Kolegij:

Baze podataka I.

Projekt:

Sustav za upravljanje knjižnicom

Autor:

Petra Tuškan

JMBAG 0058206931

Uvod

Kako bi odgovorili na sve veća očekivanja korisnika u pogledu uvida u kvalitetu knjiga, stanje njihovih primjeraka te pravovremeno obračunavanje kazni za kašnjenja, u bazu su uvedeni podaci o recenzijama knjiga, statusima primjeraka i zakašnjelim vraćanjima.

Cilj mog modula

Dizajn i implementacija tablica: Recenzije, Primjeri te KasnoVracanje

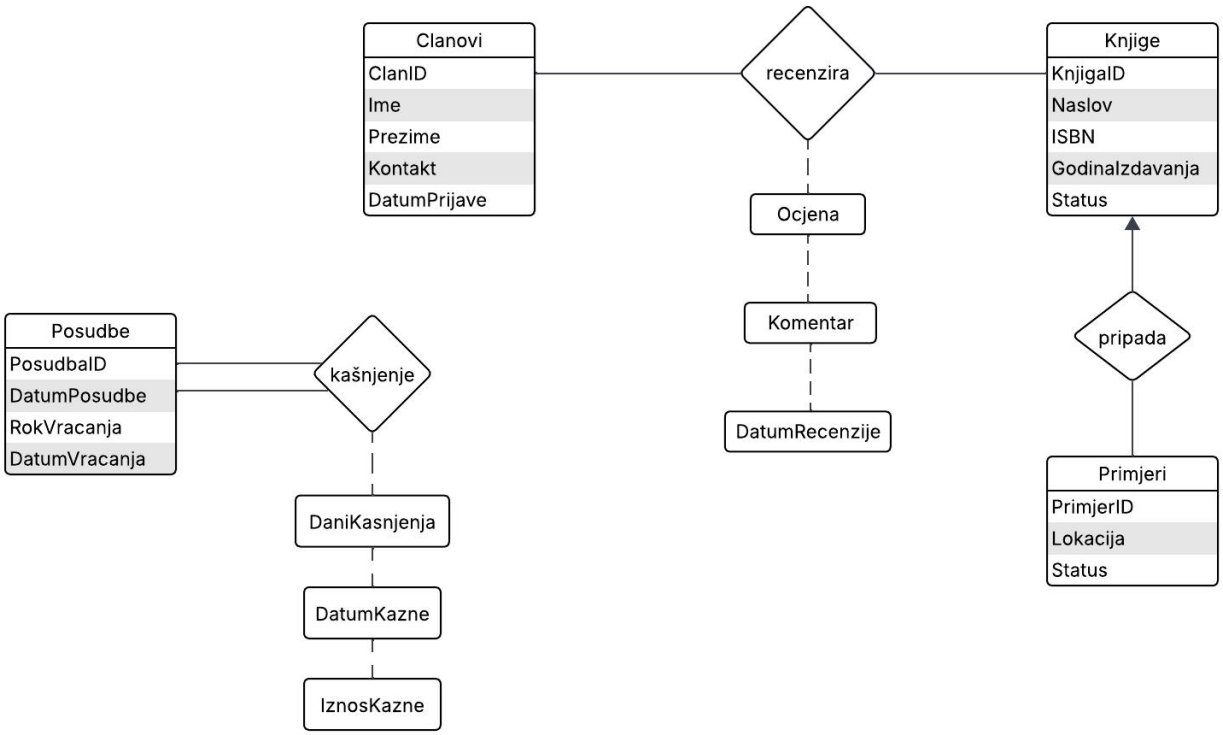
Omogućeno je:

- procjena zadovoljstva članova knjižnice putem sustava za recenziranje knjiga
- uvid u fizičko stanje i dostupnost svake knjige
- praćenje kašnjenja i pripadajućih kazni

Tehnologije

- MySQL
- MySQL Workbench
- Git i GitHub
- Microsoft Word

ER dijagram



Relacijski model:

Clanovi (ClanID, Ime, Prezime, Kontakt, DatumPrijave)

Knjige (KnjigaID, Naslov, ISBN, Status, GodinaIzdavanja)

Primjeri (PrimjerID, KnjigaID, Lokacija, Status)

Posudbe (PosudbaID, KasnoVracanjeID, DaniKasnjenja, DatumKazne, IznosKazne, DatumPosudbe, RokVracanja, DatumVracanja)

recenzira (ClanID, KnjigaID, Ocjena, Komentar, DatumRecenzije)

Opis tablica, atributa i domena

Tablica Recenzije

Atribut	Domena/Ograničenja	Komentar
RecenzijaID	INT PK, AUTO_INCREMENT, NOT NULL	Id recenzije
KnjigaID	INT FK NOT NULL	Id knjige za koju je ostavljena recenzija
ClanID	INT FK NOT NULL	Id člana koji je ostavio recenziju
Ocjena	INT CHECK (Ocjena BETWEEN 1 AND 5)	Ocjena recenzije sa vrijednošću između 1 I 5
Komentar	TEXT	Tekstualni komentar recenzije
DatumRecenzije	DATE	Datum kada je recenzija ostavljena

Tablica Primjeri

Atribut	Domena/Ograničenja	Komentar
PrimjerID	INT PK, AUTO_INCREMENT, NOT NULL	Id primjerka knjige
KnjigaID	INT FK NOT NULL	Id knjige kojoj primjerak pripada
Lokacija	VARCHAR(255)	Lokacija unutar knjižnice gdje se primjerak knjige nalazi
Status	VARCHAR(50)	Opisuje dostupnost pojedinog primjerka

Tablica KasnoVracanje

Atribut	Domena/Ograničenja	Komentar
KasnoVracanjeID	INT PK, AUTO_INCREMENT, NOT NULL	Id pojedinog zapisa o kašnjenju
PosudbaID	INT FK NOT NULL	Id posudbe na koju se odnosi kasno vraćanje
DaniKasnjenja	INT NOT NULL	Broj dana kašnjenja
Iznos Kazne	DECIMAL(10,2) NOT NULL	Novčani iznos kazne

Upiti

Članovi koji su ostavili barem jednu recenziju poredano po broju recenzija i prezimenu člana

```
CREATE VIEW clanovi_recenzije AS
SELECT
      c.ClanID,
      c.Ime,
      c.Prezime,
      COUNT(r.ClanID) AS broj_recenzija
FROM
      Clanovi c
INNER JOIN
      Recenzije r ON c.ClanID = r.ClanID
GROUP BY
      c.ClanID, c.Ime, c.Prezime
HAVING
      COUNT(r.ClanID) >= 1
ORDER BY
      broj_recenzija DESC,
      c.Prezime ASC;
```

Detalji:

- pomoću naredbe SELECT odabiremo ID člana knjižnice te njegovo ime i prezime iz tablice Clanovi (alias c), a pomoću COUNT agregatne funkcije brojimo recenzije koje je pojedini član ostavio
- INNER JOIN služi za spajanje tablice Recenzije s tablicom Clanovi preko zajedničkom stupca ClanID. Na taj način se osigurava da se u rezultat uključe isključivo oni članovi koji su napisali barem jednu recenziju
- GROUP BY agregatnu funkciju moramo koristiti nakon korištenja agregatne funkcije COUNT. Ona služi za grupiranje podataka po c.ClanID, c.Ime i c.Prezime stupcima
- HAVING naredba se koristi nakon grupiranja i služi za filtriranje rezultata tako da ostaju samo oni za koje je broj recenzija minimalno 1

- iako je u ovom konkretnom slučaju HAVING suvišan (jer s obzirom na spajanje s tablicom Recenzije, svaki član automatski ima minimalno jednu recenziju), ipak će biti ostavljen jer bi u budućnosti moglo doći do nekih promjena u uvjetima
- ORDER BY sortira rezultate na način da se prvo sortiraju po broju recenzija u silaznom redoslijedu, a ako više članova ima isti broj recenzija, tada se oni dodatno sortiraju abecednim redoslijedom prema prezimenu

Primjer korištenja u praksi:

Knjižnica želi potaknuti članove na aktivnije ostavljanje recenzija s ciljem stvaranja korisničkih preporuka i povratnih informacija o knjigama.

Jedan od načina da se to postigne je organizacija nagradne igre u kojoj sudjeluju svi članovi koji su ostavili barem jednu recenziju.

Knjige s barem jednim kašnjenjem čiji je prosjek ocjena manji od 3

```
CREATE VIEW kasnjenja_niske_ocjene AS
SELECT
      k.KnjigaID,
      k.Naslov,
      AVG(r.Ocjena) AS prosjecna_ocjena,
      COUNT(DISTINCT kv.KasnoVracanjeID) AS broj_kasnjenja
FROM
      Knjige k
INNER JOIN Posudbe p ON k.KnjigaID = p.KnjigaID
INNER JOIN KasnoVracanje kv ON p.PosudbaID = kv.PosudbaID
INNER JOIN Recenzije r ON k.KnjigaID = r.KnjigaID
GROUP BY
    k.KnjigaID, k.Naslov
HAVING
    COUNT(DISTINCT kv.KasnoVracanjeID) >= 1
    AND AVG(r.Ocjena) < 3;
```

Detalji:

- pomoću naredbe SELECT odabiremo ID i naslov knjige iz tablice Knjige (alias k). Također, pomoću agregatne funkcije AVG izračunavamo prosječnu ocjenu koju je knjiga dobila (iz tablice Recenzije), dok se COUNT(DISTINCT kv.KasnoVracanjeID) koristi za brojanje koliko je puta neka knjiga vraćena sa zakašnjenjem
- Tri INNER JOIN naredbe spajaju 4 tablice:
  - tablicu Knjige s tablicom Posudbe preko zajedničkog stupca KnjigaID kako bi se dobile sve posudbe povezane s knjigama
  - tablicu Posudbe s tablicom KasnoVracanje preko zajedničkog stupca PosudbaID kako bi se pronašla kašnjenja u vraćanju
  - tablicu Knjige s tablicom Recenzije putem stupca KnjigaID kako bi se prikupile sve ocjene vezane za pojedinu knjigu

- GROUP BY osigurava da rezultat sadrži agregirane vrijednosti (broj kašnjenja, prosjek ocjena) za svaki jedinstveni naslov knjige
- HAVING filtrira rezultate tako da prikazuje samo one knjige koje:
  - imaju barem jedno zakašnjelo vraćanje
  - imaju prosječnu ocjenu recenzije manju od 3
- iako i u ovom slučaju možemo pretpostaviti da će spajanje s tablicom KasnoVracanje automatski ukloniti knjige bez kašnjenja, uvjet je ipak naveden zbog bolje čitljivosti i eventualne otpornosti na buduće izmjene

Primjer korištenja u praksi:

Ovaj upit pomaže knjižnici u identifikaciji "problematičnih" knjiga.

Knjižnica želi analizirati koje knjige nisu dobro prihvaćene među korisnicima, a uz to ih korisnici i često vraćaju sa zakašnjenjem. Te informacije mogu poslužiti kao temelj za donošenje odluka o uklanjaju pojedinih naslova iz fonda, njihovoj zamjeni boljim izdanjima, ili eventualno marketinškim kampanjama za promociju "nepopularnih" knjiga.

Naslovi knjiga sortirani po pripadajućim žanrovima i prosječnim ocjenama

```
CREATE VIEW Knjige_po_ocjenama_i_zanrovima AS
SELECT
      k.KnjigaID AS Knjiga_ID,
      k.Naslov AS Naslov_knjige,
      kat.Naziv AS Zanr,
      AVG(r.Ocjena) AS Prosjecna_ocjena
FROM
    Knjige k
INNER JOIN Kategorije kat ON k.KategorijaID = kat.KategorijaID
LEFT JOIN Recenzije r ON k.KnjigaID = r.KnjigaID
GROUP BY
      k.KnjigaID, k.Naslov, kat.Naziv
ORDER BY
    kat.Naziv ASC,
    Prosjecna_ocjena DESC;
```

Detalji:

- pomoću naredbe SELECT odabiremo naslov i ID knjige iz tablice Knjige (alias k), naziv žanra iz tablice Kategorije (alias kat), te prosječnu ocjenu koju je knjiga dobila, koristeći agregatnu funkciju AVG() nad ocjenama iz tablice Recenzije
- INNER JOIN se koristi za povezivanje tablica Knjige i Kategorije preko stupca KategorijaID. Time dohvaćamo podatak kojem žanru pripada pojedina knjiga (budući da svaka knjiga mora imati žanr, koristimo unutarnje spajanje)
- LEFT JOIN spaja tablicu Recenzije s tablicom Knjige prema stupcu KnjigaID. Ovaj tip spajanja omogućuje da u rezultat budu uključene i one knjige koje još nemaju nijednu recenziju (njihova prosječna ocjena će biti NULL)
- GROUP BY koristimo nakon korištenja agregatne funkcije AVG() za grupiranje rezultata po ID-u knjige, naslovu i žanru

- ORDER BY sortira rezultate prvo abecedno po nazivu žanra, a zatim po prosječnoj ocjeni unutar svakog žanra. Na taj način se unutar svakog žanra prvo prikazuju najbolje ocijenjene knjige

Primjer korištenja u praksi:

Knjižnica želi pratiti koje knjige su najpopularnije unutar svakog žanra, kako bi korisnicima mogla preporučiti naslove prema njihovim interesima.

Knjige sa statusom "dostupno" poredane po imenu autora

```
CREATE VIEW dostupne_knjige AS
SELECT
      k.KnjigaID,
      k.Naslov,
      k.Status,
      a.ImePrezime AS Autor
FROM
      Knjige k,
      KnjigaAutori ka,
      Autori a
WHERE
      k.KnjigaID = ka.KnjigaID
      AND ka.AutorID = a.AutorID
      AND k.Status = 'dostupno'
ORDER BY
      a.ImePrezime ASC;
```

Detalji:

- pomoću SELECT naredbe dohvaćamo ID knjige, njen naslov i status iz tablice Knjige (alias k) te dohvaćamo ime i prezime autora iz tablice Autori (alias a)
- u ovom upitu umjesto INNER JOIN naredbe koristimo drugačiju sintaksu gdje se sve tablice navode u FROM dijelu, a uvjeti spajanja pišu se u WHERE dijelu
- pomoću WHERE uvjeta povezujemo tablicu Knjige s tablicom KnjigaAutori preko KnjigaID, te povezujemo tablicu Knjiga Autori s tablicom Autori preko AutorID kako bismo dohvatili podatke o autorima povezanim s knjigama.
- uvjet k.Status = 'dostupno' osigurava da u rezultat budu uključene samo knjige dostupne za posudbu
- ORDER BY sortira konačni rezultat prema imenu i prezimenu autora u abecednom redoslijedu

Primjer korištenja u praksi:

Knjižnica želi brzo generirati popis svih trenutno dostupnih knjiga organiziran po autorima.

Na taj način pomaže u usluzi korisnicima na info pultu ili web katalogu te boljoj organizaciji fonda knjižnice.

Knjige sa oštećenim primjercima poredanih po broju oštećenih primjeraka i imenu autora

```
CREATE VIEW popis_ostecenih_knjiga AS
SELECT
      k.KnjigaID,
      k.Naslov,
      a.ImePrezime AS Autor,
      COUNT(p.PrimjerID) AS broj_ostecenih
FROM
      Knjige k
INNER JOIN
      Primjeri p ON k.KnjigaID = p.KnjigaID
INNER JOIN
      KnjigaAutori ka ON k.KnjigaID = ka.KnjigaID
INNER JOIN
      Autori a ON ka.AutorID = a.AutorID
WHERE
      p.Status = 'Oštećeno'
GROUP BY
      k.KnjigaID, k.Naslov, a.ImePrezime
ORDER BY
      broj_ostecenih DESC,
      a.ImePrezime ASC;
```

Detalji:

- SELECT naredbom odabiremo ID i naslov knjige iz tablice Knjige (alias k), ime i prezime autora iz tablice Autori (alias a), te broj oštećenih primjeraka koji se računa pomoću agregatne funkcije COUNT() iz tablice Primjeri
- INNER JOIN između tablica Knjige i Primjeri preko stupca KnjigaID omogućuje pristup svim primjercima određene knjige
- INNER JOIN s tablicom KnjigaAutori povezuje knjige s njihovim autorima putem many-to-many veze. Ova tablica omogućuje da knjiga ima više autora i obratno
- INNER JOIN s tablicom Autori omogućuje dohvat imena i prezimena autora

- WHERE uvjet filtrira samo one primjerke knjiga koji imaju status 'oštećeno'
- GROUP BY se koristi nakon COUNT() funkcije. Svi oštećeni primjerci iste knjige i autora se zbrajaju
- na kraju ORDER BY prvo sortira rezultat po broju oštećenih primjeraka u silaznom redoslijedu, a zatim sortira abecedno po imenu autora (u slučaju da više knjiga ima isti broj oštećenih primjeraka)

Primjer korištenja u praksi:

Knjižnica koristi ovaj pogled kako bi identificirala knjige koje su fizički najviše oštećene. Na temelju rezultata može planirati obnavljanje knjižnog fonda.

Također se mogu raditi neke analize kao što su praćenje oštećenja po nakladnicima ili popularnosti autora.

Knjige s oštećenim primjercima i prosječnom ocjenom nižom od 3

```
CREATE VIEW ostecene_knjige_s_losim_recenzijama AS
SELECT DISTINCT
      k.KnjigaID,
      k.Naslov,
      p.Status,
      AVG(r.Ocjena) AS prosjecna_ocjena
FROM
      Knjige k,
      Primjeri p,
      Recenzije r
WHERE
      k.KnjigaID = p.KnjigaID
      AND k.KnjigaID = r.KnjigaID
      AND p.Status = 'Oštećeno'
GROUP BY
      k.KnjigaID, k.Naslov, p.Status
HAVING
      AVG(r.Ocjena) < 3
ORDER BY
      prosjecna_ocjena ASC;
```

Detalji:

- pomoću naredbe SELECT dohvaćamo ID i naslov knjige iz tablice Knjige (alias k), status primjerka iz tablice Primjeri (alias p) te prosječnu ocjenu koju je knjiga dobila, koristeći agregatnu funkciju AVG() nad ocjenama iz tablice Recenzije (alias r)
- DISTINCT unutar SELECT naredbe služi za uklanjanje potencijalnih duplikata
- FROM dio upita povezuje tablice Knjige, Primjeri i Recenzije, koristeći uvjete u WHERE klauzuli. Knjiga se povezuje i s primjercima i s recenzijama preko KnjigaID
- uvjet p.Status = 'Oštećeno' unutar WHERE filtrira primjerke knjiga koji su oštećeni
- GROUP BY se koristi nakon AVG() te grupira rezultat po naslovu i statusu primjerka

- HAVING dodatno filtrira oštećene rezultate tako da u konačni rezultat ulaze samo one knjige koje imaju prosječnu ocjenu manju od 3
- na kraju ORDER BY sortira oštećene knjige tako da se prvo prikažu knjige s najnižom prosječnom ocjenom

Primjer korištenja u praksi:

Knjižnica koristi ovaj pogled za redovitu reviziju svog knjižnog fonda kako bi se osiguralo da su naslovi u ponudi fizički ispravni i zanimljivi korisnicima.

Ovo je jedan od načina kako da se objektivno prepoznaju knjige koje potencijalno nebi trebale biti dio fonda.

Zaključak

Kroz ovaj modul razvijen je dio baze podataka koji knjižnici donosi funkcionalnosti vezane uz recenzije knjiga, stanje njihovih primjeraka i praćenje kašnjenja pri vraćanju. Time je omogućeno da knjižnica bolje razumije što članovi vole čitati, koje knjige često kasne s povratom i u kakvom su stanju primjerci u fondu.

Pomoću SQL upita stvoreni su pogledi koji pomažu pri svakodnevnom radu – od otkrivanja loše ocijenjenih i oštećenih knjiga, do preporuka po žanrovima i uvida u aktivnost članova. Sve to doprinosi lakšoj organizaciji i boljim uslugama prema korisnicima.