WESTERDALS Oslo ACT
EKSAMEN
DB1100 DATABASER

Tillatte hjelpemidler: Ingen
Varighet: 180 minutter
Dato: 6. desember 2017

1 vedlegg: MySQL, utvalgte datatyper/syntaks

Denne eksamen utgjør 100% av karakteren i DB1100.

Der databasetype er relevant for spørsmålet tar oppgavene utgangspunkt i en MySQL database. Du kan anta at verktøyet MySQL Workbench benyttes der verktøy er relevant.

Pass på at du disponerer tiden riktig i forhold til vektingen av de ulike oppgavene.

Oppgi eventuelle forutsetninger du tar.

Oppgave 1: GDPR – 10%

Gjør rede for General Data Protection Regulation (GDPR). Hva er det? Hvem blir påvirket? Når trer GDPR i kraft?

Oppgave 2: Modellering – 20%

En bedrift har ansvar for å fylle på varer i automater som inneholder mat og drikke. Du skal lage en database som inneholder relevante data for denne virksomheten. Virksomheten beskrives slik:

Automatene våre har en spesifikk plassering (lokasjon). En lokasjon har en X- og Y-koordinat. En lokasjon har også et navn (eks: Westerdals). Det kan være flere automater på samme lokasjon. For en automat må det også lagres hvor mange varer automaten har plass til.

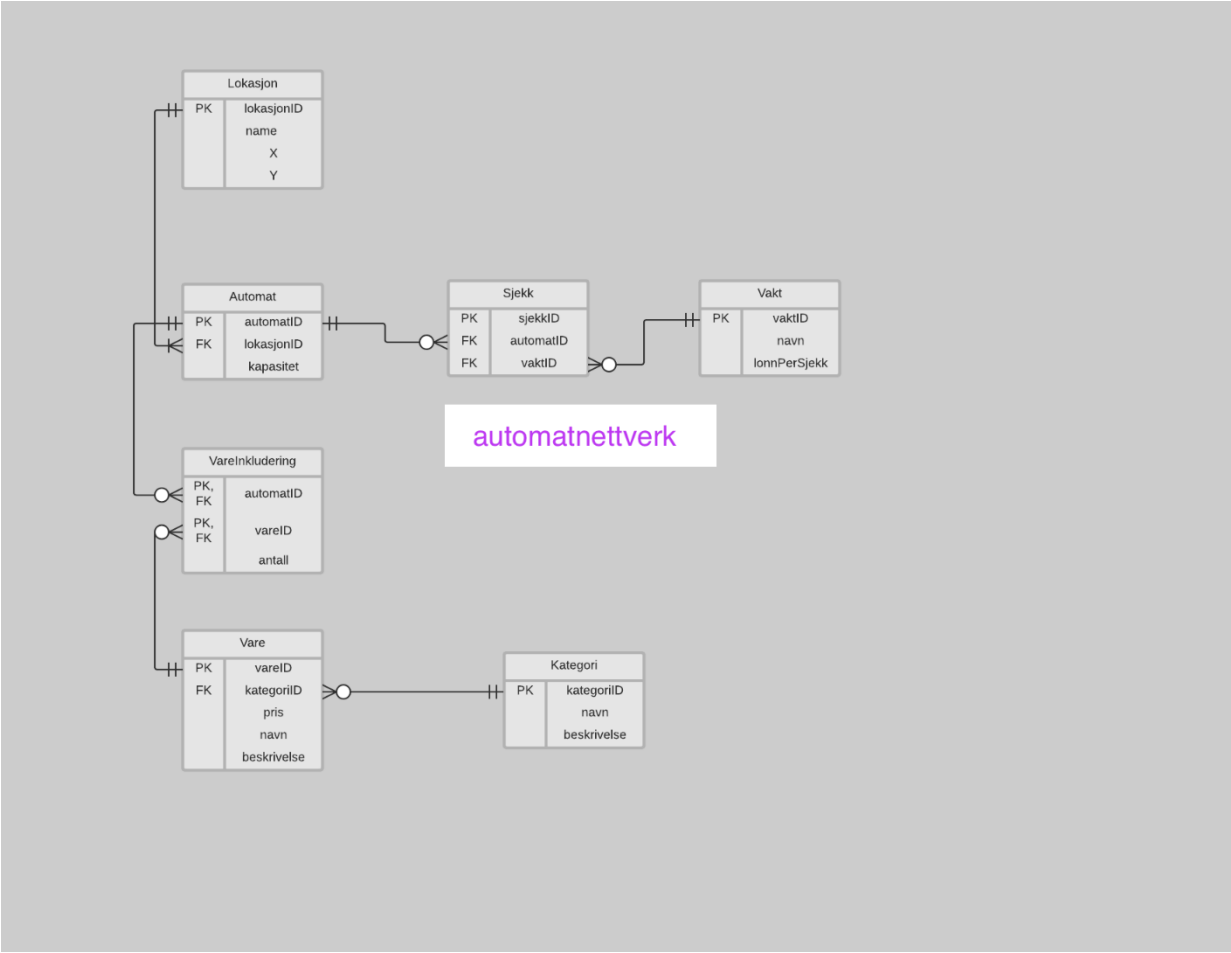
Vi fyller jo på varer på disse automatene. En vare har et navn, en pris og en beskrivelse. Eks: «Stratos» som koster 20 kr og kan beskrives på denne måten: «Luftig og porøs sjokolade med en herlig smak!». I tillegg tilhører alle varer én kategori. Flere forskjellige varer kan tilhøre samme kategori. Eksempler på kategorier kan være «Brus», «Sjokolade» etc.. I tillegg til å lagre navn på kategoriene, så vil vi lagre en beskrivelse for hver kategori også.

En automat har gjerne mange varer. Vi må vite hvilke varer som befinner seg i hvilke automater, og hvor mange eksemplarer av varen det er snakk om.

P.S. Takk til en av årets mappe-studenter som modellerte denne databasen i år. Løsningsforslaget som blir lagt ut etter eksamen viser studentens løsning.

Oppgave: Tegn en modell (kråkefot- eller UML-notasjon) for din foreslåtte løsning. Modellen din skal inneholde:

- Entitetene og deres attributter.
- Primærnøkler og fremmednøkler.
- Relasjonene mellom entitetene.
- Multiplisiteten (deltagelse og kardinalitet) for relasjonene.
- Koblingsentiteter (hvis nødvendig)



OBS! «Sjekk» og «Vakt» var ikke en del av oppgaven.

Oppgave 3: SQL – 50%

Følgende tabeller er basert på en database laget av en av dine medstudenter (som valgte mappe som vurderingsform). De tre tabellene omhandler filmer, skuespillere og hvilke filmer skuespillere har spilt i.

`actor`

- `actor_id {PK}` INT NOT NULL, AUTO_INCREMENT
- `name` VARCHAR(40) DEFAULT 'Ukjent'
- `age` INT NOT NULL
- `date_of_birth` DATE NOT NULL
- `gender` ENUM('Male', 'Female')

`film`

- `film_id {PK}` INT NOT NULL, AUTO_INCREMENT
- `title` VARCHAR(40) NOT NULL
- `release_year` INT NOT NULL
- `age_limit` INT NOT NULL
- `length_minutes` INT NOT NULL

`film_actor`

- `actor_id{PK}{FK}` INT NOT NULL
- `film_id {PK}{FK}` INT NOT NULL

Videre informasjon

- `actor_id` i `film_actor` er fremmednøkkel til `actor_id` i `actor`.
- `film_id` i `film_actor` er fremmednøkkel til `film_id` i `film`.

Se gjerne vedlegget for tips til SQL syntaks.

a) Skriv en spørring som henter ut navn og alder på alle skuespillere. Sorter resultatet på alder (den eldste skuespilleren skal komme først).

```
SELECT name, age FROM actor
ORDER BY age DESC;
```

b) Skriv en spørring som henter ut tittel på alle filmer som har en aldersgrense som er mindre eller lik 12 år.

```
SELECT title FROM film
WHERE age_limit<=12;
```

c) Lag et view som viser tittel, utgivelsesår og lengde (i minutter) for alle filmer som har 18-års aldersgrense. Viewet skal hete «Voksenfilmer».

```
CREATE OR REPLACE VIEW voksenfilmer AS
SELECT title, release_year, length_minutes
FROM film
WHERE age_limit = 18;
```

d) Skriv en spørring som henter ut navn og alder på de tre yngste skuespillerne i databasen.

```
SELECT name, age
FROM actor
ORDER BY age ASC
LIMIT 3;
```

e) Tabellen film_actor har to fremmednøkler som begge har egenskapen «ON DELETE CASCADE». Hva innebærer det?

Pensumbok s. 76.

f) Skriv en SQL som legger inn en ny skuespiller i databasen. Du velger selv hvilken informasjon som skal legges inn for skuespilleren.

```
INSERT INTO actor (name, age, date_of_birth, gender)
VALUES ('Bryan Cranston', 61, '1956-03-07', 'Male');
```

g) Skriv en spørring som oppdaterer innhold i tabellen `actor`. Skuespilleren med actor_id = 1 har endret navn. Nå skal han ha navn: «Aaron Paul».

```
UPDATE actor
SET name = 'Aaron Paul'
WHERE actor_id = 1;
```

h) Skriv en spørring som henter ut informasjon om hvor mange filmer som er registret innenfor hver aldersgrense. Du skal hente ut aldersgrensen og antall filmer for hver aldersgrense.

```
SELECT age_limit, Count(*)
FROM film
GROUP BY age_limit;
```

i) Skriv en spørring, ved bruk av subquery, som henter ut navn og kjønn (gender) på alle skuespillere som har spilt i minst én film med aldersgrense 18 år.

```
SELECT name, gender
FROM actor WHERE actor_id IN(
SELECT actor_id FROM film_actor WHERE film_id IN(
SELECT film_id FROM film WHERE age_limit=18));
```

j) Skriv en spørring, ved hjelp av JOIN, som henter ut tittel og utgivelsesår på alle filmer som skuespilleren med navn «Jennifer Lawrence» har spilt i.

```
SELECT title, release_year
FROM actor JOIN film_actor
ON actor.actor_id = film_actor.actor_id
JOIN film ON film_actor.film_id = film.film_id
WHERE actor.name = ' Jennifer Lawrence ';
```

Oppgave 4: Normalisering – 20%

Følgende tabell inneholder resultater fra verdenscupen i skihopp:

- Hoppresultater (Nr, Navn, Nasjon, Dato, Bakke, Omgang, Lengde, Stilkarakterer)

Nr identifiserer en skihopper. *Navn* er navnet til skihopperen, *Nasjon* er navnet på landet hopperen representerer, *Dato* er dato for skirennet, *Bakke* er navnet på skibakken, *Omgang* sier om det er 1. eller 2. omgang, *Lengde* er skihoppets lengde i meter og *Stilkarakterer* inneholder 5 karakterer, en fra hver av de 5 dommerne som bedømmer stilen i hvert hopprenn. Eksempel på en rad i tabellen:

(34, 'Robert Johansson', 'Norway', '18.03.2017', 'Viksersund', 1, 252, '19 18.5 19.5 19 18')

Du kan forutsette at ikke arrangeres mer enn ett hopprenn i verdenscupen per dag. Du kan velge å splitte opp kolonner i flere kolonner, hvis du mener det er hensiktsmessig.

Skriv opp funksjonelle avhengigheter, og utfør normalisering til Boyce-Codd normalform (BCNF). Merk primærnøkler og fremmednøkler i sluttresultatet.

Funksjonelle avhengigheter:

Nr->Fornavn, Etternavn, Nasjon

Dato->Bakke

Dato + Nr + Omgang ->alle andre attributter

Splitter navn i Fornavn og Etternavn, og Karakterer i 5 kolonner først.

Tabellstruktur etter normalisering:

Hopper (Nr, Fornavn, Etternavn, Nasjon)

Hopprenn (Dato, Bakke)

Skihopp (Dato*, Nr*, Omgang, Lengdepoeng, Kar1, Kar2, Kar3, Kar4, Kar5)

- Slutt på oppgavesettet –

Vedlegg: MySQL, utvalgte datatyper/syntaks

[] angir valgfrie elementer og | angir alternativer.
Dette er en noe forenklet syntaks-oversikt som forhåpentligvis skal hjelpe dere i oppgavene på denne eksamen.

DATATYPER

`CHAR[(lengde)]`	Tekst med fast lengde, fra 0 til 255 tegn
`VARCHAR(lengde)`	Tekst med variabel lengde, fra 0 til 65 535 tegn
`INT[(lengde)]`	Heltall i området -2 147 483 648 til 2 147 483 647
`ENUM(v1, v2...)`	Definert verdisett
`DATETIME`	Tidsinformasjon som rommer både dato og klokkeslett.
`DATE`	Dato. Eks: '2016-12-06'
`DECIMAL(A,B)`	Desimaltall med A siffer og B desimaler . Eks: `DECIMAL(5,2)` kan romme tall i området -999.99 - 999.99

FUNKSJONER

`COUNT(*)`	Antall
`AVG(kolonne_navn)`	Gjennomsnitt
`SUM(kolonne_navn)`	Sum
`MIN(kolonne_navn)`	Minimum
`MAX(kolonne_navn)`	Maksimum

OPERATORER

`=`	Lik (ikke m/wildcards!)
`<>` eller `!=`	Forskjellig fra
`<`	Mindre enn
`>`	Større enn
`<=`	Mindre eller lik
`>=`	Større eller lik
`LIKE`	Lik, godtar wildcards
`IN`	I gitt utvalg
`BETWEEN`	Utvalg, følges av `AND`
`_`	Wildcard, enkelt tegn
`%`	Wildcard, flere tegn
`IS [NOT] NULL`	Null / ikke null

LOGISKE OPERATORER

`AND`	Og
`OR`	Eller
`NOT`	Ikke

SYNTAKS

SELECT

```
SELECT      kolonnenavn [[AS] navn]
FROM        tabellnavn [[AS] alias]
[JOIN       tabellnavn [[AS] alias] ON felles nøkkel]
[WHERE      betingelse]
[GROUP BY   grupperingsuttrykk [HAVING betingelse]]
[ORDER BY   kolonnenavn]
```

VIEW

```
CREATE [OR REPLACE] VIEW viewnavn [(alias)]
AS kriterier
```

CREATE TABLE

```
CREATE TABLE tabellnavn
(
kolonnenavn datatype [UNIQUE|NOT NULL|DEFAULT|AUTO_INCREMENT|...],
...,
[[CONSTRAINT navn] PRIMARY KEY (kolonnenavn) ],
[[CONSTRAINT navn] FOREIGN KEY (kolonnenavn) REFERENCES tabellnavn (kolonnenavn) ]
)
```

ALTER TABLE

```
ALTER TABLE tabellnavn
[ADD kolonnenavn datatype],
[CHANGE kolonnenavn_nå kolonnenavn_ny datatype_ny],
[DROP COLUMN kolonnenavn]
```

DROP TABLE

```
DROP TABLE tabellnavn
```

INSERT INTO

```
INSERT INTO tabellnavn
VALUES (verdi1, verdi2, verdi3, ...)
eller
INSERT INTO tabellnavn (kolonnenavn1, kolonnenavn2, ...)
VALUES (verdi1, verdi2, ...)
```

UPDATE

```
UPDATE tabellnavn
SET kolonnenavn1 = verdi1, kolonnenavn2 = verdi2, ...
[WHERE betingelse]
```

DELETE FROM

```
DELETE FROM tabellnavn WHERE betingelse
```