## Uvod

SQL(structured query language) je jezik koji se koristi za slanje upita ka RDBMS(relational database management system) koji skladisti podatke po relacionom modelu. Relacioni model je semanticki model za prezentovanje podataka koji se bazira iz dve grane matematike – set theory(teorija skupova) i predicate logic.

T-SQL je dijalekt SQL-a.

### SQL

SQL je ANSI i ISO standard language za queryovanje I storovanje podataka RDBMS-a.

SQL se sastoji iz nekoliko kategorija izraza:  
- **DML** (data manipulation language) (select, update, insert… tipovi izraza za upit i manipulaciju podataka)  
- **DDL** (data definition language) (create, alter, drop tipovi izraza, za manip. definicija podataka)  
- **DCL** (data control language) (grant, revoke… tipovi izraza za manipulaciju dozvola)

Ova knjiga se fokusira na DML tipove izraza, znaci SELECT, INSERT, UPDATE, DELETE, TRUNCATE, and MERGE

Common misunderstanding is that TRUNCATE is a DDL statement, but in fact it is a DML statement

### Set Theory

*By a set we mean any collection M into a whole of definite, distinct objects “m” (which are called the “elements” of M) of our perception or of our thought.*

* *Georg Cantor*

//Mozes malo vise prouciti o skupovima kasnije ako te interesuje, ali osnovna skola

### Predicate logic

Koristi se za upravljanje podacima i izvrsavanje upita nad podacima. PREDICATE je u sustini izraz/svojstvo koje je tacno ili nije tacno(holds, or doesn’t hold)

Relacioni model se oslanja na predicate za zadrzanje logickog integriteta podataka i definisanje strukture istih. U sustini primer bi bio CONSTRAINT koji ce nad tabelom “employees” da postavi pravilo da plata mora da bude > 0 => salary > 0

### The relational model

Relacioni model je sematicki model za upravljanje podacima i manipulaciju koji se bazira na teoriji skupova i logici. Cilj ovog modela jeste omogucavanje konzistentne prezentacije podataka uz minimalnu ili bez imalo redundantnosti gde ne moras da se oslanjas na intuiciju jer jelte, bazira se na matematici.

Relacioni model ukljucuje koncepte poput “propositions, predicates, relations, tuples, attributes & more”.

### Propositions, predicates & relations

Da deo “relacioni” u izrazu “relacioni model” stoji tu na osnovu koncepta “relacija” je netacan. Zapravo potice iz matematickog izraza relacija. U teoriji skupova, relacija je representation(predstava) skupa. U relacionom modelu, relacija je skup (related) informacija. U SQL-u taj skup se naziva tabela(not exact counterpart, but close).

Key point -> relacija jeste jedna tabela.

Operacije nad relacijama na osnovu relacione algebre rezultuju relacijama

//Relational model razlikue relaciju od relacione promenljive but to keep things simple, necemo o tome sada. Koristicemo izraz “relacija” za oba.

Relacija se sastoji iz headinga i bodyja.

Heading jeste spisak atributa(kolona u SQL-u) gde svaka kolona ima svoj naziv i tip podatka koje ce da sadrzi.

Body jeste skup tuple-a(row u SQL-u) gde se svaki element identifikuje kroz key. To KIS, reci cemo da je tabela skup redova.

A diagram of a workflow

AI-generated content may be incorrect.

Keep in mind da elementi u skupu(tuples) i kolone/atributi mozda mogu izgledati kao da imaju redosled. Oni nemaju redosled(order) i ne mogu imati redosled, po definiciji skupova.

To isto vazi i za recorde unutar tabele.

Kada formalizujes neku pojavu u prirodu tako da je mozes prezentovati kroz skup – ti ces u sustini koristiti razne predicates. Za opisivanje kolone/atributa, npr ti ces koristiti TIP(type).

**TYPE** je najjednostavniji oblik predikata. Npr int ogranicava kolonu da vrednost bude u opsegu –2,147,483,648 do 2,147,483,647. Tip nije samo integer/date. To su jednostavni tipovi.

Tip moze biti i kompleksan gde bi kompleksan bio npr neka enumeracija vrednosti.

### Missing values

Jelte, postoji debata o tome da li bi predikat trebao podrzavati dvo-vrednostne ili trovrednosne vrednosti. Dvovrednosni predicate bi bio npr true/false.

Trovrednosni bi bio true/false/null tj nepostojanje vrednosti. Dvovrednosni sistem podrazumeva “the law of excluded middle” sto je matematicko pravilo koje kaze da “nema sredine”. Ili jesi, ili nisi. Sa Trovrednosnim sistemom ti kazes “unknown/ne znam” da je takodje vrednost.

Neki tvrde da se NULL/unknown i trovrednosni predicate logic na ubrajaju u relacioni model.

SQL supports three value predicate logic.  
Postoji “Codd” koji tvrdi da treba da postoji cetvorovrednosni predicate sistem u kom kada postoji nepoznata vrednost(null), ona se deli na “primenjivu” i “neprimenjivu”. Neprimenjiva bi bila kada employee nema mobilni telefon pa onda postojanje atributa mobilnog telefona nema primenu tj non-applicable.

Ukoliko employee ima mobilni telefon onda je “primenjiva” nepoznata vrednost jer jelte, ima telefon samo mi ne znamo koji je. SQL ne podrzava cetvorovrednosni predicate sistem.

### Constraints

Jedna od prednosti relacionog modela jeste mogucnost za definisanjem integriteta podataka kao deo modela. Constraints su pravila koja se jelte definisu u modelu, a koje sprovodi RDBMS. Jedan od najcescih jeste da li atribut dozvoljava ili ne dozvoljava null marker.

Constraints takodje se namecu kroz sam model – npr ako imas tabelu sa 3 kolone, onda neces imati 4 atributa :)

Za sprovodjenje referential integrity koriste se candidate keys i foreign keys. Na osnovu candidate key mozes uniquely identifikovati record u tabeli npr. Tipicno je da se jedan od candidate keys koristi kao primary key.   
U slucaju ljudi to ce biti JMBG npr, a mozes i “id”. Svi drugi candidate keys pored tvog primary key-a se nazivaju alternate keys.

Foreign keys are used to enforce referential integrity. Definise se nad jedan ili vise atributa u relaciji i referencira candidate key u toj relaciji ili nekoj drugoj relaciji.

### Normalization

Relacioni model poseduje pravila normalizacije, kao i u matematici. Naime, sluzi da se smanji redundantnost podataka.

Postoje x3 nominalne forme - 1NF, 2NF, 3NF

1st nominal form(1NF) kaze da svaki record(tuple) u tabeli(relaciji) mora biti unique i da svaka kolona(atribut) mora biti atomican.

2nd nominal form(2NF) ima dva pravila. Prvo kaze da podaci moraju da zadrze njihov prvu normalnu formu. Drugo je vise vezano za kljuceve(candidate & foreign).

Svaki atribut u relaciji sa candidate key, teba da bude funkcionalno zavisan od tog candidate key-a. To znaci da atribut ne moze da bude parcijalno zavisan od candidate key-a.

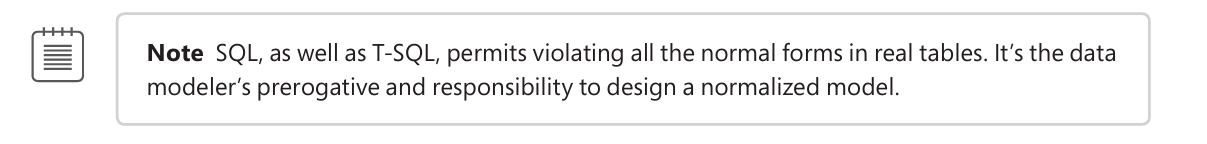
To put it more informally, if you need to obtain any nonkey attribute value, you need to provide the values of all attributes of a candidate key from the same tuple

A screenshot of a computer

AI-generated content may be incorrect.

TLDR ,ne mozes da imas dva primary key-a gde ces na osnovu jednog ili na osnovu drugog raditi pretrage.

3rd nominal form(3NF) takodje ima dva pravila. Prvo je da mora se ispostovati 2nd nominal form. Drugo je da svi non-key atributi moraju biti individualni i nezavisni jedni od drugih(must be dependent on candidate keys nontransitively.)



## Types of database workloads

Dva najcesca workloada su za OLTP(Online transaction processing) i DW(data warehousing).

Za DW to mozes sa AzureSQL/SqlServer koji koristi SMP(symmetric multiprocessing) architecture ILI za malo zahtevnije workloade Synapse Azure Analytics koji koristi MPP(Massive parallel processing)

A diagram of a diagram of a database

AI-generated content may be incorrect.

Here’s a quick description of what each acronym represents:

■ OLTP: online transactional processing   
■ DSA: data-staging area   
■ DW: data warehouse   
■ ETL / ELT: extract, transform, and load; or extract, load, and transform

### OLTP – Online Transaction Processing

Relacioni model jeste primarno “optimizovan” za OLTP workloads koji predstavljaju neki bazicni CRUD kada ti imas online transakcije koje na taj nacin rade sa podacima, znaci primarni cilj ti je data entry bez da imas neki reporting.

Ovde su tabele cesto i normalizovane, ali upravo zbog te normalizacije ovaj use case ne odgovara reportingu jer u OLTP mi cemo imati puno entiteta sa kompleksnim relationshipima.

Ako bi to pokusali na reporting scenariu, mi cemo imati mnogo kompleksnih queryja sa losim performansom.

### DW – Data Warehousing

Ovo je okruzenje koje je optimizovano za pribavljanje podataka i reporting purposes. Ukoliko opsluzuje celu organizaciju onda se to naziva data warehouse. Ako opsluzuje samo departman unutar organizacije ili jedan mali specifican context onda se naziva data mart.

U ovim slucajevima model podataka je optimizovan primarno za retrieval needs.

The model has intentional redundancy, fewer tables, and simpler relationships, ultimately resulting in simpler and more efficient queries than an OLTP environment.

Najjednostavniji data warehouse dizajn se naziva **star schema**. Ona ukljucuje nekoliko dimenzija schema i fact table(tabelu cinjenica). Svaka dimenzija predstavlja jednu temu na osnovu koje zelis da radis reportove – na primer sistem koji radi sa “orders & sales” verovatno ce zahtevati sledece dimenzije: customers, products, employees & time.

Svaka dimenzija je implementirana kao jedna tabela sa redundantnim podacima.

single ProductDim(product dimension) table instead of three normal ized tables: Products, ProductSubCategories, and ProductCategories

Ukoliko uzmes i normalizujes tabelu dimenzije(u vise tabela jelte), onda dobijas nesto sto se naziva **snowflake dimension**. Takva schema koja podrzava normalizovane dimenzije se naziva **snowflake schema**(znaci imas jedan ili vise snowflake, ne mora sve u snowflake).

**Fact table** tj tabela cinjenica sadrzi “facts and measures”(cinjenice i mere) poput kolicina, vrednost za svaku relevantnu kombinaciju kljuceva dimenzija.

Npr za svaku relevantnu kombinaciju customer, product, employee & day – tabela cinjenica ce imati red koji sadrzi quantity & value.

Note – podaci u dimenzijama su cesto “pre agregirani” za razliku od OLTP podataka koji su skladisteni cesto “po transakciji”.

### ETL/ELT

Proces koji povlaci podatke sa izvornih sistema(OLTP & drugi), manipulise im i ucitava ih u datat warehouse se naziva **Extract, transform & load ETL.**

Neki sistemi prvo povlace podatke sa izvornih sistema, ucitavaju ih u data warehouse i na kraju ih transformisu se nazivaju **Extract Load Transform – ELT**

**Microsoft provides an on-premises tool called Microsoft SQL Server Inte gration Services (SSIS) to handle ETL/ELT needs which comes with SQL Server license.**

Microsoft also provides a serverless cloud service for ETL/ELT solutions called **Azure Data Factory**

Cesto procesi koji vrse OLTP -> DW i koji sadrze neki ETL/ELT posao, imaju i DSA**(Data-Staging-Area)** koja moze biti u nekoj npr relacionoj bazi i sluzi kao data cleansing area. Zona gde ce se podaci “preciscavati”.

## SQL Server Architecture

### On-Prem & Cloud flavors of Sql Server.

Inicijalno je tu bio samo jedan RDBMS – Microsoft Sql Server koji je sluzio za on premises baze. Danas unutar svoje “data platforme”, MSFT nudi onprem, box solutions i service based cloud solutions. Mnogo vise :)

### On Premises

Ova nijansa RDBMS-a se naziva SQL Server koji je najtradicionalnija verzija sto msft nudi. Ovde customer vodi racuna o svemu – hardver, patchovi, disaster recovery, data retention i slicno.

Customer moze da instalira i vise instanci SQL servera na istom backendu i radi sta hoce u sustini. Max kontrola.

### Cloud

Cloud computing varijanta nudi cloud compute & cloud storage opcije “po zahtevu” sa deljenog “pool of resources”. MSFT RDBMS mogu biti ponudjene i sa public cloud i private cloud varijanti.

Za private cloud infrastrukturu se misli na cloud infrastrukturu koju jedna organizacija koristi putem tehnologije virtuelizacije. Cesto je hostovana lokalno.

Za public cloud infrastrukturu – ovde microsoft sve resava za tebe tako sto svi servisi public cloud infrastrukture su ti dostupni preko “network” tj mreze.

Ovde imamo x2 opcije, a to su IaaS i PaaS.

**IaaS** je kada hostujes SQL Server na VM. Ovde je na tebi da podesavas sve osim underlying hardware, jel tako. It’s essentially like maintaining your own SQL Server installation—one that happens to reside on Microsoft’s hardware.

**PaaS** je ukoliko koristimo microsoft managed sql server instance. Hardware, software installation and maintenance, high availability and disaster recovery, and patching are all responsibilities of Microsoft.   
Customer jedino o cemu treba da brine je index & query tuning.

### SQL Server Instances

U sustini ovo su instalacije SQL Servera na jednom on prem servisu. Svaka instalacija(proces) ne deli nista drugo izmedju sebe osim hardvera. Kompletno je nezavisna.

A diagram of a server

AI-generated content may be incorrect.

Kada imas vise instanci na istom serveru, samo jedna instanca moze biti default instanca. Sve ostale moraju biti named instances. O tome koja instanca je default, odlucuje se na kreiranju same instance i ne moze se kasnije promeniti.

Da bi se client povezao na instancu:  
**(default)** – **{pc\_ip/pc\_name} (Server1)**  
**(named\_instance)** – **{pc\_ip/pc\_name}\{instance\_name} (Server1\Inst2)**

### Databases