FREDRIK JONSSON SUT24

TEORIHANDBOKEN - DATABASER

Frågor:

1. **Olika typer av databaser**

    1. **Vad finns det för olika databastyper och vad är deras olika för- och nackdelar?**

    Man brukar dela upp det i 2 huvudtyper, relationsdatabaser och icke-relationella databaser ”NoSQL” som är ett samlingsnamn för dessa typer.

    Fördelar med relationsdatabaser är bra sökfunktioner med sql, ACID-kompatibelt, dataintegritet och strukturerat, några nackdelar är att det kan vara långsammare vid hantering av stor mängd ostrukturerad data.

    Fördelar med NoSQL är att det är flexibelt för att hantera olika datatyper, har en hög skalbarhet och prestanda för stora datamängder. Nackdelar är att det generellt har mindre stöd för komplexa frågor och transaktioner.

    2. **Hur ser datastrukturer ut i de typer av databaser du beskrev ovan?**

    Relationsdatabaser använder en tabellstruktur med rader och kolumner där det skapas relationer mellan dessa med hjälp utav Primary och foreign keys.

    NoSQL kan variera lite beroende på vilken typ det är, dokumentdatabaser använder JSON-liknande dokument. Nyckel-värde-databaser lagrar data som nyckel-värde-par.

    3. **Gör en lite mer fördjupad beskrivning av relationsdatabaser och SQL. Beskriv framför allt vad som gör det till just en *relations*databas, exempelvis primär och sekundär nyckel. Beskriv också vad SQL är för språk och hur det i grunden är uppbyggt.**

    En relationsdatabas är en databas där data är organiserad i tabeller och relationer mellan dessa tabeller är definierade. Detta görs genom att använda primärnycklar (Primary Key) som är unika identifierare för varje rad i en tabell och sekundärnycklar(Foreign Key) är referenser till primärnycklar i andra tabeller på så sätt så skapas relationer mellan tabellerna.

SQL står för Structured Query Language och är ett språk för att skapa, läsa, uppdatera och ta bort data ifrån relationsdatabaser. Det är uppbyggt utav olika satser som

DML (Data Manipulation Language): SELECT, INSERT, UPDATE, DELETE.

Där Select hämtar data, Insert lägger till data, Update uppdaterar data och delete tar bort data.

2. **Modellera databaser**

    1. **Vad är databasmodellering (ER och EER) och varför gör vi en sådan?**

        ER står för Entity-Relationship och används för att designa en databas, lite som ett UML diagram. En visuell representation av databasens struktur och relationer.

        EER står för Extended Entity-Relationship och är om ett utökat er diagram där man även skriver med vilka datatyper som används i de olika entiteterna.

        Det är bra att göra dessa modeller för att få en tydlig bild utav hur databasen är uppbyggd, och kan även ge en ökad förståelse mellan utvecklare och intressenter.

    2. **Vad är relationsalgebra? Ge något enkelt exempel på relationsalgebra och varför det används.**

        Ett formellt språk för att uttrycka databasfrågor med olika operationer.

        Exempel på relationsalgebraiska operationer är Select där den väljer de tupler(rader) ifrån ett speciellt predikat. Tillexempel om vi använder oss av det senaste projektet med skolan och vi vill sortera ut lärare ifrån en tabell med personal så kan det se ut såhär SELECT FROM Staff WHERE Role = ’Teacher’.

3. **Funktionalitet i databaser**

    1. **Vad är Views? Varför kan dessa vara bra att använda och finns det några nackdelar?**

        Views är virtuella tabeller som skapas från en eller flera tabeller, views är baserade på en SQL fråga. Fördelar med detta är att det kan förenkla komplexa frågor, ge en ökad säkerhet och logisk struktur. Man kan säga att man sparar komplexa frågor under ett nytt namn. Nackdelar med views är att det kan påverka prestandan om de är för komplexa.

Vad är Stored procedures? Varför kan dessa vara bra att använda och finns det några nackdelar?

Stored procedures är nästan som ett miniprogram som Reidar sa, en fördefinierad SQL sats som lagras i databasen, det ger även ökad säkerhet då man kan ge användaren tillgång till proceduren utan att de får direkt tillgång till tabellerna, enkelt att återandvända och det körs snabbare eftersom databasen kan optimera koden vid lagring. Nackdelar med stored procedures kan vara att det är svårare att underhålla och att felsöka de blir även låsta till en specifik databasplattform.

4. **Kommunicera med databaser**

    1. **Under kursen har vi kollat på två sätt att kommunicera med databaser från C#. Beskriv dessa två metoder lite kort.**

        <u>ADO.NET</u>

        ADO.NET är ett ramverk för att ansluta till och arbeta med databaser i C#. Det ger direkt kontroll över databasoperationer genom att använda klasser som SqlConnection och SqlCommand för att hantera anslutningar och SQL-kommandon. Med ADO.NET kan man skicka frågor i form utav querys, uppdatera data och hantera transaktioner direkt från C#-koden.

        <u>ENTITY FRAMEWORK</u>

        Är ett ORM-verktyg som står för (Object-Relational Mapping) som gör det möjligt att arbeta med databaser på ett mer objektorienterat sätt. Det gör om alla tabeller till klasser och rader till objekt som vi kan arbeta med i våran kod istället för direkta SQL frågor. Detta innebär att vi kan manipulera databasen med de olika klasserna och använda LINQ(Language Integrated Query).

    2. **Vad är några för- och nackdelar med dessa olika metoder?**

        <u>ADO.NET</u>

        Fördelar med ADO är att det arbetar snabbt och du har full kontroll över alla databasoperationer. Nackdelar är att det krävs mer kod och det är ökad risk för SQL injektionsattacker om frågeställningarna inte hanteras korrekt.

ENTITY FRAMEWORK

EF är enklare att hantera och är inte heller lika mycket kod som ADO, smidigt att jobba med när det är objektorienterat minskar risken för SQL fel. Negativt med EF kan vara att det arbetar långsammare än ADO.NET för vissa operationer, du har även lite mindre kontroll då mycket sköts ”bakom kulisserna”. Det kan även vara begränsad funktionalitet då EF inte alltid stöder alla funktioner är tillgängliga den underliggande databasen.