prijemnice.

Primer 2 (Oracle) – Identifikovati efekat

```
CREATE OR REPLACE TRIGGER TRG_AIU_StanjeZaliha
AFTER INSERT OR UPDATE ON StanjeZaliha
REFRENCING OLD TABLE AS DELETED
REFRENCING NEW TABLE AS INSERTED
FOR EACH STATEMENT
DECLARE L_BR NUMBER(4);
BEGIN
    SELECT COUNT(*) INTO L_BR
    FROM INSERTED
    WHERE KolStanje < 0;
     IF L_BR > 0 THEN
        RAISE_APPLICATION_ERROR (-20001, 'Negativno stanje zaliha')
   END IF;
END;
```

Primer 2 – Efekat

- Napisati triger kojim se pri ažuriranju stanja zaliha proverava da li je količina proizvoda na zalihama posle promena manja od nule.

Primer 3 (Oracle) – Identifikovati efekat

```
CREATE OR REPLACE TRIGGER radno_vreme
BEFORE INSERT OR UPDATE OR DELETE ON STANJENAZALIHAMA
BEGIN
    IF (TO_CHAR(SYSDATE, 'HH24') NOT BETWEEN '08' AND '18')
    THEN
            IF DELETING THEN
                    RAISE_APPLICATION_ERROR (-20500, 'Mozete da obrisete slog samo tokom trajanja radnog
    vremena.');
            ELSIF INSERTING THEN
                    RAISE_APPLICATION_ERROR (-20502, 'Mozete da unesete slog samo tokom trajanja radnog
    vremena.');
            ELSIF UPDATING ('KOLSTANJE') THEN
                    RAISE_APPLICATION_ERROR (-20503, 'Mozete da izmenite kolicinu samo tokom trajanja
    radnog vremena.');
            ELSE
                    RAISE_APPLICATION_ERROR (-20504, 'Mozete da izmenite stanje na zalihama samo tokom
    trajanja radnog vremena.');
            END IF;
    END IF;
END;
```

Primer 3 – Efekat

- Napisati triger kojim se dozvoljava rad nad tabelom StanjeNaZalihama samo u radno vreme. Ukoliko radnik pokuša da radi nad tabelom pre 8 časova, ili nakon 18 treba prikazati adekvatnu poruku o grešci. Napomena: Za operaciju ažuriranja je karakteristično da ovo ograničenje važi samo za količinu.

Trigeri u MS SQLServer-u

- T-SQL
- Za razliku od Oracle SUBP-a, INSTEAD OF trigeri se mogu definisati i nad tabelama, a ne samo nad pogledima
- Primer after update tigera:

```
CREATE TRIGGER [dbo].[ODLUKAJN] ON   [dbo].[ODLUKAOPOKRETANJUPOSTUPKA]
AFTER UPDATE AS
IF UPDATE (ARHIVSKIBROJ)
BEGIN
ALTER TABLE [dbo].[JAVNANABAVKA] DISABLE TRIGGER [NEMENJAJARHIVSKIBROJ]
UPDATE [DBO].[JAVNANABAVKA]
SET [ARHIVSKIBROJ] = (SELECT [ARHIVSKIBROJ]
                      FROM INSERTED)
WHERE IDODLUKEOPOKRETANJU = (SELECT [IDODLUKE]
                             FROM INSERTED)
ALTER TABLE [dbo].[JAVNANABAVKA] ENABLE TRIGGER [NEMENJAJARHIVSKIBROJ]
END;
```

Trigeri u MS SQLServer-u

- Triger nad pogledom (INSTEAD OF):

```
CREATE TRIGGER POGLEDTRIGER ON SVE_O_PONUDJACU
INSTEAD OF INSERT
AS
BEGIN
SET NOCOUNT ON
IF (NOT EXISTS (SELECT P.IDPONUDJACA
      FROM PONUDJAC P, INSERTED I
      WHERE P.IDPONUDJACA = I.IDPONUDJACA))
   INSERT INTO PONUDJAC
      SELECT IDPONUDJACA, NAZIVPONUDJACA
      FROM INSERTED

IF (NOT EXISTS (SELECT E.IDPONUDJACA
      FROM PONUDJACDETALJI E, INSERTED
      WHERE E.IDPONUDJACA = INSERTED.IDPONUDJACA))
   INSERT INTO PONUDJACDETALJI
      SELECT IDPONUDJACA, DELATNOST, KONTAKTTELEFON
      FROM INSERTED
ELSE
   UPDATE PONUDJACDETALJI
      SET IDPONUDJACA = I.IDPONUDJACA,
          DELATNOST = I.DELATNOST,
          KONTAKTTELEFON = I.KONTAKTTELEFON
   FROM PONUDJACDETALJI E, INSERTED I
   WHERE E.IDPONUDJACA = I.IDPONUDJACA
END
```

Trigeri u PostgreSQL-u

- Kao i kod MS SQLServer-a, a za razliku od Oracle SUBP-a, INSTEAD OF trigeri se mogu definisati i nad tabelama, a ne samo nad pogledima
- Mogućnost definisanja BEFORE i AFTER trigera nad pogledom, ali isključivo na nivou naredbe
- Pored standardnih operacija ažuriranja, pokretanje trigera može izazvati i TRUNCATE naredba, koja mora biti definisan na nivou naredbe

Trigeri u PostgreSQL-u

Kada	Događaj	Row-level	Statement-level
BEFORE	INSERT/UPDATE/DELETE	Tabele	Tabele i pogledi
	TRUNCATE	-	Tabele
AFTER	INSERT/UPDATE/DELETE	Tabele	Tabele i pogledi
	TRUNCATE	-	Tabele
INSTEAD OF	INSERT/UPDATE/DELETE	Tabele i pogledi	-
	TRUNCATE	-	-

Podaci, informacije i znanja

- **Podatak** – diskretna, objektivna činjenica o nekom konceptu realnog sveta. Neophodna za kreiranje informacija. Ne poseduje značenje, internu strukturu, niti veze između entiteta.

- **Informacija** – Podatak kojem je dodato značenje, definisanjem konteksta i/ili forme. Kontekst dodaje namenu i okolnosti prikupljanja podataka. Može dati odgovor na pitanja „ko“, „šta“, „gde“, „kada“ i slično.

Podaci, informacije i znanja

- Informacija se može posmatrati i kao poruka kojoj primalac poruke određuje relevantnost u informisanju.

- Karakteristike „korisnih“ informacija:
  - Istinite
  - Dostupne na vreme
  - Korisne
  - Potpune
  - Precizne

- Iako neophodne za donošenje odluka, informacije nisu dovoljne

Podaci, informacije i znanja

- **Znanje** – Poznavanje pravila neophodnih za interpretaciju informacija. Skup podataka, informacija i veština skupljenih iskustvom i/ili tokom obrazovanja.

- **Mudrost**– Kombinacija znanja i iskustva koja omogućava donošenje odluka i odabir adekvatnih veština za izvršenje.

Ekspertni sistemi

- „Ekspertni sistem je sistem koji angažuje ljudsko znanje prikupljeno u kompjuteru radi rešavanja problema koji zahtevaju ljudskog eksperta“ (Turban)

- „Kompjuterski program koji imitira ponašanje ljudskih eksperata, koji rešavaju realne probleme povezane sa određenim domenom“(Pigford & Braur)

Skladištenje znanja

- Aktivna pravila omogućavaju znanju da interpretiraju informacije i da na osnovu istih donose odluke.

- Priprema za skladištenje znanja u SUBP-u realizuje se „hvatanjem“ semantike objektnih sistema i primenom korisnički definisanih i konstruisanih tipova u samoj bazi.

Korisnički definisani tipovi

- Korisnički definisani tipovi su tipovi koje definiše korisnik. Mogu biti **distinct** ili **struktuirani**.
- Metodama se definišu operacije nad korisničkim tipovima
- Prednosti
  - Pojednostavljuje se razvoj i održavanje aplikacija
  - Mogućnost korišćenja korisnički definisanih tipova od strane više aplikacija
  - **Davanje konteksta tipu podatka** (za razliku od predefinisanih) i mogućnost kreiranja hijerarhije tipova.

Distinkt tip

- Distinkt tip je jednostavan, perzistentni, imenovani korisnički definisani tip, čijim uvođenjem je podržano strogo tipiziranje.

- Distinkt tip je preimenovani predefinisani SQL tip sa drugačijim ponašanjem u odnosu na njega.

Distinkt tip

- Sintaksa distinkt tipa:

CREATE TYPE <naziv distinkt tipa>

AS <predefinisani tip> FINAL

[<cast opcije prevođenja>]

- FINAL – označava da ne mogu imati podtipove (nije podržano nasleđivanje)
- Opcije prevođenja omogućavaju konverziju iz distinkt tipa i obrnuto
- Distinkt i predefinisani tipovi nisu direktno uporedivi

Distinkt tip

- CREATE TYPE metar AS INTEGER FINAL;
  CREATE TYPE kvadratni_metar AS INTEGER FINAL;

- CREATE TABLE Soba(SobaID integer,
  Duzina metar,
  Sirina metar,
  Povrsina kvadratni_metar);

- UPDATE Soba
  SET Povrsina = Duzina; //CAST (Povrsina AS INTEGER) i CAST (Duzina AS INTEGER)
- Nije dozvoljeno direktno poredjenje

Struktuirani tip

- Struktuirani tip omogućava definisanje perzistentnih, imenovanih, tipova sa jednim ili više atributa

- Pored strukture, definisane atributima, struktuirani tipovi imaju i ponašanje, definisano metodama

Struktuirani tip

- Sintaksa struktuiranog tipa:

CREATE TYPE <naziv struktuiranog tipa>
[UNDER <naziv nadtipa>]
AS (<naziv atributa><tip atributa>, ...)
[[NOT] INSTANTIABLE]
NOT FINAL
[<specifikacija metode>, ...]

- UNDER – podtip drugog struktuiranog tipa
- NOT INSTANTIABLE – struktuirani tip ne mora imati svoje instance
- NOT FINAL- nije konačan i može da ima podtipove

Objektni tip

- Struktuirani i distinct tip iz SQL:1999 standarda se u Oracle-u implementiraju preko Object tipa
- Object tip sa samo jednim atributom odgovara distinct tipu po SQL:1999 standardu
- Object tip sa više atributa odgovara struktuiranom tipu po SQL:1999 standardu

Objektni tip

- Object tip je šema sa tri tipa komponenti:
  - Ime :
    - Jedinstveno u šemi
  - Atributi :
    - Predefinisani ili drugi korisnički definisani tipovi
    - Atributima se definiše struktura object tipa
  - Metode :
    - Funkcije ili procedure pisane u PL/SQL-u i skladištene u bazi podatka ili napisane u programskom jeziku i skladištene eksterno
    - Metode implementiraju operacije koje aplikacija može izvršiti nad object tipom

Objektni tip

Opšta sintaksa za kreiranje Object type-a:

```
CREATE OR REPLACE naziv_tipa AS OBJECT
(lista_atributa, lista_metoda);
```

Opšta sintaksa za kreiranje Body-a Object tipa:

```
CREATE OR REPLACE TYPE BODY naziv_tipa
  AS
Definicija_metoda;
```

Objektni tip - Primer

- Primer kreiranja Object tipa: Adresa sa atributima ulicom i brojem

```
CREATE OR REPLACE TYPE obj_adresa_br AS OBJECT
(ulica VARCHAR2(50),
 broj  NUMBER,
 MEMBER FUNCTION get_ulica RETURN VARCHAR2,
 MEMBER FUNCTION get_broj RETURN NUMBER)
 instantiable not final;
```

Objektni tip - Primer

- Primer kreiranja Body-a za Object tipa Adresa

```
CREATE OR REPLACE TYPE BODY obj_adresa_br AS
  MEMBER FUNCTION get_ulica RETURN VARCHAR2 IS
      BEGIN
            RETURN self.ulica;
      END;
  MEMBER FUNCTION get_broj RETURN number IS
      BEGIN
            RETURN self.broj;
      END;
END;
```

Objektni tip - Primer

- Naredba za kreiranje tabele adresa, koja će sadržati kolonu sa definisanim tipom je:

```
CREATE TABLE adresa (
    sifra_adr NUMBER NOT NULL,
    adresa OBJ_ADRESA_BR,
    CONSTRAINT ADRESA_PK PRIMARY KEY (sifra_adr)
);
```

- Naredba za unos podataka u tabelu adresa, koja sadrži korisnički definisani tip je:

```
INSERT INTO adresa VALUES (
    1,obj_adresa_br('Jove Ilica',154)
);
```

Objektni tip - Primer

- Naredba za prikaz podataka iz tabele adresa, koja sadrži korisnički definisani tip je:

```
SELECT a.sifra_adr "Sifra adrese",
      a.adresa.get_ulica() "Ime ulice",
      a.adresa.get_broj() "Broj"
FROM adresa a;
//
.....
      a.adresa.ulica "Ime ulice",
      a.adresa.broj  "Broj“
.....
```

Konstruisani tipovi

- Po SQL:1999 standardu, konstruisani tipovi su referentni tipovi (reference), tipovi vrste i kolekcije.
- Na kraju će biti prikazan i nested table tip.
- Vodeći proizvođači SUBP-ova , poput Oracle-a, podržavaju navedene tipove.

Referentni tip

- Tabele definisane direktno nad struktuiranim tipovima mogu imati referentnu kolonu, koja služi kao identifikator n-torki.

- Sintaksa REF tipa:

ATRIBUT REF(TIP) SCOPE RELACIJA

- TIP - neki struktuirani tip
- REF(TIP) - tip reference na n-torku tipa TIP

- SCOPE - naziv relacije čije se n-torke referenciraju (u primeru je to RELACIJA)

Tip vrsta

- Ovaj tip se definiše kao par (<naziv podatka>, <tip podatka>).

- SQL:1999 uvedeo mogućnost definisanja promenljivih i parametara koji su tipa vrsta, odnosno definisanje kolone u tabeli koja će imati kompleksnu strukturu.

- Primer kreiranja raw tipa:

```
create table kolekcija_boja (boja raw(16));
```

- Primer za unos vrednosti u tabelu koja sadrži raw tip:

```
insert into kolekcija_boja values ('FFoooo');
```

- Prikazaivanje vrednosti iz tabele sa raw tipom podataka:

```
select * from kolekcija_boja where utl_raw.substr(boja, 1, 2) = 'FF';
```

Varrays

- Array predstavlja numerisani niz čiji elementi su podaci.
- Svi elementi unutar array-a su istog tipa i imaju jedinstven indeks u nizu. Maskimalna veličina se mora odrediti prilikom kreiranja arrays-a.
- Oracle dozvoljava upotrebu array-a promenljive dužine, pa je varrays skraćenica od variable arrays.
- Sintaksa za kreiranje varrays:

CREATE OR REPLACE TYPE name-of-type IS VARRAY(nn)of type

- Konkretan primer kreiranja varrays-a phones je dat u nastavku:

CREATE TYPE telefon AS VARRAY(10) OF varchar2(10);

Varrays

- Sledećom naredbom kreirani tip se koristi u tabeli:

```
create table DOBAVLJAC(DID number(5),
                       Naziv varchar2(20),
                       tel telefon);
```

- Naredba za unos podataka u tabelu sa varrays tipom:

```
insert into dobavljac values (101,'Dobavljac abc',
Telefon('1234567','7654321'));
```

- Naredba za prikaz podataka tabele sa varrays tipom:

```
Select * from DOBAVLJAC;
```

Nested tables

- Nested table je tip koji podržava viševrednosne atribute, tj. podržava kolone koje mogu sadržati kompletne druge tabele.

- Ovj vid kolekcije narušava prvu normalnu formu

Nested tables

- Primer za kreiranje ovog korisničkog tipa(tabela Odeljenje koristi ListaKurseva):

- CREATE OR REPLACE TYPE ListaKurseva AS TABLE OF VARCHAR2(64);

- CREATE TABLE Odeljenje(
  naziv VARCHAR2(20),
  direktor VARCHAR2(20),
  lokacija VARCHAR2(20),
  dostupni_kursevi ListaKurseva)
  NESTED TABLE dostupni_kursevi STORE AS kursevi_tab;