Lösningsförslag - Frågespråk
Databassystem

Relationsalgebra:

1. $\pi_{namn}\ \sigma_{pnr = '010101\text{-}0101'}(Person)$

   Projektionssymbolen π används för att hämta ut de attribut som är av intresse, i detta fall personens namn. Selktionssymbolen σ används för att lista villkoret som måste vara uppfyllt för att kunna få ut ett svar. I detta fall listas personnumret som gäller som villkor. Inom parentes listas de tabeller som användsför att hämta ut svaret. I detta fall används endast persontabellen.

2. $\pi_{pnr}\ \sigma_{namn = 'kalle' \text{ and } ålder = 18}(Person)$

   Projektion listar personnumret. Selektionsdelen innehåller de villkor som måste vara uppfyllt. I detta fall 2st, namn och ålder.

3. $\pi_{bil.toppfart}\ \sigma_{äger.regnr = bil.regnr}(Äger \times Bil)$

   I denna övning finns inte all data i en och samma tabell, vilket innebär att vi måste identifiera vilka tabeller som måste användas för att hitta det sökta svaret. Eftersom bilens toppfart är en del av svaret vet vi att biltabellen måste användas. För att säkerställa att någon äger bilen måste bilen finnas kopplad till ägertabellen. Om bilen finns med i ägertabellen finns det någon person som äger bilen. Tabellerna som måste användas är bil och äger. Eftersom det är ointressant att veta vem som äger bilen behövs inte persontabellen.

   Det som alltid måste göras när men än en tabell används är att joina dem genom att länka ihop biltabellen med ägertabellen med hjälp av deras gemensamma nyckel. Ovan syns att joinen/länkningen under selektionsdelen: äger.regnr = bil.regnr. Här syns att den gemensamma nyckeln som finns i båda tabellerna är regnr. Eftersom vi använder mer än en tabell anger alltid tabellnamnet först följt av en punkt och dess nyckel (eller attribut om vi inte joinar). Detta innebär att datan kommer från en unik tabell. Om tabellnamnet inte anges kan det bli problematiskt

då det kan finnas attribut med samma namn och som förekommer i flera andra tabeller, exempelvis finns namn i person och ägertabellerna mm. Glöm inte att länkning måste ske mellan samtliga tabeller som används. Observera att länkningen sker med hjälp av de gemensamma nycklarna i varje tabell. Exempelvid är nyckeln regnr gemensam för bil och ägertabellen. Inom parentes listas Äger x bil, vilket i praktiken innebär att en kartesisk produkt skapas mellan dessa tabeller. Se gärna kurslitteratur för att förstå vad en kartesisk produkt innebär.

4.

$$\pi_{\text{person.namn}}\ \sigma_{\text{person.pnr} = \text{äger.pnr and äger.regnr} = \text{bil.regnr and bil.färg} = \text{'blå'}}\ (\text{Person x Äger x Bil})$$

I denna övning måste data användas från tabellerna person, äger och bil. Persontabellen måste användas eftersom personens namn är datan som svaret ska innehålla. Personen måste äga en bil, därför måste personen finnas med i ägertabellen. Bilen måste vara blå samt att bilen måste ägas av någon. Tabellerna person, äger och bil innehåller all data som måste användas för att hitta det sökta svaret.

Länkningen måste ske mellan samtliga tabeller, vilket syns ovan när pnr joinar mellan person och äger, regnr mellan äger och bil. Glöm inte att lista den kartesiska produkt mellan samtliga tabeller inom parentes.

5.

$$\pi_{\text{bil.färg}}\ \sigma_{\text{äger.regnr} = \text{bil.regnr and äger.pnr} = \text{'010101-0101'}}\ (\text{Äger x Bil})$$

Bilens färg listas under projektion eftersom svaret kräver detta. Selektionens personnummer finns tillgängligt i ägertabellen, vilket innebär att tabellerna äger och bil måste användas. En vanlig lösning är att man använder sig av persontabellen också eftersom frågan inkluderar personnummer som är information om personen. Detta är inte nödvändigt eftersom personnumret redan finns med i ägertabellen. Självklart är det inte fel att ta med persontabellen också, men det ses som onödigt, se nedan:

$$\pi_{\text{bil.färg}}\ \sigma_{\text{äger.regnr} = \text{bil.regnr and äger.pnr} = \text{person.pnr and person.pnr} = \text{'010101-0101'}}\ (\text{Person x Äger x Bil})$$

En tumregel som är värd att följa är att alltid ställa sig frågan vilka data som måste användas för att besvara frågan. Nästa steg är att identifiera vart datan finns i tabellerna. Försök att undvika att använda onödiga tabeller då det kan innebära ytterligare en källa till fel, exempelvis om man glömmer att länka till tabellen mm. Huvudsaken är dock att du är bekväm med frågeställningen.

6.

$$\pi_{\text{parkeringsplats.namn}}\, \sigma_{\text{parkeringsplats.nr} = 1}\ (\text{Parkeringsplats})$$

Här tränas du i att identifiera vart data finns. Eftersom parkeringsplats är en svag entitet innehåller den både parkeringsplatsnummer och parkeringshusnamn. De behövs alltså inte använda både tabellerna parkeringsplats och parkeringshus, även om det skulle vara tillåtet. I denna övning används också punktoperatorn trots att endast en tabell används. En tumregel är att alltid lista tabellens namn följt av de attribut/nycklar som används för att besvara frågan, oavsett om en eller många tabeller används.

7.

$$\text{Alla\_personer} \leftarrow \pi_{\text{person.pnr}}\ (\text{Person})$$

$$\text{Alla\_ägare} \leftarrow \pi_{\text{äger.pnr}}\ (\text{Äger})$$

$$\text{Ej\_ägare} \leftarrow \text{Alla\_personer} - \text{Alla\_ägare}$$

$$\text{Resultat} \leftarrow \pi_{\text{person.namn}}\, \sigma_{\text{person.pnr} = \text{Ej\_ägare.pnr}}\ (\text{Person} \times \text{Ej\_ägare})$$

Denna övning går inte att lösa med ett uttryck. Istället måste temporära tabeller användas för att successivt komma fram till ett resultat. Första steget är att hämta ut alla personer som existerar. Detta sker genom att hämta personnummer från tabellen person och lägga in resultatet i den temporära tabellen Alla_personer. Nästa steg är att hämta alla personer som äger någon bil och lägga i den temporära tabellen Alla_ägare. Detta sker genom att hämta personnummer från ägertabellen. Namngivningen av de temporära tabellerna väljer du själv.

När vi känner till alla personer och alla som äger kan vi utföra en differens för att ta reda på vilka personer som inte äger en bil. Eftersom tabellerna Alla_personer och Alla_ägare är unionkompatibla (båda tabellerna innehåller personnummer) går det utföra en differens. Observera att tabellerna måste alltid vara unionkompatibla för att kunna utföra någon sorts beräkning. Det går exempelvis inte utföra en differens mellan personnummer och namn. Det är ungefär som att jämföra äpplen och päron, vilket inte fungerar inom databassystem. Resultatet av differensen, dvs. de personer som inte äger någon bil läggs i en temporär tabell Ej_ägare. I denna tabell lagras alla personnummer för dem som inte äger en bil. Eftersom svaret för frågan efterfrågar namnet för de personer som inte äger en bil så måste vi ta reda på vad alla heter som finns lagrade i tabellen Ej_ägare. Detta måste ske genom att joina med persontabellen. När detta är utfört har vi nått vårt resultat.

En bra tumregel att lära sig är när en fråga efterfrågar något som ”inte” existerar brukar en differens fungera väl.

8.

$$\text{Alla\_bilar} \leftarrow \pi_{\text{bil.regnr}} (\text{Bil})$$

$$\text{Alla\_ägda\_bilar} \leftarrow \pi_{\text{äger.regnr}} (\text{Äger})$$

$$\text{Ej\_ägda\_bilar} \leftarrow \text{Alla\_bilar} - \text{Alla\_ägda\_bilar}$$

$$\text{Resultat} \leftarrow \pi_{\text{bil.färg}} \; \sigma_{\text{bil.regnr} = \text{Ej\_ägda\_bilar.regnr}} (\text{Bilx Ej\_ägda\_bilar})$$

Samma tänk som föregående uppgift. Observera att tabellerna är unionkompatibla. Först hämtas alla bilar och alla bilar som finns med i tabellen äger. Differensen tar fram alla bilar som inte har en ägare. Sist sker en join för att ta reda på färgen för de bilar som finns i den temporära tabellen Ej_ägda_bilar.

9.

$$\text{Alla\_bilar} \leftarrow \pi_{\text{bil.regnr}} (\text{Bil})$$

$$\text{Alla\_ägda\_bilar} \leftarrow \pi_{\text{äger.regnr, äger.pnr}} (\text{Äger})$$

$$\text{Äger\_alla\_bilar} \leftarrow \text{Alla\_ägda\_bilar} / \text{Alla\_bilar}$$

$$\text{Resultat} \leftarrow \pi_{\text{person.namn}} \; \sigma_{\text{Äger\_alla\_bilar.pnr} = \text{person.pnr}} (\text{Person x Äger\_alla\_bilar})$$

Division är operatorn som måste användas för att singla ut vem som äger alla bilar. Vänstersidan av divisionen innehåller resultatsattributet och kopplingsattributet (till nämnaren). Högra sidan innehåller attributet som ska jämföras med. I detta fall alla bilars registreringsnummer. Kopplingsattributet på vänster sida måste vara uninonskompatibelt med nämnare, vilket syns då äger.regnr är unionkompatibelt med bil.regnr (samma nyckel). Divisionen singlar alltså ut de resultatsattribut som är gemensam för alla kopplingsattribut. Dvs. för alla existerande bilar måste det finnas ett gemensamt resultat (äger.pnr), annars finns det inte någon person som äger alla bilar. Se gärna kursmaterial för hur division fungerar mer detaljerat.

Eftersom resultatet av divisionen listar personnummer måste tabellen joinas med person för att ta reda på personen/personernas namn.

10.

$$\text{Alla\_personer} \leftarrow \pi_{\text{person.pnr}}(\text{Person})$$

$$\text{Alla\_personer\_som\_äger} \leftarrow \pi_{\text{äger.regnr, äger.pnr}}(\text{Äger})$$

$$\text{Ägs\_av\_alla} \leftarrow \text{Alla\_personer\_som\_äger} / \text{Alla\_personer}$$

$$\text{Resultat} \leftarrow \pi_{\text{bil.toppfart}}\, \sigma_{\text{Ägs\_av\_alla.regnr} = \text{bil.regnr}}(\text{Bil} \times \text{Ägs\_av\_alla})$$

Samma tänk på ovan. Fokus är att hitta en täljare som innehåller det resultat som är intressant, dvs. bilens registreringsnummer. Täljaren måste dessutom innehålla det värde som ska jämföras mot, vilket måste unionkompatibel med nämnaren. I detta fall personnumret för de som äger en bil. Nämnaren består av alla personer. Divisionen listar de registreringsnummer (om det finns något) som överensstämmer med alla personer som äger bilen. Resultatet av divisionen läggs i den temporära tabellen Ägs_av_alla. Eftersom frågan kräver att toppfarten listas måste en join utföras med biltabellen. För de registreringsnummer som överensstämmer med biltabellens registreringsnummer listas dess toppfart.

SQL:

1.

```
SELECT namn
FROM person
WHERE pnr='010101-0101';
```

Select hämtar det resultat som är intressant, vilket är personens namn. Where-klausulen listar det villkor som måste vara sant, vilket är personens personnummer som måste vara 010101-0101. From-klausulen listar den tabell som används för att lösa frågan, vilket är persontabellen.

2.

```
SELECT person.pnr
FROM person
WHERE person.namn='Kalle' and person.ålder=18;
```

Här gäller det att få med hela villkoret som är både personens namn och ålder. I denna övning listas också tabellens namn följt av punkt och attribut/primärnyckel. En tumregel är att alltid skriva ut tabellens namn oavsett om en eller flera tabeller används. Då undviker vi problem när attribut från olika tabeller har samma namn. Under villkorsdelen (where) kombineras de båda villkoren med ”and” som menar att båda villkoren måste tillsammans utvärderas till sant för att hela villkoret ska bli sant.

3.

```
SELECT bil.toppfart
FROM bil, äger
WHERE äger.regnr=bil.regnr;
```

Om en bil ska ägas av någon måste den finnas med i ägertabellen. Eftersom datan kommer från två olika tabeller måste dem joinas/länkas, vilket alltid sker mha. den gemensamma primärnyckeln. I detta fall är ”regnr” gemensam för båda tabellerna. Om bilens regnr finns med i ägertabellen så har bilen en ägare. Det är ointressant vem som äger den.

4.

```
SELECT person.namn
FROM person, äger, bil
WHERE person.pnr=äger.pnr and äger.regnr=bil.regnr
and bil.färg='blå';
```

Här måste data användas från tabellerna person, äger och bil. Svårigheten i övningen är att förstå hur länkningen/join sker med samtliga tabeller. Det är inte tillåtet att glömma att joina någon tabell som används. En tumregel för att undvika fel är att alltid börja med att länka alla tabeller följt av det attribut som ska utvärderas, i detta fall om bilens färg är blå.

5.

```
SELECT bil.färg
FROM äger, bil
WHERE äger.regnr=bil.regnr and äger.pnr='010101-
0101';
```

Här gäller det att förstå vilka tabeller som måste användas för att besvara frågan. Biltabellen är relevant eftersom frågan kräver att bilens färg listas, som enbart finns tillhanda i biltabellen. Men då bilen också ska ägas av en viss person ser vi att ägertabellen innehåller personnummer till de som äger bilar. I detta fall måste även ägertabellen användas. När mer än två tabeller används måste de joinas enligt ovan följt av de villkor som ska vara uppfyllt, i detta fall det listade personnumret.

Ibland är det vanligt att man vill använda mer tabeller än vad som minst krävs, vilket är helt okej så länge tabellerna länkas/joinas. Eftersom uppgiften listar ett personnummer kan det kännas sunt (mer naturligt) att använda persontabellen (eftersom pnr finns lagrat där också). Detta är alltså helt okej, men anses lite onödigt då ägertabellen innehåller det personnummer som är intressant för oss. Nedan ges ett exempel där persontabellen är inkluderad. Även om det ofta är önskvärt att använda så få tabeller som möjligt så rekommenderas att du hittar det som är bekvämt för dig. Om det är lättare att ta reda på ett svar mha. fler tabeller än nödvändigt är helt okej. Huvudsaken är att frågan besvaras korrekt.

```
SELECT bil.färg
FROM person, äger, bil
WHERE äger.regnr=bil.regnr and äger.pnr=person.pnr
and person.pnr='010101-0101';
```

6.

```
SELECT parkeringsplats.namn
FROM parkeringsplats
WHERE parkeringsplats=1;
```

Enligt relationsdatamodellen innehåller den svaga entiteten parkeringsplats parkeringshusets namn. Detta innebär att parkeringsplatstabellen är den enda tabell som behövs för att besvara frågan. Det är alltså inte något ”extra” som krävs bara för att vi använder en svag entitet. En rekommendation är att alltid titta i relationsdatamodellen för att se vilken data som finns i respektive tabell. Då är det enklare att ta reda på vilka tabeller som måste användas för att besvara frågan.

7.

```
SELECT person.namn
FROM person
WHERE NOT EXISTS (SELECT *
                  FROM äger
                  WHERE person.pnr=äger.pnr);
```

När något inte ska finnas med i en tabell finns en bra verktyg i SQL: not exists. Om personen som efterfrågas inte finns med i ägertabellen vet vi att personen inte äger någon bil. Detta syns ovan då ”not exists” står listat under villkorsdelen. Syntaxen för ett ”not exists” är att den alltid består av en nästlad fråga, vilket syns nedan: `NOT EXISTS (SELECT * FROM x WHERE y)`. Observera hur tabellerna under ”from” står listade. Först listas person eftersom personens namn tillhör frågans svar. I villkorsdelen listas äger samt att länkningen/join sker mellan ägertabellen och persontabellen. Detta innebär att det inte finns någon person som finns med i ägertabellen så ska denne tillhöra frågans svar.

Ett vanligt misstag är att samma tabell listas flera gånger, vilket ger ett inkorrekt svar. Se gärna föreläsningsmaterial för en mer djupgående förståelse angående detta. Nedan visas ett felaktigt exempel som ska undvikas.

```
SELECT person.namn
FROM person
WHERE NOT EXISTS (SELECT *
                  FROM person, äger
                  WHERE person.pnr=äger.pnr);
```

Det som sker i praktiken när samma tabell listas fler gången är att en jämförelse kommer ske inom en och samma tabell, vilket vi inte vill göra. Fokus är att ta reda på data från en tabell finns med i en helt annan tabell.

8.

```
SELECT bil.färg
FROM bil
WHERE NOT EXISTS (SELECT *
                  FROM äger
                  WHERE äger.regnr=bil.regnr);
```

Samma tänk här som tidigare uppgift. ”Not exists” returnerar ett svar då villkoret är sant, alltså om en bil inte finns med i ägertabellen. Om detta är sant vet vi att bilen inte har någon ägare. Om det returneras ett svar listas bilens färg som svar.

9.

```
SELECT person.namn
FROM person
WHERE NOT EXISTS (SELECT * FROM bil
WHERE NOT EXISTS (SELECT * FROM äger
WHERE äger.regnr=bil.regnr and
äger.pnr=person.pnr));
```

När alla av något ska hämtas blir SQL-frågan genast svår. Tyvärr finns det inte ett liknande verktyg som ”not exists” som används för att lista något som inte finns. Det enda tankesätt som fungerar är att använda dubbel-negation. Här kommer två påstående som är båda lika logiskt korrekta:

1) **Alla** tomtar har skägg.
2) Det finns **inte** någon tomte som **inte** har skägg.

Eftersom båda påståendena är lika korrekta används dubbel-negation (dvs. inte finns med 2 ggr) när vi försöker lösa en SQL-fråga som vill hämta allt av något.

Svaret för denna övning visar att ”not exists” används två gånger där man kan utläsa att det ska inte finnas någon bil som inte finns med ägertabellen. Dvs. det finns inte någon bil som inte ägs av den sökta personen. Om detta villkor är sant vet vi att svaret består av den person eller personer som äger alla bilar.

Observera hur länkningen/join sker mellan tabellerna som används. Vid dubbel-negation listas länkning/join i den sista ”not existen”.

10.

```
SELECT bil.toppfart
FROM bil
WHERE NOT EXISTS (SELECT * FROM person
WHERE NOT EXISTS (SELECT * FROM äger
WHERE äger.pnr=person.pnr and
äger.regnr=bil.regnr));
```

Samma tänk som i den tidigare uppgiften. Det spelar egentligen inte någon roll för vilken tabell man börjar med i den dubbla negationen. Huvudsaken är att länkningen/join sker sist.