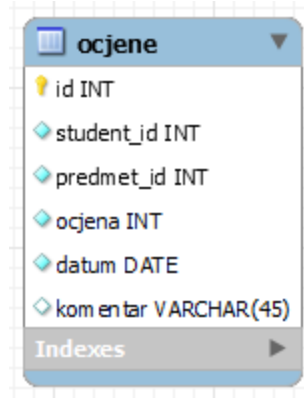


Predavanje 2-3

Agregacijske funkcije

Ukoliko postoji potreba primjenjivati dodatne operacije nad skupom podataka, tj. kolonama, to se može učiniti koristeći tzv. agregacijske funkcije. Ove funkcije pružaju mogućnost dodatne manipulacije podacima prije njihovog ispisa tj. povrata klijentskoj strani. Ove funkcije se primjenjuju nad grupisanim skupom izlaznog skupa podataka. Već je predstavljena jedna takva funkcija, `COUNT`, koja prebrojava redove. Kao primjer, neka je tabela *ocjene* tabela sa kojom se bilježe ocjene studenata na određenim predmetima.



The screenshot shows the structure of a table named 'ocjene'. It lists the following columns: 'id' (INT), 'student_id' (INT), 'predmet_id' (INT), 'ocjena' (INT), 'datum' (DATE), and 'komentar' (VARCHAR(45)). There is an 'Indexes' section at the bottom with a right-pointing arrow.

Column	Type
id	INT
student_id	INT
predmet_id	INT
ocjena	INT
datum	DATE
komentar	VARCHAR(45)

```
SELECT COUNT(*) FROM ocjene;
```

Ovo znači, prebroj sve redove iz tabele *ocjene*. To je kao da smo rekli `SELECT * FROM ocjene`; i ručno prebrojali redove izlazne tabele. Slično, moglo se reći:

```
SELECT COUNT(komentar) FROM ocjene;
```

U ovom slučaju, sistem će vratiti broj redova gdje je polje komentar popunjeno. Tj. moguće je dobiti drugačiji odgovor od `COUNT(*)` jer je polje *komentar* opcionalno i stoga može biti prazno.

Od ostalih agregacijskih funkcija, vrijedi spomenuti `SUM`, `MIN`, `MAX` i `AVG`. `MIN` i `MAX` daju minimalnu i maksimalnu vrijednost neke grupe, respektivno. Dok `AVG` vraća prosječnu vrijednost. Npr.

```
SELECT MIN(ocjena) FROM ocjene;  
SELECT MAX(ocjena) FROM ocjene;
```

će vratiti minimalnu i maksimalnu vrijednost ocjene. Pošto ove komande vraćaju samo jednu kolonu, sa vrijednosti najniže ili najviše ocjene, nije poznato kojem studentu ili predmetu pripadaju. Stoga je primamljivo napisati dodatnu kolonu, međutim ovakav upit će dati netačne rezultate:

```
SELECT student_id, MIN(ocjena) AS najmanja_ocjena FROM ocjene;  
SELECT student_id, MAX(ocjena) AS najvisa_ocjena FROM ocjene;
```

`MIN` i `MAX` se mogu primjenjivati i na nenumeričke kolone. Tako npr. ako je potrebno vratiti prvu ocjenu u tabeli po datumu, možemo primjeniti `MIN` operaciju nad kolonom *datum*:

```
SELECT MIN(datum) AS prva_ocjena FROM ocjene;
```

Ukoliko želimo koristiti više agregacijskih funkcija istovremeno, to i možemo uraditi. Npr. selektovanje najniže i najviše ocjene se može uraditi na sljedeći način:

```
SELECT MAX(ocjena), MIN(ocjena) FROM ocjene;
```

Međutim ako hoćemo da dobavimo povezanog studenta za te ocjene, ne možemo jednostavno napisati:

```
SELECT student_id, MAX(ocjena), MIN(ocjena) FROM ocjene;
```

Ovaj upit nema smisla jer najviša ocjena može biti od jednog a najmanja od drugog studenta. Prethodni upit je imao smisla jer se i najviša i najmanja biraju iz cijele tabele. I u ovom slučaju će to biti slučaj, međutim vrijednost *student_id* je nepouzdana i neće biti tačna.

Ukoliko želimo da koristimo agregacijske funkcije, zajedno sa drugim kolonama u tabeli, moramo koristiti grupisanje. Koja informacija se željela izvući u prethodnom upitu? Ispis svih najnižih i najviših ocjena po studentu. Kada se ovo predstavi na ovaj način, može se primjetiti da izlazna tabela neće imati jedan red već onoliko redova koliko ima studenata, jer će svaki student imati najmanju i najvišu ocjenu. Da postignemo ovaj rezultat, koristi se klauzula **GROUP BY**.

```
SELECT student_id, MAX(ocjena), MIN(ocjena)
FROM ocjene
GROUP BY student_id;
```

Ovo u prevodu znači, dobavi sve najmanje i najviše ocjene po studentu. To je isto kao da postoji N tabela, po jedna za svakog studenta, i da se za svaku od njih odjednom pretražuje **MIN** i **MAX**.

Još jedna agregacijska funkcija je **SUM**. Ova funkcija primjenjuje zbir na kolone numeričkog tipa. **NULL** i nenumeričke vrijednosti su ignorisane.

```
SELECT SUM(ocjena) FROM ocjene;
```

Unutar **SELECT** klauzule, možemo koristiti i aritmetičke operacije, poput dijeljenja. Da se dobije prosječna vrijednost ocjene, to može se uraditi na sljedeći način:

```
SELECT SUM(ocjena)/COUNT(ocjena) AS prosjecna_ocjena FROM ocjene;
```

Međutim da se ovo ne bi pisalo svaki put jer postoji funkcija pod nazivom **AVG**, koja vraća prosječnu vrijednost:

```
SELECT AVG(ocjena) AS prosjecna_ocjena FROM ocjene;
```

ORDER BY

Kada se izlistavaju podaci koristeći upite, korisno je te podatke poredati po rastućem ili opadajućem poretku. Da bi se ovo postiglo, koristi se klauzula **ORDER BY**. Npr.

```
SELECT * FROM ocjene ORDER BY datum ASC;
```

će izlistati sve redove iz tabele ocjene poredane od najstarijeg do najnovijeg datuma. **ASC** je skraćeno od *ascending*. Ovo je opcionalna ključna riječ i nije je neophodno napisati za uzlazno sortiranje. Slično, ukoliko je potrebno sortirati po datumu od najnovijeg ka najstarijem, to se može uraditi ovako:

```
SELECT * FROM ocjene ORDER BY datum DESC;
```

Ova klauzula se može primjenjivati na sve tipove kolona, uključujući i agregacijske kolone. Na primjeru tabele *studenti* recimo da je potrebno vratiti sva jedinstvena imena sortirana od najčešćeg imena do najrjeđeg imena:

```
SELECT COUNT(*) as broj, ime
FROM studenti
GROUP BY ime
ORDER BY broj DESC;
```

Ovaj upit će prebrojati sva imena iz tabele *studenti*, tj. pronaći broj redova za svako ime koja su međusobno jednaka, a po tome će izlaz sortirati po broju imena. Slično, ukoliko je potrebno sortirati od najrjeđeg do najčešćeg:

```
SELECT COUNT(*) as broj, ime
FROM studenti
GROUP BY ime
ORDER BY broj;
```

Potrebno je još napomenuti da je moguće sortirati po više kolona i to u različitim smjerovima. Recimo da tabela *studenti* posjeduje kolonu *grad* koja predstavlja grad iz kojega student dolazi. Ukoliko je potrebno dobiti sve studente, a sortirati abecedno grad iz kojeg dolaze ali suprotno abecedno po prezimenu, to bi se postiglo na sljedeći način:

```
SELECT * FROM studenti ORDER BY grad ASC, prezime DESC;
```

Na ovaj način je moguće manipulirati sortiranjem izlazne tabele, i sortiranje se odvija kao krajnji korak u nizu SQL operacija navedenih u upitu.