KOMPARATIVNA ANALIZA KORISNIČKI DEFINISANIH TIPOVA PODATAKA U SISTEMIMA ZA UPRAVLJANJE RELACIONIM BAZAMA PODATAKA

COMPARATIVE ANALYSES OF USER-DEFINED DATA TYPES IN THE RELATION DATABASE MANAGEMENT SYSTEMS

Srđan Grbavac, *Fakultet Organizacionih nauka, Univerzitet u Beogradu, Srbija*

Sadržaj - *Sistemi za upravljanje relacionim bazama podataka (RDBMS) omogućuju korištenje većeg broja sistemskih tipova podataka, pohranjuju datume, brojevne podatke, tekstualne podatke itd., međutim, u nekim slučajevima postoji potreba za prilagođene tipove podataka radi konzistencije u aplikaciji, a samim tim pojednostavljuje se razvoj i održavanje informacionog sistema. Rad pokazuje karakteristike, sličnosti i razlike u mogućnostima UDT-a , u 3 najrasprostranjenija RDBMS – Microsoft Sql Server 2008, ORACLE 11g i IBM DB2, kao i pozitivne i negativne strane upotrebe UDT-a. Rad obuhvata i osvrt na ISO/IEC 9075 standard, kao i standardom propisane minimalne zahtjeve koje jezici za manipulisanje podacima u RDBMS-u moraju obezbijediti za rad sa UDT.*

Ključne riječi: *Relacione baze podataka, tipovi podataka, kvalitet softvera, informacioni sistemi*

Abstract - *The relation database management systems (RDBMS) provides using of a large number system data types,store dates, numeric data, text data ... however, in some cases there is a need for custom data types for consistency in application, therefore simplifiest the development and maintenance of information systems. The research shows the charasteristics, similarities and differences in possibilities UDT, in three most common RDBMS – Microsoft Sql Server 2008, ORACLE 11g and IBM DB2, and the positive and negative sides of using UDT. Research includes review of ISO/IEC 9075 standard as well as minimum requirements which language for data manipulation in RDBMS has provide for UDT.*

Key words: *Relation databases, data types, software quality, information systems*

1. UVOD

Objektno orijentisane tehnologije olakšavaju programerima da dizajniraju objekte podataka koji podržavaju strukturu i ponašanje entiteta u realnom svijetu, poput multimedijalnih fajlova, XML dokumenata, valuta i geoprostornih aplikacija. Funkcionalnost i podaci mogu biti enkapsulirani sa objektom čineći ga diskretnim, samostalnim entitetom, iz kojeg se može doći do informacija o sadržaju i koji podržava set metoda koje se izvršavaju na tom sadržaju.
Istorijski gledano, projektanti baza podataka su u dizajniranju i implementaciji baze podataka morali na teži način postići simulaciju procesa i "real world" objekata upotrebom uobičajenih struktura i tipova podataka dostupnih u relacionim bazama podataka. Npr., jedan od načina za predstavljanje MPEG video zapisa je bio, da se kreira tabela sa kolonom gdje se čuvaju binarni podaci i drugom kolonom koja je sadržavala vrijednosti koje predstavljaju različite atribute videa. Ovaj način predstavljanja je donekle linearan i zahtjeva zasebne kolone za binarne video podatke i za svaki atribut. Više koncizan pristup bi bio da se video zapis direktno predstavi kao objekat (slično klasi u C++) sa operacijama i atributima koji usko određuju kako će se video snimak prikazivati. Baze podataka sada podržavaju objektno orijentisani pristup podržavajući korisnički definisane tipove podataka (UDT).

2.OSVRT NA SQL STANDARD ISO 9075-2:1999

UDT je šema objekat, identifikovan imenom UDT-a. Definicija UDT precizira broj komponenti ,uključujući posebnu listu definicija atributa. Tip UDT može iskazivati jedan tip podatka, u tom slučaju se kaže da je UDT poseban tip podatka, ili da iskazuje listu atributa, i u tom slučaju je struktuirani tip.

Definicija UDT-a može uključivati specifikaciju liste metoda koja se sastoji od jedne ili više specifikacija metoda. Specifikacija metoda može biti specifikacija originalnih metoda i specifikacija preovlađujućih metoda.
Svaka specifikacija originalnih metoda specificira naziv metode, specifično ime, deklaracionu listu SQL parametara, povratnog tipa podatka, rezultat konvertovanja iz nekog tipa (ukoliko postoji), specifikaciju transformacione grupe (ukoliko postoji).

Najmanje jedna funkcija konstruktora je povezana sa struktuiranim tipom i implicitno se definiše kada se struktuirani tip definiše.[1]

3. MICROSOFT SQL SERVER

3.1 Uvod u korisnički definisane tipove podataka u ms SQL Server - u

SQL SERVER obezbijeđuje mogućnost upotrebe UDT, odnosno upotrebu prilagođenih tipova podataka u tabelama, trigerima, pogledima, funkcijama, uskladištenim procedurama itd., što obezbijeđuje konzistentnost podataka u ms SQL Server bazama podataka.
Ms SQL Sever daje mogućnost kreiranja dvije vrste UDT-a:
-tradicionalni UDT i
-CLR UDT.

U ranijim verzijama ms SQL Server-a korištena je sistemska procedura sp_addtype za kreiranje UDT-a, međutim od ms SQL Server-a 2005 je u upotrebi komanda CREATE TYPE za kreiranje UDT-a.
Osnovna sintaksa za kreiranje UDT-a:

```
CREATE TYPE [ ImeŠeme. ] ImeTipa
{
FROM SistemskiTipPodatka
[ (preciznost [ vrijednost ] )  ]
[ NULL | NOT NULL ] }
```

Kreiranjem tipa, novonastali tip se dodjeljuje šemi u kojoj se nalaze tabele koje će ga koristiti, dodjeljuje se tip podatka sa ograničenjem preciznosti, dužine, i mogućnosti novog UDT-a da sadržava NULL vrijednosti. Prilikom kreiranja UDT-a moguće je dodati "constraint" sa nabrojivim vrijednostima koje može sadržavati.
Glavna pretpostavka je da će se definisani UDT koristiti kao tip podatka u tabeli, međutim postoji mogućnost upotrebe UDT kao ulaznog parametra a i same strukture uskladištenih procedura, funkcija, pogleda, trigera itd..
Naravno pored kreiranja UDT-a postoji mogućnost i brisanja istog, međutim tada mora biti ispoštovano pravilo da se ukine svako referenciranje na UDT pa tek postoji mogućnost brisanja UDT-a.[2]
Sintaksa za brisanje UDT-a:

```
DROP TYPE [ ImeŠeme. ] ImeTipa
```

SQL Server 2008 omogućava kreiranje CLR UDT-a kao unapređenje već pomenutih T-SQL UDT-a. On daje veću fleksibilnost na dva polja.
Prvo, moguće je napisati .NET klasu ili strukturu koja može biti registrovana da se ponaša kao skalarni tip u okviru SQL Server sistemskih tipova. Kao i ostali tipovi u .NET-u, SQL Server UDT može usvojiti propertije i metode. Ove metode će biti dostupne iz korisničkog koda u bazi podataka.
Drugo, tip podatka UDT-a ne mora biti konzistentan sa osnovnim tipom podataka u SQL Server-u, nasuprot tome oni mogu biti zasnovani na kombinaciji više tipova podataka.[3]
Sintaksa za kreiranje CLR UDT-a :

```
CREATE TYPE [ ImeŠeme. ] ImeTipa
EXTERNAL NAME assemblyIme [ .ImeKlase ]
```

Sintaksa za brisanje CLR UDT-a:

```
DROP TYPE [ ImeŠeme. ] ImeTipa .
```

3.2 Karakteristike UDT-a u ms SQL Sever-u

Upotrebom UDT-a, može se kreirati tip podatka sa minimalnom složenošću, u koji može da se integriše logička validacija podataka, što pomaže u podizanju kvaliteta podataka. Upotreba UDT-a obezbijeđuje jednostavnije baze podataka i veću fleksibilnost i funkcionalnost. Posebnu prednost upotrebe predstavlja upotreba CLR UDT-a koji obezbijeđuje ne samo upotrebu jednog ugrađenog tipa podatka u SQL Serveru, već kombinacija istih čime sada postoji mogućnost kreiranja tipova za geografske podatke ili tipova za geometrijeske koordinate. [4]
Jedan od nedostataka je taj da ne postoji opcija izmjene UDT-a ili nekog njegovog dijela, što automatski otežava posao projektanta baze podataka, jer već unapred mora striktno znati kakav će UDT napraviti. Testiranjem performansi sistema prilikom upotrebe UDT-a i sistemskog tipa podataka došlo se do saznanja da upotreba UDT-a ne utiče ni na smanjenje ni na povećanje performansi sistema. [5]

4. IBM DB2

4.1 Uvod u korisnički definisane tipove podataka u IBM DB2

DB2 UDB obezbijeđuje deklarisanje specijalizovanih upotreba tipova podataka i pravila vezanih za njih, u oblik korisnički definisanog posebnog tipa podatka. UDT se gradi od sistemskih tipova podataka koje podržava IBM DB2, ali s obzirom da su nekompatibilni sa sistemskim tipovima, imaju potpuno drugačiju semantiku.

Sintaksa za kreiranje UDT je:

```
CREATE [DISTINCT] TYPE ImeDistinctTipa AS SistemskiTipPodataka
[WITH COMPARISONS]
```

Ključne riječi WITH COMPARISONS označavaju da će sistem automatski izgenerisati operatore poređenja, što je preporuka da bi bio kompatibilan sa ostalim DB2 proizvodima.
IBM dozvoljava dodavanje komentara UDT-u iz razloga lakšeg prepoznavanja ukoliko njegovo ime ne daje dobar opis samog UDT-a. Komentar se dodaje UDT-u sintaksom:

```
COMMENT ON DISTINCT TYPE  <komentar>.
```

Sistem za upravljanje IBM bazom podataka ne dozvoljava brisanje UDT-a ukoliko je on u upotrebi na bar jednom mjestu, međutim ako se zadovolji uslov da se prethodno izbrišu sve reference na dati UDT,brisanje se vrši komandom:

```
DROP DISTINCT TYPE <ImeDistinctTipa>;
```

Izmjena UDT-a nije podržana , te u slučajevima da je neophodna, mora se izvršiti kombinacijom komandi za

brisanje i kreiranje UDT-a, odnosno treba napraviti alternativni UDT, zatim izmjenom tabele definisati da atributi koriste alternativni UDT, izbrisati stari, nekorektni tip podatka, kreirati novi UDT sa korektnim atributima i ponovnom izmjenom tabele definisati da atributi umjesto alternativnog koriste novi UDT.

S obzirom da UDT nije kompatibilan ni sa jednim tipom sem sa samim sobom, a da bi se mogla praviti neka poređenja sa drugim tipovima mora se koristiti uz naredbu "CAST" koja omogućava da podatak jednog tipa podatka prevede u drugi tip podatka te se nakon toga može spajati ili porediti sa drugim podacima koji su istog tipa podatka kao i tip podatka u koji je UDT konvertovan upotrebom sistemske funkcije "CAST".

4.2 Karakteristike UDT-a u IBM DB2

1. "Strong typing" osigurava da se UDT ponaša na definisan način.
2. Mogućnost konvertovanja UDT-a u neki sistemski tip podataka, kao i vraćanje na izvorni UDT.
3. UDT je visoko integrisan u IBM sistem za upravljanje bazom podataka, interno su predstavljeni kao sistemski tipovi podataka, dijele isti kod za implementaciju sistemskih tipova podataka, operatora poređenja, indeksa, spajanja itd., što utiče na povećan kvalitet baze podataka bez gubitka performansi sistema.
4. UDT je temelj za većinu objektno-orijentisanih mogućnosti, predstavlja buduću podršku za abstraktne ili kompleksne tipove podataka. [6]

5. ORACLE 11g

5.1 Uvod u korisnički definisane tipove podataka u Oracle-u

Korisnički definisani tipovi podataka u Oracle-u se znatno rezlikuju od UDT-a koje je moguće implementirati u SQL Serveru, s obzirom da u Oraclu ne postoji klasični UDT, već "Object type", mogućnost da korisnik definiše svoj tip sa ograničenjima, metodama, atributima, pored toga "Object type" takođe može da predstavlja i čitavu tabelu. [7]

Objektni tip može da sadrži skalarni tip podatka, te se za njega kaže da je to jednostavni objektni tip, i koristi se za grupisanje kolekcije povezanih metoda i atributa.
Osnovna sintaksa za kreiranje objektnog tipa :

```
CREATE OR REPLACE TYPE NazivObjektnogTipa AS
OBJECT (
Promjenljiva1    [TipPodatka],
Promjenljiva1     [TipPodatka]
);
```

Međutim, moguće je da se unutar nekog objektnog tipa koristi promjenljiva kojoj je dodjeljen objektni tip podatka koji je ranije definisan.

Objektni tip takođe ima mogućnost da sadrži kolekciju drugih objekata u okviru svoje definicje, i takvi objektni tipovi se nazivaju referencirani objektni tipovi.

Oracle podržava dvije definicije kolekcija i to:

- Nizovni tip i
- Ugnježdene tabele.

Nizovni tip se koristi za kolekcije fiksne dužine, gdje se elementi čuvaju i kojima se pristupa po istom redosledu.
Sintaksa za definisanje nizovnog tipa:

```
CREATE OR REPLACE TYPE NazivObjektnogTipa AS
VARRAY(<BrojKolekcije>) OF <SistemskiTipPodatka>;
```

Ugnježdena tabela je neograničena i nesortirana kolekcija. Veoma je korisna za veće kolekcije nepoznate veličine.
Sintaksa za definisanje ugnježdene tabele:

```
CREATE OR REPLACE TYPE NazivObjektnogTipa AS
TABLE OF LineItem_objtyp
```

Nizovni tip se nalazi u samoj strukturi tabele i SQL upitima obuhvatamo cijelu kolekciju, što može negativno uticati na performanse same aplikacije, i najčešće se koristi za manji broj podataka.
Za razliku od nizovnog tipa, ugnježdeni tip dozvoljava pristup samo dijelu kolekcije i preporučuje se za rad na tabelama sa velikim brojem podataka jer ubrzava rad aplikacije.

Kompozitni objektni tip predstavlja kombinaciju objektnog tipa i drugog tipa (nizovnog ili ugnježdenog).

Metode objektnog tipa su procedure i funkcije koje određuju moguća ponašanja objektnog tipa podataka. Metode je moguće implementirati u aplikaciji, da bi se potom pozivale u objektnom tipu ili se implementiraju direktno u bazi podataka.

Postoje 4 vrste metoda objektnog tipa:
- "Member" metode – su preferirani način pristupa instanci objekta podataka ili ponašanje instanci objekta podataka.
- Konstruktor metode – vraća novu instancu objektnog tipa, te se preko konstruktora postavljaju podrazumjevane vrijednosti atributa unutar objektnog tipa
- Metode poređenja – služe za sortiranje podataka po objektnom tipu, i dijele sa na order metode (sortiraju po bilo kojem zadatom atributu u objektnom tipu) i map metode koje automatski uzimaju skalarnu vrijednost iz objektnog tipa te vrši sortiranje po njoj.
- Statičke metode – su metode koje se koriste za implementaciju modela objektnog tipa,npr. metoda koje kontrolišu životni vijek objekta. [8]

5.2 Karakteristike UDT-a u ORACLE-u

1.Enkapsulacija – S obzirom da korisnički definisani tipovi podataka postoje kao kompletni entiteti, uključujući definicije podataka, podrazumjevane vrijednosti i vrijednosti ograničenja, ovi entiteti osiguravaju jednolikost i konzistentnost. Jednom definisan, korisnički definisani tip podatka može da sudjeluje u drugim korisnički definisanim tipovima podataka, tako da isti logički tip podatka, uvijek

ima istu definiciju, podrazumjevanu vrijednost i ograničenja, bez obzira gdje se pojavljuje u bazi podataka.
2. Upotrebljivost- Pošto je hijerarhija zajedničkih struktura podataka sastavljena, mogu se ponovo upotrebiti u mnogim definicijama, štedeći vrijeme kodiranja uz osiguranje uniformnosti.
3. Fleksibilnost- Sposobnost da projektant baze podataka ima mogućnost da kreira tip podatka koji vjerno predstavlja podatak u bazi podataka kao što izgleda u stvarnosti, odnosno,u tzv. "real-world"-u.

6. ZAKLJUČAK

Analizom korisnički definisanih tipova podataka u tri vodeća sistema za upravljanje bazama podataka u svijetu, se vidi da je UDT uz pravilnu upotrebu i pažljivo projektovanje baze podataka veoma koristan u smislu lakšeg razvoja aplikacije i kasnijeg održavanja iste, činjenica o potrebi za UDT-om u razvoju informacionih sistemi leži i u tome da je masovna upotreba UDT-a u geografskim i matematičkim informacionim sistemima "natjerala" proizvođače da ugrade neke ranije korištene UDT u sistemeske tipove podataka, te se u novim verzijama SQL Server-a i ORACLE-a kao sistemski tip može koristiti geografska širina, dužina ili koordinate tačke, međutim još postoji potreba da korisnik definiše svoje tipove podataka koji se ponavljaju u bazi podataka, gdje dodjeljuje i ponašanje samog tipa uz ograničenja, te tako smanjuje mogućnost greške, recimo, kod definisanja polja istog tipa ,a različite dužine, te posle dolazi do grešaka prilikom spajanja.

Poređenjem mogućnosti korisnički definisanih tipova podataka ms SQL Server-a, ORACLE-a i IBMDB2 dolazi se do zaključka da SQL Server i IBM DB2 imaju dosta istih funkcija i mogućnosti, te da UDT definisan na baze podataka ova dva proizvođača služi da zadovolji potrebu za nestandardnim tipom podataka, međutim ORACLE-ov UDT obezbjeđuje daleko najbolje manipulacione mogućnosti i uspostavljanje funkcionalnosti, te predstavlja najbolji izbor, pogotovo ukoliko se radi sa kompleksnim objektima podataka. ORACLE je otišao korak dalje od ostalih proizvođača uvodeći UDT kao objektni tip, te stvorio mogućnost da se UDT ponaša, ne kao običan tip podatka, već kao kompletna tabela, što njegove mogućnosti čini daleko većim. Gledajući nedostatke upotrebe UDT-a, nameće se glavni nedostatak sva tri proizvođača, a to je nemogućnost vršenja izmjene tipa podatka, što u pitanje dovodi uopšte upotrebu UDT-a, iz razloga da ne smijemo sebi dopustiti grešku u projektovanju jer su naknadne greške neispravljive, a kao što je iz prakse poznato, ne postoji projekat koji se razvija, a da se ne jave greške u projektovanju ili moguća mjesta čijom bi se izmjenom došlo do poboljšanja performansi, sigurnosti , itd.

LITERATURA

[1] *American National Standard ANSI/ISO/IEC 9075-1:1999*, Washington: National Committee for Information Technology Standards 1999

[2] Joseph Sack, *SQL Server 2005 T-SQL Recipes- A problem . solution approach*, Berkeley : Apress 2006

[3] Julian Skinner and Robin Dewson , *Pro SQL Server 2005 Assemblies ,* Berkeley : Apress 2006

[4] Microsoft team, *Microsoft SQL Server 2005 implementation and maintenance*, Washington : Microsoft press 2006

[5] http://www.mssqltips.com/tip.asp?tip=1725

[6] IBM Support team, *DB2 UDB for AS/400 Object Relational Support*, Minnesota : IBM Corporation 2000

[7] Richard Walsh Earp and Sikha Saha Bagui , *Advanced SQL functions in Oracle 10g*, Texas : Wordware Publishing 2006

[8] R. Greenwald, R. Stackowiak, G. Dodge, D. Klein, B. Shapiro, C. G. Chelliah, *Professional Oracle Programming*, Indianapolis : Wiley Publishing Inc. 2005