Relation modell: En database som oppbevarer en oppsamling av tabeller hvor hver tabell oppbevarer en verdi.

Relation: En relasjon er en terminologi som beskriver tilstanden til verdi når den er oppbevart i en tabell. Kan sees som et matematisk konsept som vi kan tolke som en tabell som rett og slett, oppbevarer verdier.

Relationsdatabase: En database som oppbevarer en samling av relasjoner.

Domain: En sett av atomiske verdier, under tabellen av navn, så kan vi ikke ha mengde av navn. Forteller hvilken datatype som godtas under som verdier. (Navn sin domene godtar strenger f.eks).

Attribute: Ett navn for en rolle angitt av domene ("Kolonne navnet"). Hvis "A" er en attribute, skriver vi ned dom(A) = D, til å uttrykke at A er en rolle som blir spilt av domene D.

Relation signature: A named set of attributes R = (A1, A2, ….., An), where R is relations name, "n" is called relations degree or arity(aritet). Husk dette med "Modeller" i 1150, og signaturen til en gitt modell med aritet n osv.

Signatur: Navnet på kolonnene i en gitt tabell. Det vil være (EMP#, Name, Bdate) osv.

Tuppel: Angir radene i en gitt tabell.

Supernøkkel: Mengde med attributter som unik beskriver enhver rad. Fødselsdato og Personnummer er et eksempel på dette, to personer kan ikke har samme fødselsdato og personnnummer, men personer kan ha samme fødselsdato. Supernøkler er rett og slett attributter som gjør at vi kan skille mellom andre relasjoner i en og samme tabell.

Kandidatnøkkel: Refereres som en minimal supernøkkel, altså hvis vi har en kombinasjon av fødsels og personnummer, så er dette en minimal supernøkkel da om vi fjerner personnummer, så har vi ikke lenger en supernøkkel. Alle supernøkler er også kandidatnøkler. Men om vi har {Adresse, fødsel, personnummer} så har vi ikke minimal supernøkkel da vi har lagt til en nøkkel-attributt som ved fjerning, ikke nødvendigvis forårsaker at supernøkkelene blir borte. Ved å fjerne Adresse så har vi fremdeles en supernøkkel, men poenget er rett og slett at kandidatnøkkel har færrest mulige nøkkel-attributter.

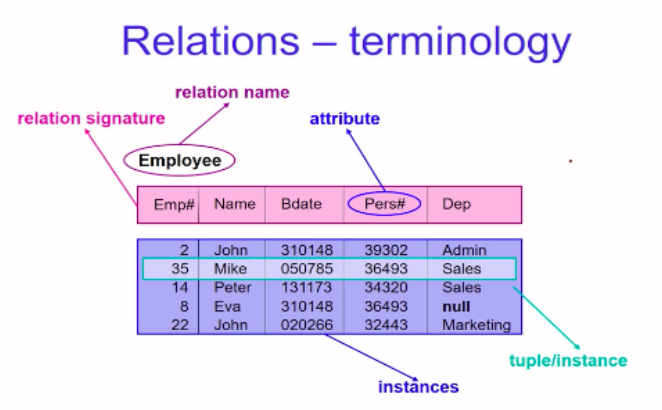
Nøkkel-attributt: En nøkkel attributt er en attributt som forekommer minst en gang i enhver supernøkkel. Fødselsdato og personnummer kan sees som nøkkel-attributter.

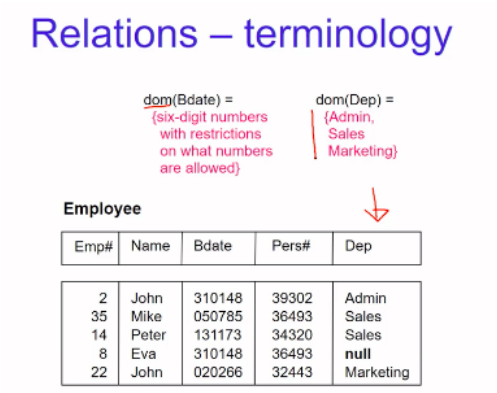
Primærnøkkel: En primærnøkkel kan sees som en kandidatnøkkel med færrest attributter og som er mest mulig unik, den nøkkelen som gjør at vi kan lettest mulig skille ut andre.

Fremmednøkkel: Kan sees som en nøkkel som viser til en annen tuppel i en annen relasjonstabell. Vi har en relasjon (Jobb) som peker på (Firma) som er en annen relasjon.

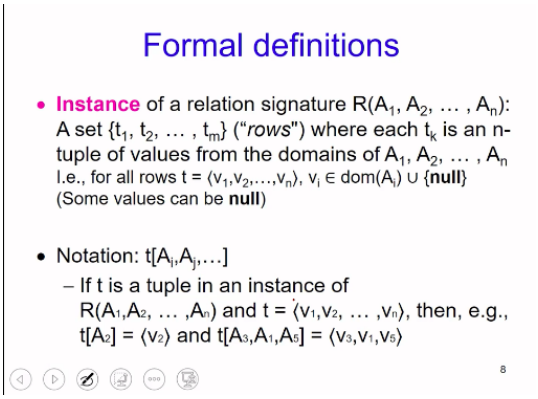
Selection: Velger de radene basert på en gitt betingelse. F.eks så velger jeg alle fotballag i en sportskategori. Eller alle lagene som har vunnet champions league.

Projection: Velger de kolonnene basert på en gitt betingelse. Samme tankegang som nevnt over.





Over så ser man hvordan man kan beskrive en gitt domene og hvilke verdier den kan oppbevare. F.eks, så kan dom(Dep) kun oppbevare (Admin, sales og marketing), kan ikke inneholde noe annet enn det.



Minner om modeller fra 1150, dette med relasjoner, aritet osv.

Ting som er viktig å vite:

- Rekkefølgen av tupler i en instans, er vilkårlig. Rekkefølgen har ingen rolle, akkurat som i 1150.
- Rekkefølgen av verdiene i en tuppel er ikke vilkårlig, altså rekkefølgen har en rolle
- Vi kan ikke ha identiske tupler i en og samme instans. Husk dette med mengder og tupler i 1150, med mengder så var det mye som var tillatt som ikke er det med tupler.
- To attributter i en relasjonell signatur, kan ha samme domene men ikke samme navn.

Relational data vs. Tabular data

(mathematical constructs vs. visual constructs)

Relational Model	Tabular Data
Relation	Table
Tuple	Row
Attribute	Column
Domain	Column data type
Arity	Number of columns
Signature (i.e., intension)	Table header
Instance (i.e., extension)	All rows (except header)

Keys and key attributes

Employee

Emp#	Name	Bdate	Pers#	Dep
2	John	310148	39302	Admin
35	Mike	050785	36493	Sales
14	Peter	131173	34320	Sales
8	Eva	310148	36493	**null**
22	John	020266	32443	Marketing

- Employees cannot to have the same Emp#
- Employees cannot have the same birth id (Bdate + Pers#)

F. eks: Jeg kan ikke legge til en ny person med Emp: 2, da det ligger der allerede. Samme gjelder med fødselsnummer og personnummer kombinert.

Keys and key attributes

- **Super key**: A combination (subset) X of attributes $\{A_1, A_2, \ldots, A_n\}$ such that if t and u are to tuples where t≠u, then t[X] ≠u[X]
  Note: The relation signature is always a super key
- **Candidate key**: A *minimal* super key
  I.e.: Removing any attribute causes the remaining attributes to no longer be a super key
- Super keys are used to express integrity rules

- **Primary key**: A selected key from the candidate keys
  All relations must have exactly one primary key

- **Kay attribute**: Attribute that is included in (at least) one candidate key

Employee

Emp#	Name	Bdate	Pers#	Dep

- A candidate key is usually marked with one line
- If there is more than one candidate key, the primary key is marked with double line

Foreign keys

Foreign key:
One or several attributes which *point/refer to* tuples in another relation

Employee

Emp#	Name	Bdate	Pers#	Dep
2	John	310148	39302	Admin
35	Mike	050785	36493	Sales
14	Peter	131173	34320	Sales
8	Eva	310148	36493	Marketing
3	Ida	190267	39421	Marketing
17	Odd	010482	34323	Sales

Project

Pname	Participant
Budget2018	2
Budget2018	14
Budget2018	3
Campaign0903	14
Campaign0903	17

Relational databases - definitions

- **Database schema** (often just "**schema**"):
  Collection of relation signature + integrity rules
- **Database instance**:
  Collection of relation instances
- **Database** = schema + instance

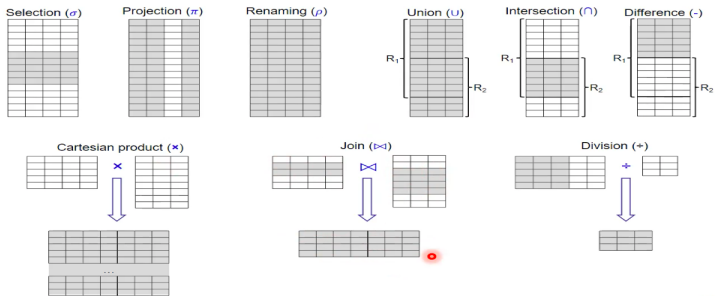
Abstract algebra	Algebra of arithmetic	Relational algebra
Operands	Variables (e.g., *x*) and constants (e.g., *2*)	Relations (e.g., *Person(Name,Age)* and associated instances)
Operations	Addition (+), subtraction (-), etc.	Selection (σ), projection (π), etc.
Properties of operations	E.g., addition is commutative	E.g., selection is commutative
Expressions*	((x + 7)/(2 / -3)) + x	$\pi_{Name}(\sigma_{Age \geq 20 \wedge Age < 30}(Person))$

Relational operations

Basic	Derived and Auxiliary	Extended

• Selection (σ)
• Projection (π)
• Cartesian product ($\times$)
• Set operations
  • Union ($\cup$)
  • Difference (- or \)
	• Renaming (ρ)
	• Join ($\bowtie$)
	  – Theta, equi, natural, etc.
	• Set operations
	  – Intersection ($\cap$)
	  – Division ($\div$)
		• Aggregate functions and grouping (γ)
		• Generalized projection
		• Sort (τ)
		• Duplicate elimination (δ)

Relational operations – graphical overview



Selection

- Return all tuples that satisfy a condition
- Notation: $\sigma_C(R)$
  - C: boolean condition that output tuples should satisfy
    - =, ≠, <, >, ≥, ≤, AND, OR, NOT, …
  - R: input relation
  - Output schema: same as input schema (i.e., R's schema)
- Selection σ is commutative

  $\sigma_{C1}(\sigma_{C2}(R)) = \sigma_{C2}(\sigma_{C1}(R))$
  - A sequence of select operations may be applied in any order
  - A cascade of select operations may be replaced by a single selection with a conjunction of all the conditions

Selection example

$\sigma_{Age > 22}(Employee)$

EID	Name	Age	Department
6	Ann	24	SD
25	John	23	IT

$\sigma_{Age > 22}$

Employee

EID	Name	Age	Department
19	Joe	22	OM
6	Ann	24	SD
3	Paul	21	OS
25	John	23	IT
5	Peter	21	SD

Projection

- Return specific attributes of all tuples
- Notation: $\pi_{A1, \ldots, An}(R)$
  - Input schema: R(B1, …, Bm)
  - A1, …, An: list of attributes to project onto, s.t. {A1, …, An} ⊆{B1, …, Bm}
  - Output schema: S(A1, …, An)
- The project operation *removes any duplicate tuples* (the result of project is a relation, i.e., a set of tuples)
- Projection is *not* commutative

Projection example

$\pi_{Name, Department}(Employee)$

Name	Department
Joe	OM
Ann	SD
Paul	OS
John	IT

$\pi_{Name, Department}$

Employee

EID	Name	Age	Department
19	Joe	22	OM
6	Ann	24	SD
	Paul	31	OS
25	John	23	IT
5	Ann	21	SD

Cartesian product

- Used to combine tuples from two relations in a combinatorial fashion
- Return the concatenation of every tuple in R_1 with every tuple in R_2
- Notation: $R_1 \times R_2$
  - Input schemas: $R_1(A1,\ldots,An)$, $R_2(B1,\ldots,Bm)$
  - Condition: $\{A1,\ldots,An\} \cap \{B1,\ldots,Bm\} = \emptyset$
  - Output schema: $S(A1,\ldots,An,B1,\ldots,Bm)$
- If R_1 has n_{R1} tuples and R_2 has n_{R2} tuples, then $R_1 \times R_2$ will have $n_{R1} * n_{R2}$ tuples
- Very rare in practice; mainly used to express joins

Cartesian product example

Employee×Department

EID	Name	Age	Department	DID	DName	Address
19	Joe	22	OM	SD	Software Development	Addr_1
6	Ann	24	SD	OM	Online Marketing	Addr_2
19	Joe	22	OM	OS	Online Sales	Addr_3
6	Ann	24	SD	SD	Software Development	Addr_1
19	Joe	22	OM	OM	Online Marketing	Addr_2
6	Ann	24	SD	OS	Online Sales	Addr_3

×

Employee

EID	Name	Age	Department
19	Joe	22	OM
6	Ann	24	SD

Department

DID	DName	Address
SD	Software Development	Addr_1
OM	Online Marketing	Addr_2
OS	Online Sales	Addr_3

Union

- Return the union of all the tuples in R_1 and R_2
- Notation: $R_1 \cup R_2$
  - Input schemas: R_1 and R_2 have the same schema, with attributes A1,…,An
    - i.e. *union-compatible*
  - Output schema: the same as the input relations
- Union is commutative and associative

Union example

Group1 ∪ Group2

GroupID	Lead
5009	Per
6002	Peter
3001	John
5045	Joe
4000	Ann

∪

Group1

GroupID	Lead
5045	Joe
4000	Ann
6002	Peter

Group2

GroupID	Lead
5009	Per
6002	Peter
3001	John

Difference

- Return the the tuples in R_1 that are not in R_2
- Notation: $R_1 - R_2$ (or $R_1 \setminus R_2$)
  - Input schemas: R_1 and R_2 are union-compatible
  - Output schema: the same as the input relations
- Difference is *not* commutative, i.e., in general $R - S \neq S - R$

Join

- One of the most important and well-studied operations for any relational database with more than a single relation
- Allows to *combine* related tuples from various relations
- The sequence of *cartesian product* followed by *selection* is often used to identify and select related tuples from various relations
  - Join combines this sequence into a single operation
- Comes in various flavors
  - Theta join
  - Natural join
  - Equi-join
  - Semi-join
  - Inner join
  - Outer join
  - Anti-join