Nettverking

video linker:

https://www.youtube.com/watch?v=Rck3BALhI5c

https://www.youtube.com/watch?v=PpsEaqJV\_A0

https://www.youtube.com/watch?v=Vc16CCAAz7Q

https://www.youtube.com/watch?v=Rck3BALhI5c

https://www.youtube.com/watch?v=UrG7RTWIJak

https://www.youtube.com/watch?v=kHceSB1X-R0

https://www.youtube.com/watch?v=7oBjpf4NUms

**DNS – Domain name system**

«IP’n» til en nettside, når man søker opp en nettside sender maskinen din til en RNS (recursive name server) som sender tilbake ipen som man trenger for å accesse siden. Om DNS ikke har IPen skjekker den 12 root servere som har den. Denne ipen kan også bli lagra i en chache lokalt slik at den ikke trenger å requeste ipen hver gang man er på nett.

**TCP/IP – Transmission control protocol/internet protocol**

Er satt opp I lag, øverste lag er application som er brukergrensesnittet. Eksempler på disse er http, smtp og ftp. Neste «lag» er transoprt layer, TCP

dette laget deler opp infoen i packeges sånn at de kan sendes raskere dit den skal på nettet. Denne inneholder også hvordan dataen er satt sammen slik at den kan samles og leses igjen senere.

Neste lag er INTERNET LAYER denne inneholder hvor pakken kom fra og hvor den skal. Siste lag er NETWORK LAYER som assigner ting som MAC adresse

**Hva er en brannmur?**

En brannmur regulerer hva som kan koble seg ut eller inn på PC'en gjennom nettverket.

F.eks. kan man sette visse programmer til å ikke kunne kommunisere utad.

**//Kan du nevne et par fordeler et firma kan ha ved å legge applikasjoner på en søkeløsning,**

**fremfor å gjøre dem på egne servere?**

Skalering, slipper stor investering i forkant, slipper drifting av grunnsoftware, tilgang overalt,

dataene er der om harddisken din krasjer.

Det koster mye penger og kunnskap å sette opp maskinvare og serverprogramvare, og sørge for at det

skalerer riktig når trafikken øker. Derfor er det mye mer praktisk å leie kapasitet fra de store

leverandørene hvor man kun betaler for faktisk bruk, og man trenger ikke sette opp noe maskinvare

og installere programvare selv. Tilbyderen har også da mer ansvar for sikkerheten.

**Hva er vranglås(deadlock) i databasesammenheng?**

Når en transaksjon som låser a, er avhengig av b, men en annen transaksjon har låst b og avhegig av a.

Transaksjonene er da i vranglås fordi verken av de kan utføres.

**Hvorfor er implementasjon av nettverk organisert i lag?**

Hvert lag har sitt ansvarsområde, og sine protokoller. Det øker fleksibiliteten både ved bruk,

og kanskje særlig ved endring. Endringer i en protokoll kan gjøres i ett lag, uten å røre ade andre.

Innføring av nye protokoller, som f.eks. IPv6 berører bare ett lag(i hovedsak, av og til må en også

gjøre justeringer på lagene rundet.). Dersom en lager et program som skal kommunisere over nettet på

en annen måte er det applikasjonslaget i TCP/IP stacken tilbyr av tjenester, er det også mulig å

programmere dirkete mot en av lagene ned i stacken. Det er generelt ofte lurt å ha en viss lagdeling i programmer.

På de fleste hjemmenttverk får PCer etc. automatisk en Ipv4 adresse. Når du på netter sjekker hvilken Ipv4-

adresse du har, er det oftest en annen en det PCen din har intern i netterverket. Hva heter denne mekanismen som

gjør at intern og ekstrern Ipv4 adresse er forskjellige, og hva er hensikten med denne mekanismen?

Der er flere ting involvert, men hovedmekanismen som gjør dette er NAT-ing(Network Address Transfer),

er en funksjonalitet som ofte finnes i routere.

**Forklar oppdelinga av IP-adresse i en nettverksdel og en host-del:**

Nettverksdelen sier hvilket nettverk PCen tilhører, det er den som routes på nettet. Host-delen brukes

til å nummerere PCer innenfor et nettverk, slik at IP-adressen blir unik.

**Hva er den største fordelen ved å gå over fra Ipv4 til IPv6?**

Antall adresser.

**En pC har IPv4 adressen 172.16.254.1/24. Hvilke av PCene i) til v) er i samme nettverk som denne?**

i)172.16.254.5 ii)172.16.1.1 iii)172.16.254.4 iv)172.6.254.128 v) 72.16.254.1

i og iii. De tre første feltene er like

**Forkort denne IPv6 adressen så mye som mulig: 2001:0000:0000:aaaa:0001:0000::**

2001:0:0:aaaa:1::

En del av Ipv6 adresseomeårder er reservert for spesiell bruk. En av dem er link-local, som brukes

internt i nettverket. Om link-local står det: <In IPv6, the block fe80/10 is reserved for IP adress

autoconfiguration. The implementation of these link-local adresses is mandatory, as various functins

of the IPv6 protocol depends on them>

Er adressen fe9:3456:0:af03::ffe5:2f4 ei link-local adresse etter denne definisjonen?

fe80::/10: de første bitene er;; 1111 1110 10

feb9::3456: de første bitene: 1111 1110 10

Disse stemmer overens så det er en link-local adresse.

**Forklar hva ei MAC adresse er, og hvordan denne brukes.**

MAC = Media Access Control adress. Det er ei 48 eller 64 bit adresse på ethvert nettverkskort.

Den skal være unik på verdensbasis. I nettverk kalles den en fysisk adresse, ettersom den er

fysisk koblet til nettverkskort. MAC-adressen brukes til adressering internt i et nettverk.

Switcher bruker MAC-adressa til adressering, i motsetning til routere som bruker IP-adressa.

**Forkort denne IPv6 adressen 20a1: 00b8: 0bbf: 000e:0000:0000:0000:0d00**

20a1:b8:bbf:e::d00

**Gitt adresse 2001:0DB8:0000:CD30::/60**

**Hvilke av de to(ingen, en eller begge) skrivemåtene nedenfor er en gyldig skrivemåte for denne adressen:**

I. 2001:0DB8::CD30:0:0:0:0/60 Gyldig. Det er bedre å bruke :: på slutten, men denne er gyldig

II. 2001:0DB8:0:CD3/60 Ugyldig, har bare 4 grupper og ingen ::

**Adressetype Link-Local unicast er definert slik: FE80::/10. Hvilke av adressene nedenfor er en gyldig Link-**

**Local unicast adresse**

a. FF80::/64 = 1111 1111 1000 0000.... Ugyldig, de 10 første bitene matcher ikke

b. FE91::7/48 = 1111 1110 1001.... Gyldig

c. FE80:FE80:0000/64 = 1111 1110 1000... Gyldig

De 10 første bitene skal være lik det som er angitt i FE80::/10 -> Skrevet binært: 1111 1110 1000......

a. FF80::/64 =1111 1111 1000 0000.... Ikke denne. De 10 første bit stemmer ikke

b. FE91::7/48 = 1111 1110 1001.... Dette er en gyldig Link-Local unicast adesse

c. FE80:FE80:0000/64 = 1111 1110 1000... Ikke denne. Her stemmer de 10 første bitene, men det er ei

ugyldig IPV6 adresse fordi den ikke er 128 bit lang.

Svar: b.

**Det er satt av forskjellige adresseomeråder i IPv6, for forskjellige typer adresser.**

**En av dem er <<Global Unicast>>, som har fått adresseomerådet definert ved 2000::/3.**

**a)Hva betyr <<Global Unicast>>, hva brukes den til?**

Det er en global entydig IPv6 adresse, som brukes for en til en kommunikasjon.

Den er beskrevet i RFC 3587.

**b)Er denne IPv6 adresse av type <<Global Unicast>>: 3f01: 0db8: b000: 0001:0000:0000:0000:0200**

Ja, 2000::/3 betyr at de 3 første bitene må stemme overens med de tre første bitene i 2000,

så det er en global unicast adresse:

2000 = [0,0,1,x], [x,x,x,x], [x,x,x,x], [x,x,x,x]

3f01 = [0,0,1,1], De tre første stemmer overens.

**Skriv 3f01: 0db8: b000: 0001:0000:0000:0000:0200 så kort som mulig:**

3f01:db8:b000:1:200

**I dette nettverket er det 8 byte som gir adressen til nettverket.**

**Hvordan vil du angi det for adressen overfor?**

Med slash 64 etter adressen. 3f01:db8:b000:1:200/64

**Hvordan virker NAT-ing:**

Netwrok Adress Transfer. NATing gjør at selv om du er koblet til et privat nettverk-slik at ip-adressen

din ikke routes på internett- kan du allikevel <<snakke>> med datamaskiner via internett. Når du prøver å nå ei ip adresse utenfor det private nettverket du er i, vil routeren din endre avsenderadresse

på ip-pakka før den sendes videre. Den endres til roturenes IP-adresse som er Public og kan routes på internett.

I tillegg legger den på et unikt portnr, for å skille mellom de forskjellige PCer og programmer som

skal routes tilbake til. Routeren holder rede på hvor svaret skal sendes tilbake, for konvertering av ip-adresse til protnr når svaret kommer.

**Hvorfor er det mindre behov for NATing i IPv6 nettverk?**

Hovedårsaken til bruk av private IPv4 nettverk, er at det er mangel på adresser. Fordi det ikke er

mangel på IPv6 adresser, blir det dermed mindre bruk av private nettverk, og mindre behov for NATing.

Example of a step by step normalization[[edit](https://en.wikipedia.org/w/index.php?title=Database_normalization&action=edit&section=4" \o "Edit section: Example of a step by step normalization)]

Normalization is a database design technique, which is used to design a [relational database](https://en.wikipedia.org/wiki/Relational_database" \o "Relational database) table up to higher normal form.[[9]](https://en.wikipedia.org/wiki/Database_normalization" \l "cite_note-9) The process is progressive, and a higher level of database normalization cannot be achieved unless the previous levels have been satisfied.[[10]](https://en.wikipedia.org/wiki/Database_normalization" \l "cite_note-:0-10)

That means that, having data in [unnormalized form](https://en.wikipedia.org/wiki/Unnormalized_form" \o "Unnormalized form) (the least normalized) and aiming to achieve the highest level of normalization, the first step would be to ensure compliance to [first normal form](https://en.wikipedia.org/wiki/First_normal_form" \o "First normal form), the second step would be to ensure [second normal form](https://en.wikipedia.org/wiki/Second_normal_form" \o "Second normal form) is satisfied, and so forth in order mentioned above, until the data conform to [sixth normal form](https://en.wikipedia.org/wiki/Sixth_normal_form).

However, it is worth noting that normal forms beyond [4NF](https://en.wikipedia.org/wiki/4NF) are mainly of academic interest, as the problems they exist to solve rarely appear in practice.[[11]](https://en.wikipedia.org/wiki/Database_normalization#cite_note-11)

*The data in the following example were intentionally designed to contradict most of the normal forms. In real life, it is quite possible to be able to skip some of the normalization steps because the table doesn't contain anything contradicting the given normal form. It also commonly occurs that fixing a violation of one normal form also fixes a violation of a higher normal form in the process. Also one table has been chosen for normalization at each step, meaning that at the end of this example process, there might still be some tables not satisfying the highest normal form.*

**Initial data**[[edit](https://en.wikipedia.org/w/index.php?title=Database_normalization&action=edit&section=5" \o "Edit section: Initial data)]

Let a database table exist with the following structure:[[10]](https://en.wikipedia.org/wiki/Database_normalization#cite_note-:0-10)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Title** | **Author** | **Author Nationality** | **Format** | **Price** | **Subject** | **Pages** | **Thickness** | **Publisher** | **Publisher Country** | **Publication Type** | **Genre ID** | **Genre Name** |
| Beginning MySQL Database Design and Optimization | Chad Russell | American | Hardcover | 49.99 | |  | | --- | | MySQL | | Database | | Design | | 520 | Thick | Apress | USA | E-book | 1 | Tutorial |

For this example, it is assumed that each book has only one author.

As a prerequisite to conform to the relational model, a table must have a [primary key](https://en.wikipedia.org/wiki/Primary_key), which uniquely identifies a row. Two books could have the same title, but an ISBN number uniquely identifies a book, so it can be used as the primary key:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ISBN#** | **Title** | **Author** | **Author Nationality** | **Format** | **Price** | **Subject** | **Pages** | **Thickness** | **Publisher** | **Publisher Country** | **Publication Type** | **Genre ID** | **Genre Name** |
| 1590593324 | Beginning MySQL Database Design and Optimization | Chad Russell | American | Hardcover | 49.99 | |  | | --- | | MySQL | | Database | | Design | | 520 | Thick | Apress | USA | E-book | 1 | Tutorial |

**Satisfying 1NF**[[edit](https://en.wikipedia.org/w/index.php?title=Database_normalization&action=edit&section=6" \o "Edit section: Satisfying 1NF)]

To satisfy [First normal form](https://en.wikipedia.org/wiki/First_normal_form" \o "First normal form), each column of a table must have a single value. Columns which contain sets of values or nested records are not allowed.

In the initial table, **Subject** contains a set of subject values, meaning it does not comply.

To solve the problem, the subjects are extracted into a separate **Subject** table:[[10]](https://en.wikipedia.org/wiki/Database_normalization" \l "cite_note-:0-10)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Book** | | | | | | | | | | | |
| **ISBN#** | **Title** | **Format** | **Author** | **Author Nationality** | **Price** | **Pages** | **Thickness** | **Publisher** | **Publisher country** | **Genre ID** | **Genre Name** |
| 1590593324 | Beginning MySQL Database Design and Optimization | Hardcover | Chad Russell | American | 49.99 | 520 | Thick | Apress | USA | 1 | Tutorial |

|  |  |
| --- | --- |
| **Subject** | |
| **ISBN#** | **Subject name** |
| 1590593324 | MySQL |
| 1590593324 | Database |
| 1590593324 | Design |

A foreign key column to the **Subject**-table is added, which refers to the primary key of the row from which the subject was extracted. The same information is therefore represented but without the use of non-simple domains.

Instead of one table in [unnormalized form](https://en.wikipedia.org/wiki/Unnormalized_form" \o "Unnormalized form), there are now two tables conforming to the 1NF.

**Satisfying 2NF**[[edit](https://en.wikipedia.org/w/index.php?title=Database_normalization&action=edit&section=7" \o "Edit section: Satisfying 2NF)]

The **Book** table has one [candidate key](https://en.wikipedia.org/wiki/Candidate_key" \o "Candidate key) (which is therefore the [primary key](https://en.wikipedia.org/wiki/Primary_key" \o "Primary key)), the [composite key](https://en.wikipedia.org/wiki/Composite_key" \o "Composite key) **{Title, Format}**.[[12]](https://en.wikipedia.org/wiki/Database_normalization" \l "cite_note-12) Consider the following table fragment:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Book** | | | | | | | | | |
| **Title** | **Format** | **Author** | **Author Nationality** | **Price** | **Pages** | **Thickness** | **Genre ID** | **Genre Name** | ***Publisher ID*** |
| Beginning MySQL Database Design and Optimization | Hardcover | Chad Russell | American | 49.99 | 520 | Thick | 1 | Tutorial | *1* |
| Beginning MySQL Database Design and Optimization | E-book | Chad Russell | American | 22.34 | 520 | Thick | 1 | Tutorial | *1* |
| The Relational Model for Database Management: Version 2 | E-book | E.F.Codd | British | 13.88 | 538 | Thick | 2 | Popular science | *2* |
| The Relational Model for Database Management: Version 2 | Paperback | E.F.Codd | British | 39.99 | 538 | Thick | 2 | Popular science | *2* |

All of the attributes that are not part of the candidate key depend on *Title*, but only *Price* also depends on *Format*. To conform to [2NF](https://en.wikipedia.org/wiki/Second_normal_form" \o "Second normal form) and remove duplicities, every non candidate-key attribute must depend on the whole candidate key, not just part of it.

To normalize this table, make **{Title}** a (simple) candidate key (the primary key) so that every non candidate-key attribute depends on the whole candidate key, and remove *Price* into a separate table so that its dependency on *Format* can be preserved:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **Book** | | | | | | | | | **Title** | **Author** | **Author Nationality** | **Pages** | **Thickness** | **Genre ID** | **Genre Name** | ***Publisher ID*** | | Beginning MySQL Database Design and Optimization | Chad Russell | American | 520 | Thick | 1 | Tutorial | *1* | | The Relational Model for Database Management: Version 2 | E.F.Codd | British | 538 | Thick | 2 | Popular science | *2* | | |  |  |  | | --- | --- | --- | | **Format - Price** | | | | **Title** | **Format** | **Price** | | Beginning MySQL Database Design and Optimization | Hardcover | 49.99 | | Beginning MySQL Database Design and Optimization | E-book | 22.34 | | The Relational Model for Database Management: Version 2 | E-book | 13.88 | | The Relational Model for Database Management: Version 2 | Paperback | 39.99 | |

Now, the **Book** table conforms to [2NF](https://en.wikipedia.org/wiki/Second_normal_form).

**Satisfying 3NF**[[edit](https://en.wikipedia.org/w/index.php?title=Database_normalization&action=edit&section=8" \o "Edit section: Satisfying 3NF)]

The **Book** table still has a transitive functional dependency ({Author Nationality} is dependent on {Author}, which is dependent on {Title}). A similar violation exists for genre ({Genre Name} is dependent on {Genre ID}, which is dependent on {Title}). Hence, the **Book** table is not in 3NF. To make it in 3NF, let's use the following table structure, thereby eliminating the transitive functional dependencies by placing {Author Nationality} and {Genre Name} in their own respective tables:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Book** | | | | | |
| **Title** | **Author** | **Pages** | **Thickness** | **Genre ID** | ***Publisher ID*** |
| Beginning MySQL Database Design and Optimization | Chad Russell | 520 | Thick | 1 | *1* |
| The Relational Model for Database Management: Version 2 | E.F.Codd | 538 | Thick | 2 | *2* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Format - Price** | | | | **Title** | **Format** | **Price** | | Beginning MySQL Database Design and Optimization | Hardcover | 49.99 | | Beginning MySQL Database Design and Optimization | E-book | 22.34 | | The Relational Model for Database Management: Version 2 | E-book | 13.88 | | The Relational Model for Database Management: Version 2 | Paperback | 39.99 | |

|  |  |
| --- | --- |
| **Author** | |
| **Author** | **Author Nationality** |
| Chad Russell | American |
| E.F.Codd | British |

|  |  |
| --- | --- |
| **Genre** | |
| **Genre ID** | **Genre Name** |
| 1 | Tutorial |
| 2 | Popular science |