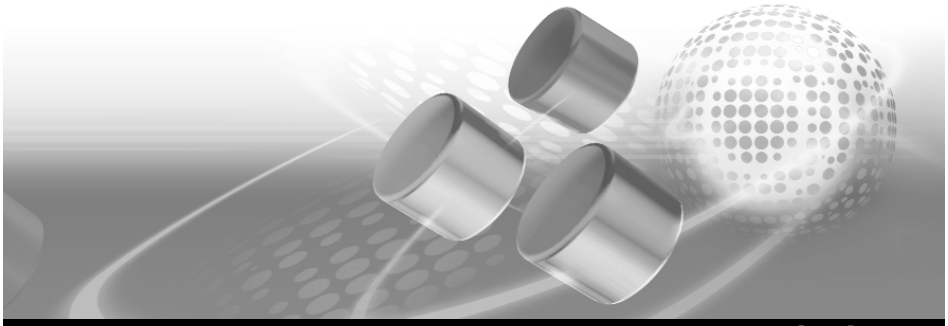

Sistemi baza podataka i SQL Server

SQL Server je sistem za upravljanje bazama podataka (DataBase Management System), koji je razvio i na tržište softvera plasirao Microsoft. On je najvažniji deo Microsoft Back Officea, poslovnog paketa klijent-server aplikacija. Kao dodatak SQL Serveru, Microsoft Back Office uključuje još i Windows NT Server, SNA Server, Systems Management Server, Exchange Server, Microsoft Transaction Server, Internet Information Server i MSMQ Server.

SQL Server se može koristiti isključivo pod Windows NT i Windows 95/98 operativnim sistemima. Microsoftovo opredeljenje da se usmeri samo na ta dva (svoja) operativna sistema ima mnogo prednosti i jedan nedostatak. Najvažnije prednosti u ovakvom pristupu su:

- **SQL Server funkcioniše** kao prirodno proširenje Windows NT sistema (t.j. Windowsa), zato što je čvrsto integrisan sa njim. Zbog toga, korisnici ne treba da uče drugi korisnički interfejs da bi mogli da rade sa ovim sistemom baze podataka.
- **SQL Server poseduje iste karakteristike** podešavanja (setup) i održavanja (maintenance), kao i Windows NT. To je ostvareno objedinjavanjem opcija jednostavne instalacije sistema, eliminisanjem mnogih složenih zadataka koji se tiču administriranja baza podataka i, opšte gledano, korišćenjem grafičkog okruženja za izvršavanje skoro svih zadataka administriranja.
- **SQL Server koristi uslužne servise** Windowsa NT i time nudi nove, ili proširene mogućnosti u radu sa bazama podataka, uključujući slanje i primanje poruka, kao i upravljanje i kontrolu bezbednosti pristupa i identifikacije (login security).

Sa druge strane, usredsređujući se jedino na Microsoftove operativne sisteme, SQL Server ne može iskoristiti prednosti nekih "naprednijih" operativnih sistema, kao što je, na primer, UNIX, koji u delu poboljšanih paralelnih arhitektura, ili kompletnih proračuna (enterprise computing) još uvek imaju prednost nad Windowsom NT.

Najvažnije karakteristike SQL Servera 7 su:

- **SQL Server je veoma jednostavan za korišćenje**.
- **Raspon u kome se SQL Server** može prilagoditi za korišćenje kreće se od laptop računara do sistema sa simetričnim multiprocesorima (Symetric MultiProcessor - SMP, sistem sa više procesora, koji svi "motre" i ravnopravno izvršavaju sve funkcije - prim.prev.).
- **SQL Server ima i mogućnosti** "skladištenja" podataka (data warehousing), koje su sve do sada bile na raspolaganju jedino kod Oraclea i nekih drugih, mnogo skupljih DBMS-a.

Skoro svi relacioni DBMS-i proistekli su iz UNIX operativnog sistema. To za posledicu ima da su postojeći korisnički interfejsi kojima ti sistemi raspolažu mnogo teži za korišćenje. Microsoftova namera je da od SQL Servera načini najjednostavniji sistem za implementaciju i upravljanje aplikacijama baza podataka. Jedan od načina na koji SQL Server 7 omogućava ostvarenje takvog cilja predstavlja korišćenje mnogih wizarda za izvršavanje gotovo svih zadataka administriranja.

Skalabilnost (prilagodljivost) znači da se isti DBMS može "pokrenuti" i na mobilnim laptop računarima i na jednoprocesorskim, ili multiprocesorskim hardverskim sistemima. Jedan od postignutih ciljeva pri kreiranju takvog DBMS-a je mogućnost prilagođavanja, koja se, kako je to već rečeno, kreće u rasponu od jednoprocesorskih računara do SMP sistema. DBMS-i postaju sve više povezani sa centralnim procesorom/procesorima (CPU-bound), zbog toga što su i aplikacije baza podataka u većoj meri "okrenute" centralnom procesoru (CPU - intensive database applications).

Microsoft je objedinio OLAP Server sa SQL Serverom 7, kako bi kreirao sveobuhvatan pristup za proces skladištenja podataka. OLAP Server treba da olakša kreiranje "skladišta" i "prodavnica" podataka, kako korišćenjem nove Microsoftove tehnologije, tako i korišćenjem postojećih tehnologija u softverima za skladištenje podataka drugih kompanija. U Delu IV naći ćete dodatne iscrpne informacije o mogućnostima skladištenja podataka koje pruža SQL Server.

NAPOMENA

SQL Server sistem baza podataka je prvobitno razvio i implementirao Sybase Inc. Microsoft je 1988. godine licencirao taj DBMS za OS/2 operativni sistem i nekoliko godina kasnije započeo njegovu primenu na Windows NT operativnom sistemu. Skoro u isto vreme odustalo se od daljeg razvoja SQL Servera za OS/2 operativni sistem. U aprilu 1994. godine Microsoft je okončao/raskinuo ugovor o saradnji sa Sybaseom Inc.

SQL Server je, od samog početka, zamišljen i projektovan kao klijent-server DBMS. Klijent-server arhitektura za upravljanje i kontrolu je razvijena na velikom broju različitih mrežom (network) povezanih računara (PC računari, radne stanice i SMP mašine/sistemi). Funkcionalnost SQL Servera je podeljena između klijenata i jednog, ili više servera. Klijent obezbeđuje jedan, ili više različitih korisničkih interfejsa, koji se koriste za formulisanje korisničkih zahteva DBMS-u. Server (t.j. DBMS) obrađuje te zahteve i rezultat vraća nazad klijentu.

NAPOMENA

Klijent-server arhitektura ne uključuje neophodno i DBMS. Takođe je moguće da u takvom okruženju postoje i drugi, posebno specijalizovani serveri, kao što su, na primer, print server, ili server "proračuna" (computing server). Međutim, DBMS je gotovo uvek sastavni deo klijent-server arhitekture.

Sistemi baza podataka: opšti pregled

Sistem baze podataka je celishodan skup različitih softverskih komponenti baze podataka i drugih baza podataka, koji sadrži sledeće delove:

- **programe aplikacija baze podataka**
- **ulazne (front-end),** t.j. klijent komponente
- **sisteme za upravljanje bazom podataka**
- **baze podataka**

Program aplikacije baze podataka je specijalizovani softver koji projektuju i primenjuju korisnici, ili je, pak, implementiran od neke posredne softverske kompanije (third-party software company - kompanija koja nije ni dobavljač, ni vlasnik - prim.prev.). Nasuprot tome, ulazne (front-end) komponente predstavljaju takve softvere baze podataka koje je projektovao i implementirao vlasnik baze podataka, ili su, pak, isporučene kao posredni (third-party) softver. Korišćenjem programa aplikacija baze podataka i ulaznih komponenti, korisnici mogu unutar baze podataka "upravljati" podacima, odnosno postavljati upite nad podacima.

Osnovni zadatak sistema za upravljanje bazom podataka sastoji se u upravljanju i kontroli podataka smeštenih u bazi podataka. Baza podataka može da se sagleda iz najmanje dve perspektive: sa strane korisnika i sa strane DBMS-a. Korisnici bazu podataka vide kao skup logično povezanih podataka. Za DBMS baza podataka je, jednostavno, niz bajtova, koji su, obično, snimljeni na disku.

Iako su ova dva pogleda na bazu podataka potpuno različita, oni, ipak, imaju i ponešto zajedničko. Sistem baze podataka ne treba samo da obezbedi interfejse koji korisnicima omogućavaju kreiranje baze podataka i preuzimanje, ili modifikovanje podataka, već i odgovarajuće komponente sistema za upravljanje i kontrolu snimljenih podataka. Sistem baze podataka mora imati sledeće mogućnosti/osobine:

- **raznovrsnost korisničkih interfejsa**
- **fizička nezavisnost podataka**
- **logička nezavisnost podataka**
- **optimizacija upita**
- **integritet podataka**
- **kontrola konkurentnosti**
- **backup i oporavak**
- **bezbednost i autorizacija**

U sledećim odeljcima dat je kratak opis ovih pojmova.

Raznovrsnost korisničkih interfejsa

Najveći broj baza podataka je projektovan i namenjen različitim tipovima korisnika, koji mogu imati različite nivoe znanja. Zbog toga, sistem baze podataka treba da raspolaže sa više različitih korisničkih interfejsa, koji omogućavaju postavljanje upita na osnovu primera (query-by-example—interfejs ekranskih formi, u kome korisnik daje primere, a softver na osnovu njih izvodi, na primer, kriterijume pretrage - prim.prev.), korišćenje običnog jezika (natural language - korišćenje pisanog, ili govornog jezika, umesto programskog jezika računara; najčešće se u te svrhe koristi engleski jezik - prim.prev.), kao i pripremu odgovarajućih formi (formulara) za krajnje korisnike, dok za iskusne korisnike treba da omoguće korišćenje interaktivnog jezika upita (interactive query language).

Fizička nezavisnost podataka

Ukoliko postoji fizička nezavisnost podataka označava da programi aplikacija baze podataka ne zavise od fizičke strukture podataka smeštenih u bazi podataka. Ta veoma važna osobina omogućava izvođenje promena na (smeštenim/snimljenim) podacima, a da za to nije potrebno izvršiti bilo kakvu promenu na programima aplikacija baze podataka. Na primer, ukoliko su podaci smešteni u bazi prethodno poređani na osnovu jednog kriterijuma i ukoliko kasnije treba da se urede prema nekom drugom kriterijumu, modifikacija ne treba da utiče na postojeće aplikacije baze podataka, ili na postojeću šemu baze podataka (database scheme - opis baze podataka koji je generisan korišćenjem jezika za definisanje podataka DBMS-a).

Logička nezavisnost podataka

Tokom procesiranja fajlova (korišćenjem tradicionalnih programskih jezika), deklaraciju nekog fajla postavlja program aplikacija, tako da svaka izmena strukture tog fajla, obično, zahteva i modifikaciju svih programa koji taj fajl koriste. Sistemi baza podataka omogućavaju logičku nezavisnost podataka—drugim rečima, moguće je izvršiti izmenu na logičkoj strukturi baze podataka nezavisno od programa aplikacija baze podataka. Na primer, ukoliko u DBMS postoji šema nekog objekta pod nazivom PERSON i ukoliko želimo da tom objektu dodamo neki atribut (na primer, adresu), mora se izmeniti samo logička struktura baze podataka, a da, pri tom, ne treba menjati ni jedan od postojećih programa aplikacija baze podataka.

Optimizacija upita

Svaki sistem baze podataka sadrži i komponentu pod nazivom optimizator (optimizer), kojom se mogu proceniti različite “strategije” izvršavanja/postavljanja upita nad podacima, nakon čega se bira najefikasniji pristup. Izabrana strategija se označava kao plan izvršavanja upita (execution plan of the query). Optimizator daje procenu, razmatrajući, na primer, koliko su velike tabele na koje se upiti odnose, koji indeksi postoje i koji se Booleovi (logički) operatori koriste u WHERE klauzuli. (optimizacija upita je, sa svim neophodnim detaljima, razmotrena u Poglavlju 10).

NAPOMENA

Kod nekih ranijih aplikacija baza podataka, koje su implementirane korišćenjem low-level jezika baze podataka (jezika napisanih za određenu procesorku/mašinsku arhitekturu - prim.prev.), programer je birao plan izvršavanja upita.

Integritet podataka

Jedan od zadataka DBMS-a je identifikacija logičke nekonzistencije (nepovezanosti, ili neispravnosti) podataka, koje bi trebalo izbaciti iz baze podataka (datum 30. februar, ili vreme 5:77:00 po podne su dva primera takvih podataka). Pored toga, najveći broj problema iz realnog života koji se implementiraju korišćenjem sistema baze podataka imaju i tzv. ograničenja integriteta (*integrity constraints*), koja moraju biti istinita (true) za podatke iz baze (ukupan broj zaposlenih u nekoj kompaniji predstavlja jedan primer ograničenja integriteta, jer mora biti neki najviše petocifreni ceo broj). Zadatak održavanja integriteta baze podataka mogu izvršavati korisnik u programima aplikacija baze podataka, ili sam DBMS. Koliko god je to moguće, ovaj zadatak treba da izvršava i kontroliše DBMS (integritet podataka se razmatra u dva poglavlja ove knjige: deklarativni u Poglavlju 4, a proceduralni u Poglavlju 13).

Kontrola konkurentnosti

DBMS je multikorisnički softverski sistem, što znači da mnogo korisničkih aplikacija istovremeno pristupa bazi podataka. Zbog toga, svaki DBMS mora da poseduje neku vrstu kontrolnog mehanizma, kako bi se osiguralo da veći broj aplikacija koje eventualno pokušavaju da ažuriraju/izmene iste podatke to rade na neki kontrolisani način. U daljem tekstu je dat primer jednog takvog problema koji se može javiti ukoliko DBMS ne sadrži takve kontrolne mehanizme:

1. **Vlasnici bankovnog računa** 4711 u banci X imaju na računu 2.000 dolara.
2. **Dva zajednička vlasnika tog računa** otišli su u dve različite ekspoziture i u istom trenutku podigli po 1.000 dolara.

Nakon tih transakcija, stanje na računu 4711 treba da bude 0, a ne 1.000 dolara.

Svi DBMS-i moraju imati mehanizme za razrešavanje situacija poput ove u navedenom primeru. Kontrola konkurentnosti je detaljno razmotrena u Poglavlju 14.

Backup i oporavak baze podataka

DBMS mora imati podsistem koji će biti odgovoran za oporavak u slučaju hardverskih, ili softverskih grešaka. Na primer, ukoliko se greška javi u trenutku kada neka aplikacija baze podataka ažurira stotine redova neke tabele, podsistem za oporavak mora da poništi sva prethodno izvedena ažuriranja, kako bi se osigurala konzistencija podataka nakon pojave geške. (više detalja o backupu i oporavku naći ćete u Poglavlju 20).

Bezbednost i autorizacija

Bezbednost podataka označava da su u bazi podataka zaštićeni od neautorizovanog pristupa određenih korisnika, odnosno od zloupotrebe. Na primer, pristup delu podataka koji su označeni kao zarada (salary) i koji sadrže podatke o platama zaposlenih u nekoj kompaniji treba da bude dozvoljen, ili odobren samo autorizovanim (ovlašćenim) osobama. Pored toga, može se podesiti da neki korisnici podatke mogu samo pregledati (read only acces), dok drugi mogu i pregledati i unositi podatke (write acces).

Svaki DBMS raspolaže sa nekom vrstom kontrole autorizacije/ovlašćenja, za šta se, obično, koriste tzv. korisnički nalozi (user accounts), kojima se korisnicima sistema odobravaju (grant), ili ukidaju/poništavaju (revoke) neke od privilegija/povlastica (privileges). U Poglavlju 12 razmotreni su ovi pojmovi sa mnogo više detalja.

Sistemi relacionih baza podataka

SQL Server je relaciona baza podataka. Pojam relacione baze podataka je prvi put upotrebio E. F. Codd 1970. godine u svom članku "Relacioni model podataka za velike deljive 'banke' podataka" ("A relational model of data for large shared data banks"). Za razliku od ranijih sistema baza podataka (mrežni i hijerarhijski), *sistemi relacionih baza podataka* su zasnovani na relacionom modelu podataka, koji ima čvrsto matematičko zaleđe. To znači da je model podataka u sistemima relacionih baza podataka zasnovan na relacionoj algebri, koja, pak, predstavlja skup operacija za manipulisanje relacijama.

NAPOMENA

Model podataka predstavlja skup koncepata/načela, relacija i njihovih ograničenja, koji se koristi za pogodno predstavljanje podataka nekog realnog problema.

Centralni koncept relacionog modela podataka je relacija, odnosno tabela. Zbog toga, sa gledišta korisnika, relacione baze podataka sadrže tabele i ništa više. U tabeli mogu biti jedna, ili više kolona i nula, ili više redova. U preseku svakog reda i kolone tabele (okcu/ćeliji tabele) je uvek tačno jedan podatak.

Rad sa bazom podataka koja je data kao primer u ovoj knjizi

Baza podataka koja je data kao primer u ovoj knjizi predstavlja kompaniju sa odeljenjima i zaposlenima; svako odeljenje ima jednog, ili više zaposlenih, od kojih svaki u isto vreme radi na jednom, ili više određenih projekata - za svaki projekat angažovani su jedan, ili više zaposlenih.

Podaci u bazi podataka datoj kao primer mogu se prikazati korišćenjem četiri tabela:

- **department** (odeljenje)
- **employee** (zaposleni)
- **project** (projekat)
- **works_on** ("radi na")

Na slikama 1-1 do 1-4 prikazane su sve tabele ove baze podataka.

Tabela **department** prikazuje sva odeljenja kompanije. Svako odeljenje ima sledeće atribute:

department (dept_no, dept_name, location)

U **dept_no** koloni nalaze se jedinstveni (unique) brojevi svakog odeljenja ponaosob, **dept_name** daje nazive odeljenja, a **location** predstavlja sedišta odgovarajućih odeljenja.

Tabela **employee** prikazuje sve zaposlene u kompaniji. Svaki od njih ima sledeće atribute:

employee (emp_no, emp_fname, emp_lname, dept_no)

emp_no predstavlja jedinstveni (identifikacioni) broj svakog zaposlenog, **emp_fname** i **emp_lname** su ime i prezime svakog zaposlenog, respektivno, a **dept_no** je broj odeljenja kome zaposleni pripada.

Svi projekti kompanije prikazani su u tabeli **project**, koja ima sledeće kolone:

project (project_no, project_name, budget)

project_no predstavlja jedinstveni broj projekta, a **project_name** i **budget** navode nazive i budžete svakog projekta, respektivno.

Tabela **works_on** prikazuje relacije između zaposlenih i projekata. Ova tabela ima sledeće kolone:

works_on (emp_no, project_no, job, enter_date)

emp_no prikazuje (identifikacione) brojeve zaposlenih, a **project_no** (identifikacione) brojeve projekata na kojima zaposleni rade. Kombinacija vrednosti podataka iz ove dve kolone je uvek jedinstvena. **job** i **enter_date** prikazuju poslove/zanimanja i datume početaka rada zaposlenih na odgovarajućim projektima, respektivno.

Slika 1.1 Tabela department

dept_no	dept_name	location
d1	Research (istraživanje)	Dallas
d2	Accounting (računovodstvo)	Seattle
d3	Marketing	Dallas

Slika 1.2 Tabela employee

emp_no	emp_fname	emp_lname	dept_no
25348	Matthew	Smith	d3
10102	Ann	Jones	d3
18316	John	Barrimore	d1
29346	James	James	d2
9031	Elke	Bertoli	d2
2581	Elisa	Kim	d2
28559	Sybill	Moser	d1

Slika 1.3 Tabela project

project_no	project_name	budget
p1	Apollo	120000
p2	Gemini	95000
p3	Mercury	185600

Slika 1.4 Tabela works_on

emp_no	project_no	job	enter_date
10102	p1	Analyst (analitičar)	1997.10.1 00:00:00
10102	p3	Manager (direktor)	1999.1.1 00:00:00
25348	p2	Clerk (činovnik)	1998.2.15 00:00:00
18316	p2	NULL	1998.6.1 00:00:00
29346	p2	NULL	1997.12.15 00:00:00
2581	p3	Analyst	1998.10.15 00:00:00
9031	p1	Manager	1998.4.15 00:00:00
28559	p1	NULL	1998.8.1 00:00:00
28559	p2	Clerk	1999.2.1 00:00:00
9031	p3	Clerk	1997.11.15 00:00:00
29346	p1	Clerk	1998.1.4 00:00:00

Korišćenjem ove baze podataka, mogu se opisati neka važna svojstva sistema relacionih baza podataka:

- **Redovi u tabelama** ne moraju da imaju neki određeni redosled.
- **Kolone u tabelama** ne moraju da imaju neki određeni redosled.
- **Svaka kolona** u tabeli mora da ima jedinstveni naziv. Sa druge strane, kolone iz različitih tabela mogu imati iste nazive. Na primer, ova baza podataka ima kolonu **dept_no** u tabeli **department** i kolonu sa istim nazivom u tabeli **employee**.
- **Svaka pojedinačna stavka** (item) podataka u jednoj tabeli mora imati samo jednu vrednost. To znači da u svakoj ćeliji tabele (preseku neke kolone i nekog reda) nikada ne može biti grupa (više vrednosti) podataka.
- **Za svaku tabelu mora postojati bar jedan identifikator** - u tabeli mora postojati takva (kombinacija) kolona koja ima svojstvo da ne postoje dva reda sa istim (kombinacijama) vrednostima podataka. U relacionom modelu podataka jedan takav identifikator se naziva kandidat ključ (*candidate key*). Ukoliko u tabeli postoji više od jednog kandidat ključa, "projektant" baze podataka jedan od njih označava kao glavni ključ (*primary key*) tabele. Na primer, kolona **dept_no** je primarni ključ tabele department; kolone **emp_no** i **project_no** su glavni ključevi tabela **employee** i project, respektivno. Konačno, primarni ključ tabele **works_on** je kombinacija dvaju kolona (**emp_no**, **project_no**).
- **U tabeli nikada ne mogu postojati dva identična reda**. Ovo svojstvo je samo teorijski pristup; SQL Server, kao i sve ostale relacione baze podataka, u opštem slučaju, dozvoljavaju postojanje identičnih redova unutar neke tabele.

SQL: Jezik relacionih baza podataka

SQL Serverov jezik relacionih baza podataka nosi naziv Transact-SQL. Transact-SQL je "dijalekat" najvažnijeg jezika baza podataka današnjice -SQL-a, što predstavlja skraćenicu od Structured Query Language. Nastanak SQL-a blisko je povezan sa projektom pod nazivom System R, koji je projektovao i softverski zaokružio IBM, ranih 80-tih godina. Taj projekat je pokazao da je moguće, koristeći teorijske postavke koje dao E. F. Codd, izgraditi sistem relacione baze podataka. SQL je (kao jezik) izgrađen u prvoj fazi projekta, čiji je cilj bio implementacija prototipa sa ograničenom funkcionalnošću.

Nakon uspeha Systema R, veliki broj novih kompanija je, koristeći SQL jezik, izgradio sopstvene sisteme relacionih baza podataka. Svi ti sistemi bili su prošireni dijalekti originalnog jezika, zavisno od toga kako je koja kompanija implementirala sopstvena proširenja. Zbog toga su American National Standards Institute (ANSI) i International Standards Organization (ISO) 1982. godine osnovali zajednički komitet, sa ciljem da se projektuje standardna verzija SQL-a. Prvi standard, koji je, uglavnom, bio zasnovan na IBM dijalektu tog jezika, predstavljen je četiri godine kasnije. Nakon što je 1989. godine bio usvojen prelazni standard, u decembru 1992. je konačno razvijen mnogo obimniji standard, pod nazivom SQL92. Trenutno, obe organizacije razvijaju novi standard, SQL3, koji treba da sadrži više novih koncepata baza podataka, uključujući i pokretačke događaje (triggers), snimljene procedure (stored procedures), kao i druge brojne objektno orijentisane koncepte. Najvažniji deo SQL3 standarda - Foundations (Osnova) je završen u 1999. godini.

Za razliku od tradicionalnih, kao što su C, C++ i Java, SQL je "grupno orijentisani" (set-oriented) jezik (navedeni jezici se još nazivaju i "zapisno orijentisani" - record-oriented, ili "jedan zapis u svakom trenutku orijentisani" - record-at-time jezici.) To znači da se SQL-om upiti mogu postavljati na većem broju redova jedne, ili više tabela, korišćenjem samo jedne naredbe. Ta osobina predstavlja jednu od najvažnijih prednosti SQL-a, jer dozvoljava korišćenje ovog jezika na višem logičkom nivou, nego što je to slučaj sa proceduralnim jezicima.

Drugo važno svojstvo SQL-a je neproceduralnost. Svaki program napisan u nekom od proceduralnih jezika (C, C++, Java) opisuje, korak po korak, kako neki zadatak treba da se izvrši. Nasuprot tome, SQL, kao i bilo koji drugi neproceduralni jezik, opisuje šta je to što korisnik želi. Zbog toga je sistem "odgovoran" za nalaženje odgovarajućeg načina za izvršenje određenog zahteva korisnika.

SQL, kao i svi drugi jezici baza podataka, sadrži dva podjezika: za definisanje podataka (DDL—Data Definition Language) i za modifikovanje podataka (DML—Data Modifikation Language). DDL naredbe se koriste za opisivanje šeme tabela baze podataka. DDL sadrži tri "generičke" naredbe: CREATE objekat, ALTER objekat i DROP objekat. Ovim naredbama se kreiraju, menjaju i uklanjaju objekti baze podataka (objekat može biti baza podataka, tabela, kolona, ili indeks). Ove naredbe su detaljno razmotrene u Poglavlju 4. Za razliku od njih, DML obuhvata sve operacije kojima se upravlja/manipuliše podacima. Uvek postoje četiri "generičke" operacije za manipulisanje bazom podataka: preuzimanje/obnavljanje podataka (retrieval), dodavanje/umetanje podataka (insertion), brisanje/uklanjanje podataka (deletion) i modifikovanje/menjanje podataka (modification). Naredba za preuzimanje podataka SELECT opisana je u poglavljima 5 i 6, dok su INSERT, DELETE i UPDATE naredbe detaljno razmotrene u Poglavlju 7.

Konvencije sintakse

U ovoj knjizi se za sintaksu Transact-SQL naredbi i oznake u tekstu koriste konvencije prikazane u tabeli 1-1.

NAPOMENA

Za razliku od srednjih/ugaonih zagrada (bracket—[ ]) i velikih/vitičastih zagrada (braces—{ }), koje su deo konvencija označavanja/sintakse, male zagrade, ( ), su sastavni deo zapisa neke naredbe i moraju se uvek otkucati!

Tabela 1.1: Konvencije sintakse/označavanja

Konvencija	Označava
Italik/kurziv	novi izraz, ili posebno naglašavanje nekog pojma
VELIKA SLOVA	Transact-SQL ključne reči, na primer, CREATE TABLE - dodatne informacije o Transact-SQL ključnim rečima mogu se naći u Poglavlju 2
mala slova	promenljive unutar Transact-SQL naredbi, na primer, CREATE TABLE nazivtabele (korisnik mora reč nazivtabele zameniti stvarnim nazivom tabele)
var1 | var2	alternativno korišćenje ili var1, ili var2 (možete izabrati neku od opcija odvojenih vertikalnom crticom)
{ }	alternativno korišćenje više nabrojanih stavki/opcija; na primer: { expression | USER | NULL }
[ ] [FOR LOAD]	neobavezujuće (opcione) stavke/opcije napisane unutar zagrada; na primer:
{ } …	stavke unutar zagrada mogu se ponavljati neograničeni broj puta; na primer: {, @param1 typ1} …
podebljan/bold	naziv objekta baze podataka (neku bazu podataka, tabelu, kolonu) u tekstu
podrazumevana vrednost	podrazumevana vrednost je uvek podvučena; na primer: ALL | DISTINCT

Projektovanje baze podataka

Projektovanje neke baze podataka je veoma važna faza u "životnom ciklusu" baze podataka, koja prethodi svim drugim fazama, osim prikupljanju zahteva i analizi. Ukoliko se baza podataka projektuje uglavnom intuitivno, bez nekog određenog plana, ona, najverovatnije, neće ispuniti očekivanja korisnika koja se tiču performansi/učinka tokom rada. Druga posledica lošeg projektovanja je preterana redundantnost (dupliranje, ili ponavljanje) podataka, što, samo po sebi, vodi do naredne dve neželjene posledice: postojanja nepravilnosti u zapisu podataka i zauzimanja previše prostora na disku.

Normalizacija podataka je proces kojim se postojeće tabele baze podataka testiraju, kako bi se pronašle određene zavisnosti (dependencies) između kolona tabele. Ukoliko takve zavisnosti postoje, vrši se prestruktuiranje originalne tabele u više tabela (obično dve), čime se eliminiše zavisnost između kolona tabele. Ukoliko neka od novogenerisanih tabela i dalje sadrži zavisne podatke, proces normalizacije se mora ponoviti, sve dok se ne otklone sve zavisnosti.

Proces eliminisanja redundantnosti podataka iz neke tabele zasnovan je na teoriji funkcionalnih zavisnosti (theory of functional dependencies). Pojam *funkcionalne zavisnosti* znači da se na osnovu poznatih vrednosti podataka jedne kolone uvek mogu jednoznačno odrediti odgovarajuće jedinstvene vrednosti podataka iz druge kolone (isto važi i za grupe kolona). Funkcionalna zavisnost koja postoji između kolona A i B neke tabele označava se kao AB, čime se, zapravo, označava da neka vrednost kolone A može uvek da se iskoristi za određivanje odgovarajuće vrednosti kolone B ("B je funkcionalno zavisno od A").

Sledeći primer prikazuje funkcionalnu zavisnost između dva atributa tabele **employee** iz baze podataka koja se koristi kao primer u ovoj knjizi.

PRIMER 1.1

emp_no→emp_lname

Pošto svi u kompaniji imaju jedinstvene vrednosti (identifikacionih) brojeva, može se na osnovu tih vrednosti odrediti prezime odgovarajućeg zaposlenog (kao i svi ostali njegovi atributi). Ovakva vrsta funkcionalne zavisnosti, u kojoj neka kolona zavisi od ključa tabele, označava se kao trivijalna. Druga vrsta zove se viševrednosna (multivalued dependency). Za razliku od upravo opisane (*trivijalne*) funkcionalne zavisnosti, viševrednosna se odnosi na atribute sa više vrednosti (multivalued attributes). To znači da se, na osnovu poznatih vrednosti jednog atributa (kolone), može jednoznačno odrediti skup vrednosti nekog drugog viševrednosnog atributa. Viševrednosna zavisnost se označava sa →→.

Sledeći primer ilustruje viševrednosnu zavisnost koja važi za dva atributa objekta BOOK.

PRIMER 1.2

ISBN→→Authors

U ISBN oznaci neke knjige uvek su sadržani svi autori te knjige. Zbog toga, atribut **Authors** je viševrednosno zavisan od atributa **isbn**.

Normalne forme

Normalne forme se koriste u procesu normalizacije podataka, odnosno u toku projektovanja baze podataka. Teorijski, postoji najmanje pet normalnih formi, ali su prve tri od najvećeg praktičnog značaja. Treća normalna forma neke tabele se može postići testiranjem prve i druge, kao prelaznih međukoraka. Obično se cilj projektovanja baze podataka može uspešno ostvariti ukoliko su sve tabele baze podataka u trećoj normalnoj formi.

NAPOMENA

Viševrednosna zavisnost se koristi za testiranje četvrte normalne forme neke tabele. Zbog toga, ovu vrstu zavisnosti nećemo dalje razmatrati.

Prva normalna forma

Prva normalna forma (1NF) neke tabele označava da ta tabela nema atribute sa više vrednosti, ili složene atribute (složeni atribut sadrži druge atribute i može se podeliti na manje delove). Sve relacione baze podataka su po definiciji u 1NF, zato što vrednosti u svakoj koloni nekog reda moraju biti "elementarne" (*atomic*) - moraju imati samo jednu vrednost.

Prvu normalnu formu demonstriraćemo na primeru za koji ćemo iskoristiti deo tabele **works_on** iz baze podataka date kao primer (slika 1-5). Redovi tabele **works_on** se mogu zajedno grupisati, ukoliko se za to iskoristi odgovarajući broj zaposlenih (employee number). Tabela na slici 1-6 nije u 1NF, zato što kolona **project_no** sadrži grupu (dve) vrednosti (p1 i p3).

Slika 1.5 Deo Tabele works_on

emp_no	**project_no**	**…….**
10102	p1	…….
10102	p3	…….
…….	…….	…….

Slika 1.6 Ova tabela nije u 1NF

emp_no	**project_no**	…….
10102	(p1, p3)	…….
…….	…….	…….

Druga normalna forma

Tabela ima drugu normalnu formu (2NF), ukoliko već ima 1NF i ukoliko u tabeli nema kolone koja nije ključ zavisan od dela ključa (parcijalnog ključa) tabele. To znači da ukoliko (A, B) predstavlja kombinaciju dvaju kolona tabele od kojih je izgrađen ključ, u toj tabeli nema kolone koja zavisi samo od A, ili samo od B (kolone zavise od čitavog ključa, ne od nekog njegovog dela).

Na primer, razmotrimo tabelu prikazanu na slici 1.7, koja je identična tabeli **works_on**, izuzev što ima i kolonu **dept_no**. Glavni ključ te tabele je kombinacija kolona (**emp_no**, **project_no**). Kolona **dept_no** zavisi od parcijalnog ključa **emp_no** (a nezavisna je od dela ključa **project_no**), tako da razmatrana tabela nema 2NF (originalna tabela **works_on** je u 2NF).

NAPOMENA

Svaka tabela za koju je jedna kolona glavni ključ uvek ima 2NF.

Treća normalna forma

Tabela je u trećoj normalnoj formi (3NF) kada je u 2NF i ukoliko između kolona koje nisu iskorišćene za pravljenje ključa nema funkcionalnih zavisnosti. Na primer, tabela sa slike 1.8, koja je, osim što ima dodatnu kolonu **dept_name**, identična tabeli employee, nije u 3NF, zato što se za svaku poznatu vrednost iz kolone **dept_no** može jednoznačno odrediti odgovarajuća vrednost u koloni **dept_name** (originalna tabela **employee**, kao i sve ostale tabele iz baze podataka date kao primer, ima 3NF).

Slika 1.7 Tabela works_on1

emp_no	project_no	Job	enter_date	dept_no
10102	p1	Analyst	1997.10.100:00:00	d3
10102	p3	Manager	1999.1.100:00:00	d3
25348	p2	Clerk	1998.2.1500:00:00	d3
18316	p2	NULL	1998.6.100:00:00	d1
……..	……..	……..	……..	……..

Slika 1.8 Tabela employee1

emp_no	emp_fname	emp_lname	dept_no	dept_name
25348	Mathew	Smith	d3	Marketing
10102	Ann	Jones	d3	Marketing
18316	John	Barrimore	d1	Research
29246	James	James	d2	Accounting
……..	……..	……..	……..	……..

Opšti pregled Microsoft SQL Server-a

Najvažnije prednosti Microsoft SQL Servera su:

- **SQL Server radi kao prirodno proširenje** Windows NT i Windows 95/98 operativnih sistema.
- **Zahvaljujući upotrebi grafičkog okruženja** za gotovo svaki zadatak administriranja sistema i baze podataka, SQL Server je relativno jednostavan za korišćenje i kontrolu.
- **SQL Server koristi uslužne programe** Windowsa NT i, time, pruža nove, ili proširuje stare mogućnosti rada sa bazama podataka, uključujući slanje i primanje poruka, kao i upravljanje bezbednosnim pristupom i identifikacijom (login security).
- **SQL Server je jednostavan za korišćenje**.
- **SQL Server se može prilagoditi za rad u opsegu** od mobilnih laptop računara do sistema sa simetričnim multiprocesorima.

- **SQL Server poseduje mogućnosti skladištenja podataka**, koje su do sada stajale na raspolaganju samo kod Oraclea i drugih, mnogo skupljih DBMS-a.

Zaključak

SQL Server je sistem za upravljanje relacionim bazama podataka koji koristi klijent-server podelu (distribuciju) poslova/zadataka. Kao i svi drugi sistemi za upravljanje bazama podataka, i on poseduje sledeće mogućnosti/osobine:

- **raznovrsnost korisničkih interfejsa**
- **fizička nezavisnost podataka**
- **logička nezavisnost podataka**
- **optimizacija upita**
- **integritet podataka**
- **kontrola konkurentnosti**
- **backup i oporavak**
- **bezbednost i autorizacija**

U sledećem poglavlju razmotrićemo dve najvažnije SQL Server alatke: SQL Server Enterprise Manager i SQL Server Query Analyzer. SQL Server Enterprise Manager je alatka sistem administriranja, kojom se može izvršiti skoro svaki zadatak vezan za sisteme baza podataka. Sa druge strane, SQL Server Query Analyzer je korisnička (end-user) alatka za izvršavanje i analiziranje ad hoc upita. Poglavlja 1 i 2 predstavljaju uvodni deo ove knjige.