**1DV503 Databasteknik och modellering**

**Ebbe Karlstad**

Skolan för datavetenskap, fysik

och matematik, Linnéuniversitetet, Sverige

ek224ev@student.lnu.se

**Uppgift 1 SQL-frågor med MySQL Workbench DBMS**

|  |  |
| --- | --- |
| **Fråga 1** | **SQL:** SELECT e.fname, e.lname  -> FROM employee e  -> JOIN works\_on w ON e.ssn = w.essn  -> JOIN project p ON w.pno = p.pnumber  -> WHERE p.pname = 'Middleware'  -> OR p.pname = 'DatabaseSystems';  **Resultat:** |
| **Fråga 2** | **SQL:** select e.fname, e.lname  -> from employee e  -> join works\_on w ON e.ssn = w.essn  -> join project p on w.pno = p.pnumber  -> where p.pname = 'DatabaseSystems' and w.hours > 40;  **Resultat:** |

|  |  |
| --- | --- |
| **Fråga 3** | **SQL:** select p.pnumber, d.dnumber, p.plocation, e.lname, e.address, e.bdate  from project as p  join department as d on p.dnum = d.dnumber  join employee as e on d.mgrssn = e.ssn  where p.plocation = 'Houston';  **Resultat:** |
| **Fråga 4** | **SQL:**  select e.fname, e.lname, s.fname, s.lname  from employee as e  join employee as s on e.superssn = s.ssn;  **Resultat:** |
| **Fråga 5** | **SQL:**  select fname, lname, sex, address  from employee  where sex = 'F' and address like '%Houston%';  **Resultat:** |
| **Fråga 6** | **SQL:**  select fname, lname, bdate  from employee  where month(bdate) = 6;  **Resultat:** |
| **Fråga 7** | **SQL:**  select d.dname, avg(e.salary) as avg\_salary  from department as d  join employee as e on d.dnumber = e.dno  group by d.dname;  **Resultat:** |
| **Fråga 8** | **SQL:** select e.fname, e.lname, e.ssn, w.essn  from employee as e  left join works\_on as w on e.ssn = w.essn  where w.essn is null;  **Resultat:** |
| **Fråga 9** | **SQL:**  select e.fname, e.lname, d.dnumber, p.pname, e.salary  from employee as e  join department as d on e.dno = d.dnumber  join project as p on p.dnum = d.dnumber  where p.pname = 'LaserPrinters' and d.dnumber = '7' and e.salary > '50000';  **Resultat:** |
| **Fråga 10** | **SQL:**  select e.fname, e.lname, e.address, d.mgrssn  from employee as e  join department as d on e.dno = d.dnumber  where e.address like '%Houston%' and d.mgrssn = '333445555';  **Resultat:** |
| **Fråga 11** | **SQL:**  select e.fname, e.lname, e.salary, d.dname  from employee e  join department d ON e.dno = d.dnumber  where d.dnumber = (  select d.dnumber  from employee e  join department d ON e.dno = d.dnumber  group by d.dnumber  order by avg(e.salary) desc  limit 1  );  **Resultat:** |
| **Fråga 12** | **SQL:**  select d.dnumber, d.dname, avg(e.salary) as avg\_salary, count(e.ssn) as num\_e  from department as d  join employee as e on d.dnumber = e.dno  group by d.dname  having avg\_salary > '35000';  **Resultat:** |
| **Fråga 13** | **SQL:**  select e.fname, e.lname, e.address, d.mgrssn, dep.relationship  from employee as e  join department as d on e.dno = d.dnumber  left join dependent as dep on e.ssn = dep.essn  where d.mgrssn = '333445555'  order by e.fname asc;  **Resultat:** |
| **Fråga 14** | **SQL:**  select p.pname, sum(w.hours) as worked\_hours, count(e.ssn) as num\_employees  from project as p  join works\_on as w on p.pnumber = w.pno  join employee as e on w.essn = e.ssn  group by p.pname;  **Resultat:** |
| **Fråga 15** | **SQL:**  select d.dname, count(distinct p.pnumber) as num\_projects, count(distinct e.ssn) as num\_employees  from department as d  left join project as p on d.dnumber = p.dnum  left join employee as e on d.dnumber = e.dno  group by d.dname  **Resultat:** |

**Uppgift 2 Funktionella beroenden**

Uppgift 2.1 Lösning

Primärnyckel: Det finns inte en enda primärnyckel, då alla kolumner i tabellen har dubbletter. Dock kan kombinationen av EMPLOYEE\_ID och JOB\_ID användas som en primärnyckel.

|  |  |
| --- | --- |
| Funktionellt beroende | Förklaring/Exempel |
| {EMPLOYEE\_ID} 🡪 {NAME}  {JOB \_ID} 🡪 {JOB\_TITLE }  {POSTAL\_CODE } 🡪 {CITY } | Personalens ID kan fastställa namnet.  Jobbets ID kan fastställa titeln.  Postnumret bestämmer staden. |

**2.2 Lösning**

Primärnyckel: Igen finns det inte en enda primärnyckel, utan här kan en kombination av Product, Part och Type användas som en primärnyckel.

Funktionella beroenden:

* {Product, Part, Type} 🡪 {Material}
* {Product, Part, Type} 🡪 {Quantity}
* {Product, Part, Type} 🡪 {Product\_price}
* {Product, Part, Type} 🡪 {Part\_price}
* {Product, Part, Type} 🡪 {Part\_supplier}
* {Type} 🡪 {Material}

|  |  |
| --- | --- |
| *Anomali* | *Motivering/Förklaring* |
| Redundans | Product\_price repeteras flera gånger i tabellen, vilket inte är nödvändigt ifall priset på delen i fråga är samma, oberoende av vilken produkt den tillhör. |
| Uppdatering | Om Product\_price behöver uppdateras måste den göra det på flera ställen i tabellen, vilket kan leda till problem ifall den inte uppdaterats korrekt på alla ställen. |
| Radering | Ifall t.ex. alla delar av en produkt raderas från tabellen, tas också priset bort, eftersom priset delvis är beroende av delarna. |
| Införande | Införande av nya produkter är svårt då de måste innehålla alla delar (product, part, material, type, osv.) för att kunna finnas i tabellen. |

**Uppgift 3 Normalisering**

**3.1 Lösning**

Finaltabeller/relationer alla i 3NF-formen:

Här har vi i stället för att ha en tabell gjort tre tabeller. Product Table, Parts Table, och Product\_Part Table (som används för att länka produkter med deras delar).

----------------------------------------------------------

A screenshot of a phone

Description automatically generated

------------------------------------------------------------------------------------------------------

A white background with black text

Description automatically generated

-----------------------------------------------------------

A screenshot of a table

Description automatically generated

**3.2 Lösning**

|  |  |
| --- | --- |
| **Förklaring** | **Resultat** |
| Baserat på den givna primärnyckeln, finns denna relation i 1NF, 2ND eller 3NF?  *Ditt svar här:* Relationen finns i 1NF och inte i de andra då discount och commission endast är beroende av delar av primärnyckeln, inte hela. För att konvertera denna till 3NF eliminerar vi dessa partiella beroenden genom att skapa flera tabeller. | Diagram före normalisering: |
|  |  |
| *Sista Resultat*  Jag delade först upp tabellen för att se till att all information (som provision och rabatt) beror på en hel unik identifierare, och inte bara delar av den. Därefter såg jag till att saker som förändras, som t.ex. rabatter på olika dagar har sina egna rader. Nu när tabellerna är i 3NF kommer inte ändring av ett attribut förstöra för andra. | Det sista diagrammet visar alla tabeller i 3NF:  CAR\_SALE Table R1    **---------------------------------**  DISCOUNT\_TABLE R2    **----------------------------------**  COMMMISION\_TABLE R3 |