ТЕОРИЈА *[25 поена]*

ТРАНСАКЦИЈЕ

1. Шта најбоље описује трансакцију у бази података?

А. Скуп независних SQL наредби које се извршавају произвољним редоследом, без међусобне повезаности.
Б. Логичка целина више операција над подацима која треба да буде извршена као једна недељива јединица.
Ц. Механизам за аутоматско индексирање свих табела које учествују у обради података.
Д. Поступак којим се свака SQL наредба аутоматски чува у посебној табели историје.

2. Која тврдња најтачније објашњава особину атомичности?

А. Трансакција може делимично да успе, а делимично да буде враћена ако је то ефикасније за систем.
Б. Трансакција се извршава постепено, при чему се свака успешна наредба трајно чува без обзира на остатак.
Ц. Трансакција је успешна само ако су сви њени кораци извршени, у супротном се систем враћа на претходно стање.
Д. Трансакција је атомична само ако садржи један UPDATE упит и један COMMIT.

3. Шта представља евентуална конзистентност у системима са репликацијом?

А. Све реплике у сваком тренутку морају имати потпуно исте податке пре него што се дозволи било које читање.
Б. Подаци могу привремено бити различити на репликама, али ће се након одређеног времена ускладити.
Ц. Конзистентност значи да се измене никада не шаљу са master базе на реплике док се server не рестартује.
Д. Евентуална конзистентност подразумева да се подаци синхронизују само када корисник ручно покрене освежавање.

4. Која тврдња најпрецизније описује улогу WAL механизма?

А. Промене се прво трајно уписују у data фајлове, па се затим по потреби накнадно уписују у log.
Б. Log запис служи само за евидентирање неуспешних трансакција, док успешне не морају бити записане.
Ц. Свака промена се прво записује у log на диску, па се тек онда сматра трајно потврђеном.
Д. WAL омогућава да се COMMIT прескочи јер log аутоматски потврђује сваку SQL наредбу.

5. Који изолациони ниво дозвољава да трансакција види само потврђене измене, али и даље може да прочита различиту вредност истог реда при поновном читању?

А. Read Uncommitted
Б. Read Committed
Ц. Repeatable Read
Д. Serializable

6. Који проблем настаје када трансакција два пута изврши исти упит, а у другом резултату види нове редове које је друга трансакција у међувремену убацила?

А. Dirty Read
Б. Lost Update
Ц. Phantom Read
Д. Deadlock

7. Који SQL упит представља атомско ажурирање које може помоћи у спречавању lost update проблема?

А. SELECT kolicina FROM inventar WHERE id = 1;
Б. UPDATE inventar SET kolicina = 9 WHERE id = 1;

Ц. UPDATE inventar SET kolicina = kolicina - 1 WHERE id = 1;
Д. INSERT INTO inventar VALUES (1, 9);

8. Који изолациони ниво најбоље спречава и dirty read, и non-repeatable read, и phantom read, али може успорити систем?

А. Read Uncommitted
Б. Read Committed
Ц. Repeatable Read
Д. Serializable

ИНДЕКСИ

9. Када је индекс најкориснији?

А. Када услов враћа велики проценат табеле и када се често користе функције над колонама
Б. Када услов враћа мали број редова и не користе се функције над индексираном колоном
Ц. Када табела нема много редова, без обзира на упит
Д. Када се подаци често уписују, а ретко читају

10. Шта представља селективност колоне?

А. Однос броја индексираних колона и броја свих табела у бази
Б. Број свих редова које база може да прочита из индекса у једној секунди
Ц. Однос броја враћених редова и укупног броја редова у табели
Д. Број различитих индекса који постоје над једном колоном

11. Зашто функција над индексираном колоном често „убија“ индекс?

А. Зато што функција аутоматски брише постојећи индекс из базе
Б. Зато што DBMS више не може директно да користи постојећи индекс над оригиналним вредностима колоне
Ц. Зато што се функције могу користити само над примарним кључем
Д. Зато што функција претвара индекс у hash индекс

12. Када се најчешће користи Index Seek?

А. Када треба прочитати скоро целу табелу
Б. Када је услов веома селективан и тражи се тачна или врло уска група вредности
Ц. Када нема индекса па optimizer бира најбржу опцију
Д. Када се врши сортирање над колоном која није индексирана

13. Шта значи Index Only Scan?

А. База користи само табелу, без коришћења индекса
Б. База користи индекс само за сортирање, али податке чита из табеле
Ц. База све потребне податке за упит проналази у индексу, без одласка у саму табелу
Д. База користи само cluster индекс, а никада non-cluster индекс

14. У ком случају ће optimizer највероватније изабрати Full Table Scan, иако индекс постоји?

А. Када услов враћа велики проценат редова и процени се да је скенирање целе табеле јефтиније
Б. Када је колона део примарног кључа
Ц. Када упит користи оператор = над јединственом вредношћу
Д. Када је индекс направљен над почетном колоном композитног индекса

15. Шта је основна сврха команде EXPLAIN?

А. Да изврши упит брже користећи све доступне индексе
Б. Да обрише неискоришћене индексе из табеле
Ц. Да прикаже како DBMS планира да изврши упит, без самог извршавања упита

Д. Да аутоматски креира индекс који недостаје упиту

ПЛАН ИЗВРШЕЊА УПИТА

16. Шта представља логички план извршења упита?

А. Конкретан план који садржи index seek, hash join и sort оператор
Б. План који описује које операције треба извршити, без детаља о физичкој имплементацији
Ц. План који садржи само процену броја редова после JOIN-а
Д. План који приказује само редослед читања са диска

17. Шта је суштина оптимизације познате као predicate pushdown?

А. Да се JOIN изврши што раније, а филтери што касније
Б. Да се сортирање изврши пре филтрирања како би се убрзао JOIN
Ц. Да се филтери спусте што ближе табелама над којима важе, пре JOIN-а
Д. Да се пројекција колона одложи до самог краја физичког плана

18. Од чега директно зависи процена кардиналности у execution plan-у?

А. Од процене колико редова остаје после филтера и других операција
Б. Од броја индекса у целој бази
Ц. Од броја корисника који истовремено извршавају упит
Д. Од величине RAM меморије сервера, без обзира на упит

19. У ком случају ће оптимизатор највероватније изабрати Hash Join?

А. Када су обе табеле већ сортиране по join колони и постоји потреба за ≤ условом
Б. Када је једна табела врло мала, а над другом постоји одговарајући индекс
Ц. Када је JOIN заснован на једнакости, табеле су велике и нема одговарајућих индекса
Д. Када се спајање ради над опсезима вредности, а не над једнакостима

20. Зашто се у Hash Join алгоритму hash табела обично гради над мањом табелом?

А. Зато што мања табела обично садржи више различитих кључева
Б. Зато што мања табела лакше стаје у меморију и lookup постаје бржи
Ц. Зато што се само над мањом табелом може применити hash функција
Д. Зато што већа табела не може учествовати у probe фази

21. Који услов мора бити испуњен да би Merge Join био погодан избор?

А. Барем једна табела мора имати hash индекс над join колоном
Б. Једна табела мора бити веома мала, а друга без индекса
Ц. JOIN услов мора обавезно садржати BETWEEN оператор
Д. Обе табеле морају бити сортиране по истој join колони и истом редоследу

ОПТИМИЗАЦИЈА УПИТА

22. Која употреба wildcard оператора је најповољнија за индексирану string колону?

А. LIKE '%srt%'
Б. LIKE '%srt'
Ц. LIKE 'srt%'
Д. Све три имају исто понашање над индексом

23. Зашто је боље користити WHERE уместо HAVING када услов не зависи од агрегације?

А. Зато што HAVING може да се користи само са DISTINCT
Б. Зато што WHERE филтрира редове пре груписања, па се обрађује мањи скуп података
Ц. Зато што HAVING увек враћа погрешан број редова

Д. Зато што WHERE аутоматски прави индекс над колоном

24. Зашто је UNION ALL бржи од UNION?

А. Зато што UNION ALL аутоматски сортира резултате
Б. Зато што UNION ALL не проверава и не уклања дупликате
Ц. Зато што UNION ALL враћа само јединствене редове
Д. Зато што UNION не може да ради над две табеле

25. У ком случају је према презентацији погодније користити EXISTS уместо DISTINCT?

А. Када желимо да спојимо табеле у релацији 1:M и занима нас само постојање повезаног реда
Б. Када желимо да прикажемо све дупликате из обе табеле
Ц. Када желимо да сортирамо резултат по две колоне
Д. Када желимо да бројимо све редове из спољне табеле без услова

ЗАДАЦИ

Дате су следеће релације базе података KULTURA.

- **UMETNIK** (ID, IME, PREZIME, EMAIL, GOD_ROD, GRAD, HONORAR)
- **USTANOVA** (ID, NAZIV, ADRESA, GOD_OSNIVANJA, TIP)
- **ANGZAMAN** (*ID_UMETNIKA*, *ID_USTANOVE*, DATUM_OD, DATUM_DO, ULOGA)

Релација UMETNIK садржи: јединствени идентификатор, име, презиме, email, годину рођења, град и хонорар (уколико је плаћен).
У релацији USTANOVA чувају се: јединствени идентификатор, назив, адреса (подразумевана вредност је "Nis"), година оснивања и тип установе (нпр. "pozoriste", "muzej").
Релација ANGAZMAN бележи који уметник је ангажован у којој установи, од када (обавезно), до када (NULL ако је још активан) и која му је улога. Улога може бити: "gost", "stalni" или "saradnik". Могу постојати уметници без ангажмана.

Задатак 1. *[15 поена]*
За описану базу података написати следеће SQL упите:

a) *[7 поена]*
Приказати име, презиме и email свих уметника чије име почиње словом 'A', 'D' или 'J', а чији се email завршава доменом '.com' или '.rs'.

b) *[8 поена]*
Приказати назив установе и њен тип, али само за установе чији назив садржи реч од тачно 6 слова (нпр. "Centar", "GalerS") или чија адреса садржи макар једно мало слово 'x' или 'z' (нпр. "Knez 12z"). Приказати само оне установе које имају барем једног ангажованог уметника у улози сталног ("stalni") члана.

Задатак 2. *[15 поена]*
Креирати улогу (role) под називом administrator_kulture. Овој улози доделити привилегије за читање, измену и брисање података над табелама UMETNIK и ANGAZMAN. *[8 поена]*
Након тога, креирати корисника moderator_kult са лозинком "Kultura2026#", доделити му новокреирану улогу и поставити је као подразумевану. *[7 поена]*

Задатак 3. *[15 поена]*
Написати тригер regulate_honorar који, пре промене хонорара уметника, проверава да ли је нови хонорар већи од старог за **више** од 50%. Уколико јесте, тригер треба да не мења износ старог хонорара. *[7 поена (потпис тригера) + 8 поена (логика)]*

Задатак 4. *[30 поена]*
Написати ускладиштену процедуру povecaj_honorar_sa_izvestajem.
Процедура прихвата параметар id_ustanove_param и излазни параметар ukupna_isplata_povecanja. *[5 поена]*

За све уметнике који су тренутно ангажовани у задатој установи (проверити DATUM_DO), потребно је повећати хонорар за 12%.

У процедури је потребно:

a) *[5 поена]*
Користити локалну променљиву faktor_povecanja којој је додељена вредност 1.12,

b) *[7 поена]*

Израчунати укупан износ за колико су укупно порасли хонорари и уписати га у ukupna_isplata_povecanja,

c) *[7 поена]*
Ажурирати хонораре у табели UMETNIK,

d) *[6 поена]*
Користити сесијску променљиву @brojac_izmena која се увећава при сваком позиву процедуре како би се пратило колико је пута у току сесије вршена корекција хонорара.

ПРЕДМЕТНИ ПРОФЕСОР И ПРЕДМЕТНИ АСИСТЕНТ