1. Šta čini sistem realcione baze podataka?

Sistem baze podataka čini baza podataka, softver za skladištenje i upravljanje podacima u bazi podataka, aplikacioni programi koji koriste podatke i njihovi krajnji korisnici.

1. Koji je standardan scenario obavljanja operacije nad bazom podataka iz Java programa?
   1. Instalisanje JDBC uslužnih programa (drajvera)
   2. Uspostavljanje veze
   3. Kreiranje SQL iskaza
   4. Izvršavanje SQL iskaza
   5. Obrada ResultSet
2. Šta je JDBC?

JDBC je Java API koji se koristi razvoj Java aplikacija koje koriste relacione baze podataka. JDBC obzbeđuja programerima u Javi jedan uniformni interfejs za pristup i za rad sa relacionim bazama podataka. Upotrebom JDBC API , aplikacije pisane uJavi mogu da izvrše SQL iskaze, da dobijaju rezultate, da predstave podatak na način koji je ugodan korisniku i da vrate promenjene podatke u bazu podataka. JDBC API takođe omogućava da sa radi sa više izvora podataka u distribuiranim heterogenim okruženjima.

1. Šta predstavlja drajver baze podataka?

JDBC uslužni program (driver) služi kao interfejs koji olakšava komunikaciju između JDBC i neke posebne baze podataka, JDBCuslužni programi se razvijaju za specifične baze podataka, tj.DBMS, te ih najčešće nude prodavci tih sistema baza podataka. Na primer, vama je potreban MySQL JDBC uslužni program da bi pristupili MySQL bazi podataka, a Oracle JDBC uslužni programi da bi pristupili Oracle bazama podatak. Za Access baze podataka, upotrebite JDBC-ODBC mostni uslužni program koji je uključen u JDK.

1. Šta je RDBMS?

Ovo je relational database management system, tj sistem za upravljanje relacionim bazama podataka. Ovaj sistem omogućava kreiranje, održavanje I kontrolu baze podataka.

1. Šta čini relacioni model baze podataka?

Relacioni model podataka čine tri komponente: Struktura, Integritet (celovitost) I Jezik. Struktura definiše predstavljanje podataka. Integritet uvodi ograničenja podacima. Jezik obezbeđuje način pristupa i manipulacije sa podacima.

1. Šta je struktira baze podataka?

Struktura definiše predstavljanje podataka.

1. Šta je integritet baze podataka? Koja ograničenja integriteta postoje?

Svako ograničenje integriteta definiše uslov koje vrednosti u nekoj tabeli moraju da zadovolje.

Postoji tri tipa ograničenja: Ograničenja domena, Ograničenja primarnog ključa I Ograničenja sekundarnog ključa.

Ograničenja domena i primarnog ključa su intra-relaciona ograničenja jer se odnose samo na jednu relaciju (tabelu), Ograničenje sekundarnog ključa je inter-relaciono, jer se odnosi na više od jedne relacije (tabele).

Ograničenja domena definišu dozvoljene vrednosti nekog atributa. Mogu da se odnose korišćenje nekog od standardnih tipove podataka, a mogu da budu i uži, da ograniče opseg njihovih vrednosti.

Ključevi po kojima se mogu povezivati tabele mogu se podeliti i na sledeći način:

Superključ: atribut ili skup atributa koji na jedinstveni način identifikuje jednu realciju. Skup svih atributa u jednoj relaciji (slogu) je superključ.

Ključ-kandidat: podskup super ključa, koji može d aima više ključa-kandidata.

Primarni ključ je jedan od ključeva-kandidata, određen pod strane projektanta baze i često se koriste za identifikaciju slogova tabele

Sekundarni (strani) ključ ograničava veze između tabela (relacija).

Skup atributa FK je strani ključ (foreign key) u relaciji R koja povezuje relaciju T ako zadovoljava sledeća dva pravila:

* 1. Atributi FK imaju isti domen kao i primarni ključ u T.
  2. Vrednost nonull za FK u R mora da odgovara vrednosti primarnog ključa u T.

1. Šta je jezik baze podataka?

Jezik obezbeđuje način pristupa i manipulacije sa podacima. SQL je univerzalni jezik za pristupanje relacionim sistemima baza podataka. Krajnji korisnik ne mora da zna i koristi SQL, ako mu aplikacioni program to omogućava. Međutim, taj aplikacioni program mora da koristi SQL (u pozadini) da bi korisnik preko GUI, koristio podatke iz relacione baze podataka.

1. Koji tipovi SQL upita postoje (onsovne dve grupe)? U zavisnosti od tipa upita, koja metoda se poziva kada joj se kao String argument predaje SQL upit?

Prva grupa SQL upita je za kreiranje baze I njene definicije, a druga grupa upita je za manipulaciju podatacima u bazi.

1. Koje JDBC interfejse poznajete i šta je njihov zadatak?

Četiri ključna interfejsa neophodna za razvoj aplikace sa bazom podataka su: **Driver , Connection , Statement , i ResultSet** . Ovi interfejsi predstavljaju radni okvir za pristup generičkim SQL bazama podataka. JDBC API definiše ove interfejse a proizvodđači JDBC uslužnih programa (drajvera) obezbeđuju primenu specifikacija ovih interfejsa. Programeri upotrebljavaju ove interfejse.

JDBC aplikacija koristi odgovarajući JDBC uslužni progragram (drajver) upotrebom interfejsa Driver, povezuje se sa bazom podataka upotrebom interfejsa Connection, kreira SQL iskaz upotrebom interfejsa Statement, i obrađuje rezultate upotrebom interfejsa ResultSet/

1. Kada i kako se dobija ResultSet objekat? Kako se obavlja izvačenje rezutata iz ovog objekta?

**Statement statement = connection.createStatement();**

**ResultSet resultSet = statement.executeQuery(String query);**

Klasa ResultSet održava tabelu čiji se red čita. Početna pozicija reda je null. Možete koristiti metod next() da se izvršenje pomeri na sledeći red. Koriste se i različiti “geter” metodi za dobijanje vrednosti iz tekućeg reda.

1. Šta je PreparedStatement? Po čemu ovaj tip Statement objekta ima najčešću primenu u programima?

PreparedStatement proširuje interfejs Statement i omogućava kreiranje parametrizovanih SQL upita. On omogućava dinamičko kreiranje SQL upita i sprečava SQL injection napade.

1. Šta je CallableStatement i kako se koristi?

Interfejs CallableStatement je projektovan za izvršenje memorisanih SQL procedura (SQL-stored procedures).

**CallableStatement callableStatement = connection.prepareCall("{call sampleProcedure(?, ?, ?)}");**

Ovime pozivamo funkciju iz SQL baze pod nazivom sampreProcedure.

1. Šta su meta podaci baze podataka? Kako možemo da dobijemo ove podatke?

Metapodaci baze podataka, kao što su URL, ime korisnika (username), ime JDBC drajvera, dobijaju se upotrebom DatabaseMetaData interfejsa. Skup dobijenih meta podataka, kao što su broj kolona tabele, nazivi kolona, mogu se dobiti upotrebom ResultSetMetaData interfejsa.

DatabaseMetaData dbMetaData = connection.getMetaData();

dbMetaData.getURL();

dbMetaData.getUserName();

1. Šta je ORM?

ORM je tehnika programiranja koja vrši konverziju podataka između relacionih baza podataka i objektno-orijentisanih jezika, kao što su Java, C# i dr. ORM sistem ima sledeće prednosti u odnosu na JDBC:

* 1. Dozvoljava kodu aplikacije da pristupi objektima, a ne tabelama baze.
  2. Sakriva detalje SQL upita od OO logike.
  3. Koristi u pozadini JDBC.
  4. Ne zavisi od primenjenog sistema baza podataka (RDBMS)
  5. Entiteti se zasnivaju na poslovnim konceptima, a ne na strukturi baze podataka.
  6. Upravljanje transakcijama i automatska generacija ključa.
  7. Brz razvoj aplokacije.

Kada radimo sa objektno-orijentisanim sistemima, postoje neusaglašenost između objektnog modela i modela relacione baze podataka. RDBMS predstavljaju podatke u obliku tabele. Pri preuzimanju objekata iz baze i kod uskladištenja objekata u bazu, javljaju se sledeći problemi uvezani za: granularnost, nasleđivanje, identitet, asocijacije i navigaciju.

Object-Relational Mapping (ORM) je rešenje za navedene probleme neusklađenosti objektnog i relacionog modela.

1. Šta je Hibernate?

Hibernate je softverska komponenta koja se nalazi između Java objekata (u aplikaciji) i servera sa bazom podataka, i koja obavlja celokupan posao vezan za mapirane objekata u relacione tabele.

Hibeate mapira Java klase u tabele baze podataka i tipove podataka Jave u SQL tipove podataka i oslobađa programera od potrebe da realizuje 95% uobičajenih zadataka pri radu sa smeštanjem i korištenjem objekata u trajnoj memoriji (relacionoj bazi podataka).

Prednosti primene Hibernate OMR rešenja:

* 1. Hibernate se brine o mapiranju Java klasa u tabele baze podataka upotrebom XML fajlova bez potrebe da se napiše ni jedna linija programskog koda.
  2. Obezbeđuje jednostavne API za smeštaj i preuzimanje Java objekata iz baze podataka i ka bazi podataka.
  3. Ako dođe do promene u bazi (Database) ili u nekoj tabeli potrebo je samo da se promeni odgovarajuće svojstvo u XML fajlu.
  4. Nema potrebe da se radi sa SQL tipovim, te obezbeđuje rad samo sa Java objektima.
  5. Hibernate ne zahteva korištenje aplikacionog servera.
  6. Radi sa složenim asocijacijama objekata u bazi podataka.
  7. Obezbeđuje jedostavne upite podataka.

1. Kako se konfiguriše (podešava) Hibernate?

Objekat Configuration je prvi Hibernate objekat koji treba da kreirate u vašoj Hibernate aplikaciji i najčešće se kreira samo jedanput za vreme inicijalizacije aplikacije. Ovaj objekat predstavlja konfiguraciju ili fajl sa svojstvima koju zahteva Hibernate. Configuration objekat ima dve ključne komponente:

* 1. Database Connection: Ovu komponentu obezbeđuje jedan ili više konfiguracionih fajlova koji podržavaju Hibernate. Ovi fajlovi su hibernate.properties i hibernate.cfg.xml.
  2. Klasa Mapping Setup: Ova komponenta kreira vezu između Java klasa i tabele baze podataka.

1. Šta se postiže glavnim Hibernate konfiguracionim fajlom, a šta mapirajućim fajlovima za svaku entitetsku klasu?

Glavni konfigruacioni fajl kreira podešavanja za bazu, link do baze, jezik baze, korisničko ime I šifru. Mapirajući fajlovi određuju kako će se zasebne klase čuvati u bazi podataka.

1. Kako je moguće eliminisati XML mapirajuće fajlove primenom Java anotacija­?

Hibernate Annotations tj. korišćenje napomena, je moćan način obezbeđivanja meta podataka za mapiranje objekata u relacione tabele. Svi mеta podaci se stavljaju zajedno u POJO java fajl zajedno sa programskim kodom, te korisnik razume strukturu tabela i POJO simultano za vreme razvoja.

Hibernte Annotations je najnoviji način za definisanje mapiranja objekata u tabele i obrnuto, bez upotrebe XML fajla. Možete koristiti Hibernate Annotations kao dopunu ili kao zamenu XML mapiranja meta podataka. dobijate i veću fleksibilnost nego da ste koristili XML definisanje potrebnog mapiranja.

1. Šta je HQL?

Hibernate Query Language (HQL) je objektno-orijentisan jezik za rad sa bazom podataka u okviru Hibernate ORM. HQL radi sa trajnim objektima i njihovim svojstvima. HQL upiti se prevode u uobičajene SQL upite (od strane Hibernate-a) koji izvršavaju potrebne akcije na bazi podataka.

Vi možete i direktno da koristite SQL upita i u okviru Hibernate-a upotrebmo Native SQL, ali je prporučljivo da koristite HQL uvek kada možete, da bi izbegli probleme sa prenosivošću aplikacije (sa jedne na drugu bazu podataka), HQL takođe nudi i neke prednosti, kao što su strategije generisanja i keširanje