NTNU
Kunnskap for en bedre verden

Introduksjon til databaser

Tore Mallaug, NTNU

1. Relasjonsmodellen

Resymé: Denne leksjonen gir en kort innføring i relasjonsmodellen (eller relasjonsdatamodellen) ved å henvise til lærebokas kap. 2. Bruken av primær- og fremmednøkler nevnes. Til slutt et lite eksempel.

Innhold

1.1. Kommentarer til læreboka

Les hele kapittel 2.1-2.5. Alle begrepene i figur 2.1 er essensielle for forståelsen av hvordan relasjonsdatabaser er bygd opp (unntatt begrepet 'relasjonens grad' som brukes lite). Begrepsforståelsen her er viktig for resten av kurset, ikke minst når du skal skrive spørringer i SQL (senere leksjoner), men også i de andre leksjonene i kurset (f.eks. normalisering og oversetting fra ER-diagram). Med andre ord: disse begrepene må du kunne på eksamen! Men du har fortsatt noen uker på å lære deg disse, så alt trenger ikke å sitte støpt etter å kun ha lest kapittel 2.

Vi har brukt begreper *relasjon* i dette kapittelet fordi dette er mest i trå med det matematiske begrepet, *relation*, som relasjonsmodellen bygger på. I praksis vil en relasjon være det samme som en tabell i MS Access eller andre relasjonsdatabasesystem. Det er enklere å snakke om tabeller med tupler (eller poster om du vil) og attributter enn relasjoner. I kap. 2 brukes relasjon for å unngå forvirring.

Merk deg at på norsk, for eksempel i den norske versjonen av MS Access, brukes ofte ordet relasjon misvisende som oversetting av det engelske *relationship*. Vi har valgt å oversette relationship til *sammenheng* (brukes senere i kurset). I databasefaget er det dessverre flere ord som egentlig betyr det samme, så det kan bli litt begrepsforvirring i starten. Dette går seg til etter hvert.

Merk deg ellers bruken av ulike typer nøkler (kap. 2.4), spesielt *primær- og fremmednøkler*. Disse to nøklene er essensiell for god tabelldesign.

Kap. 2.5 omtales som eget punkt under (les kommentarene der).

1.2. Valg av nøkler i tabeller

1.2.1. Primærnøkkel

Det er god databasedesign å la alle tabellene i en relasjonsdatabase ha en primærnøkkel. Primærnøkkelen må være entydig (unik) for alle tuplene i tabellen. Dette betyr i praksis at f.eks. etternavn ikke kan brukes som primærnøkkel fordi det kan forekomme flere personer med samme etternavnet i tabellen. Typisk vil en velge et entydig nummer som primærnøkkel, som ansattnr, varenr, strekkode, kundenr, kontonr eller fødselsnr (det siste krever imidlertid tillatelse fra Datatilsynet). Merk deg at slike id-nr ikke nødvendigvis trenger å være et heltall, det kan være en streng som godtar gitte kombinasjoner av tall og evt. bokstaver. Lovlige kombinasjoner blir attributtets domene.

Det enkleste er å velge et enkeltattributt som primærnøkkel fremfor en sammensatt nøkkel. Av databasetekniske årsaker bør ikke en sammensatt nøkkel være for lang, bl.a. fordi primærnøkkelen ofte brukes til å lage indeksfiler (læreboka kap. 8). En sammensatt nøkkel med to attributter går bra, men man bør kanskje stoppe der.

I mangel på en naturlig entydig nøkkel i en tabell velges ofte det boka kaller en *surogatnøkkel*. Dette er et ekstra attributt en legger til tabellen som kun har til funksjon å være primærnøkkel. For å gjøre denne unik velger man ofte å la datatypen til nøkkelen være en tellervariabel (eng. counter), som kan være et heltall som telles fortløpende oppover etterhvert som en legger inn nye tupler i tabellen. Typisk vil databasesystemet tilby funksjonalitet for å støtte opp rundt slike tellere (f.eks. at databasesystemet automatisk legger inn en verdi i feltet ved innsetting av et nytt tuppel).

1.2.2. Fremmednøkkel

En fremmednøkkel gir en referanse til en primærnøkkel i en annen tabell eller i samme tabell (rekursiv sammenhengstype – læreboka kap. 6).

Husk følgende:

- Et attributt kan både være en primærnøkkel og en fremmednøkkel
- Et attributt kan være en del av en sammensatt fremmednøkkel
- En tabell har bare en primærnøkkel, men kan ha mange fremmednøkler
- En fremmednøkkel må alltid referere til en primærnøkkel

I datamodelleringsdelen av kurset kommer vi tilbake til bruk av fremmednøkler.

1.3. Referanseintegritet

Dataintegritet i kap. 2.5 er også viktig. *Referanseintegritet* sammen med bruk av fremmednøkler kan brukes til å unngå feil (inkonsistens) i en relasjonsdatabase. I eksemplet i figur 2.8 kan databasesystemet sjekke referanseintegriteten både ved innsetting og sletting av tupler i tabellen Fagvalg:

1.3.1. Innsetting av et nytt tuppel i Fagvalg

- Ved innsetting av et nytt fagvalg-tuppel kan databasesystemet sjekke om studnr som lagres finnes i Student-tabellen (heretter Student). D.v.s. det er ikke lov å registrere et fagvalg for en student som ikke er registrert i Student. Eller sagt på en annen måte: En student må være registrert (lagret) i Student før en kan legge inn annen informasjon om vedkommende, som hvilke fag hun / han tar.
- Videre kan databasesystemet sjekke om fagkode finnes i Fag-tabellen. Dette blir samme argumentasjon som over. En student kan ikke ta et fag som ikke er registrert i Fag først.

1.3.2. Sletting av tupler fra tabellene Student, Fag og Fagvalg

- Hvis en ønsker å slette et student-tuppel fra Student er ikke dette mulig hvis studenten har tatt fag registrert i Fagvalg. Årsaken er at det vil bryte referanseintegriteten hvis en tillatter at Fagvalg inneholder studnr som er slettet fra Student. Hvis en skal slette en student uten å bryte refereanseintegriteten må en først slette alle tuplene i Fagvalg som refererer til studenten, så tuplet i Student.
- Det samme forholder gjelder for sletting av tupler i Fag. En kan ikke slette et fag som studenter har tatt. Da må en i så fall slette de aktuelle fagvalgene først.
- Sletting av tupler i Fagvalg er derimot problemløst. Det ødelegger ikke referanseintegriteten mellom de tre tabellene at tupler i Fagvalg slettes.

1.4. Automatisk generert primærnøkkel

Valg av primærnøkkel i en tabell kan i praksis ofte være vanskelig. Kanskje det ikke finnes noe attributt som naturlig er entydig. I tillegg er det uheldig å endre verdien på en primærnøkkel, for eksempel hvis en har valgt kundenummer som primærnøkkel og man senere ønsker å endre på kundenummeret blir det fort mye rot. Derfor tilbyr de fleste relasjonsdatabasesystemer automatisk genererte primærnøkler som brukerne ikke trenger å tenke på. Dette er som regel et heltall (integer) som DBMS sørger for å holde entydig for alle tuplene i en tabell. Velg automatisk primærnøkkel i tilfeller hvor det ikke er noen annen naturlig nøkkel.

1.5. Et lite eksempel

1.5.1. Tabeller med nøkler

La oss si vi ønsker å lage en liten relasjonsdatabase over kunder, varer og hvilke varer (ordrer) de ulike kundene kjøper / bestiller. Vi kan da opprette to tabeller; KUNDE og VARE. KUNDE-tabellen inneholder her dataene for å lage et kunderegister, mens VARE-tabellen inneholder en vareliste, eller produktliste om du vil. Tabellene kan se slik ut:

KUNDE			
kundenr	**navn**	**adresse**	**telefon**
1	Tore Mallaug	Trondheim	
2	Ole Olsen	Moss	
3	Eva Larsen	Levanger	

VARE		
varenr	**varenavn**	**pris**
1	hammer	kr 49,90
3	sag	kr 79,00
4	slagbor	kr 395,00

Understrekne attributt er valgte primærnøkler. KUNDE og VARE har ingen fremmednøkler. Datatypene til attributtene er ikke tatt med her.

Vi trenger så en sammenheng (mellom KUNDE og VARE. Vi kan lage denne som en mange-til-mange sammenhengstype. Mange-til-mange i relasjonsmodellen krever en koblingstabell som vi kaller VARESALG. Denne sammenhenger gjør det mulig for en kunde å handle null, en eller flere varer. For hver handel blir fakturadato, antallet av varer som kjøpes registrert, pluss attributtet 'betalt' som viser om antallet er betalt eller ikke (boolean datatype). Hvert tuppel i VARESALG kan være ei linje på en samlefaktura for kunden. Hver vare kan handles av ingen, en eller flere kunder. Tabellen:

VARESALG					
ID	**Kundenr***	**Varenr***	**fakturadato**	**antall**	**betalt**
1	1	3	01.09.2004	3	Ja
3	2	1	01.09.2004	200	Nei
4	1	3	02.09.2004	5	Nei
8	2	4	02.09.2004	1	Nei
9	1	4	02.09.2004	1	Nei

I tabellen VARESALG har jeg latt databasesystemet (i dette tilfellet var det MS-Access) generere en primærnøkkel automatisk, her ID, siden det ikke finnes noen naturlig entydig

nøkkel her (kombinasjonen av Kundenr + Varenr er ikke entydig siden en kunde kan bestille samme varen flere ganger).

* indikerer fremmednøkler. Dette betyr at hvert tuppel i VARESALG har en sammenheng til både en kunde i KUNDE og en vare i VARE. Fremmednøklene Kundenr og Varenr tilsvarer primærnøklene i henholdsvis KUNDE og VARE.

1.5.2. Dataintegritet

***Dataintegritet*:** Hvis vi inkluderer referanseintegritet i databasen har dette innvirkning på hvordan sletting av tupler i KUNDE og VARE må foregå. Referanseintegriteten medfører at det ikke er mulig å slette et tuppel som refereres av andre tupler i databasen. Her betyr det at det ikke er mulig å slette en kunde i KUNDE som har kjøpt varer. Kjøpene kunden har registrert i VARESALG må slettes først. Det er heller ikke mulig å slette en vare som kunder har kjøpt.

1.5.3. Bruk av data i tabellene

Tabellene i databasen kan f.eks. brukes til å skrive ut fakturaer til kunder. F.eks. en faktura til kunde Tore Mallaug over ubetalte kjøp.

KUNDEFAKTURA

kundenr	*navn*	*adresse*	*varenr*	*varenavn*	*pris*	*ID*	*varenr*	*fakturadato*	*antall*	*sum*
1	*Tore Mallaug*	*Trondheim*								
			4	slagbor	kr 395,00	9	4	02.09.2004	1	kr 395,00
			3	sag	kr 79,00	4	3	02.09.2004	5	kr 395,00
						Total sum:				kr 790,00

Utskriften over er laget i rapportgeneratoren til MS-Access med data fra tabellene. Poenget med dette lille eksempelet er å vise:

- Det er mulig å skrive ut bare et utvalg av tuplene fra tabellene. Her er det skrevet ut kun tupler for kundenr=1 hvor 'betalt'-attributtet i VARESALG er lik "Nei" (usann / false).
- Summeringsfelt (også kalt aggregat-felt) trenger i prinsippet ikke å lagres i tabellene. Disse kan regnes ut ved å summere attributtverdier i databasen. Så 'sum' og 'totalt sum' er altså ikke lagret i tabellene, men regnet ut på grunnlag av antall*pris. I enkelte tilfeller kan det riktignok være praktisk å lagre enkelt utregninger i databasen, f.eks. hvis en sum brukes veldig ofte av mange sluttbrukere av databasen.