TRAFIKVERKET

SPECIFIKATION

NVDB Teknisk lösning – ID-hantering och transaktioner

Version 1.0

Publikation 2012:232

Dokumenttitel: NVDB Teknisk lösning – ID-hantering och transaktioner
Skapat av: Per Isaksson
Dokumentdatum: 2006-05-15
DokumentID:
Publikationsnummer: 2012:232
Version: 1.0

Publiceringsdatum: 2006-05-15
Utgivare: Trafikverket
Kontaktperson: Mats-Ove Mattsson

NVDB Teknisk lösning - ID-hantering och transaktioner

Ändringsförteckning:

Versionsnr	Datum	Orsak samt ändring mot tidigare version	Ansvarig
1.0	2006-05-15	Första version för publicering	Per Isaksson, Sitp

Innehållsförteckning

1 Syfte

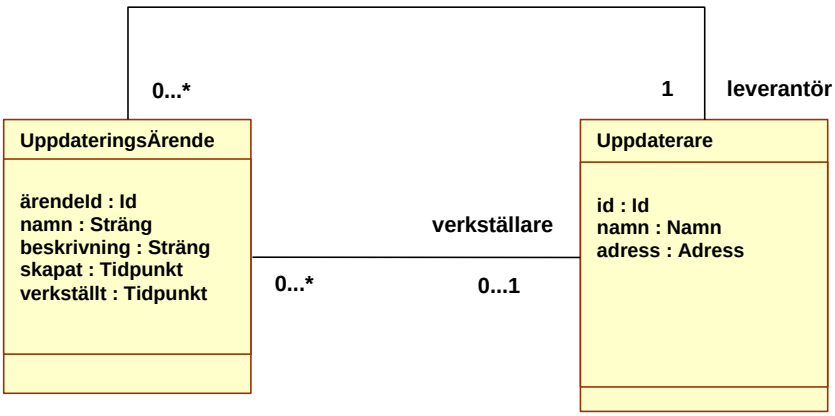
Dokumentet syftar till att specificera grundläggande principer för uppdateringsärenden och hantering av identiteter och versioner för olika slags objekt i NVDB-systemet. En förståelse av dessa principer och begrepp är mycket viktiga för olika slags NVDB-aktörer i och med att systemets externa gränssnitt beror av dessa.

2 Uppdatering

2.1 Uppdateringsärende och uppdaterare

Inom NVDB-systemet sker all uppdatering inom ramen för ett uppdateringsärende. I NVDB-systemet ingår en webb-applikation där leverantören fyller i erforderliga uppgifter varvid ärendet registreras och tilldelas ett unikt ärendeid (äid). Ett äid är ett positivt heltal (signed longword) och har utfallsrummet 0< n <=2147483647.

Uppdatering inom NVDB-systemet görs av dataleverantörer och vid behov justeras uppdateringarna i produktionscentralen av verkställaren.



I ovanstående figur visas förhållandet mellan uppdateringsärende och uppdaterare.

2.2 Uppdatering via Slussen

Inom Slussen inleds en uppdatering med en utcheckning från huvuddatabasen. I samband med utcheckningen registreras uppdateringsärendet. Identiteten för detta ärende (äid) följer med i den utcheckade informationen och hanteras internt i Slussen-applikationen.

2.3 Övrig uppdatering

Annan leverantör av data måste också registrera ett uppdateringsärende för att kunna leverera data till NVDB-systemet.

3 Principer för ID-hantering

3.1 Allmänt

En över tiden stabil och inom någon ”rymd” globalt unik identifierare har vi valt att kalla för *gid*. En inom NVDB-systemet unik gid kallas ibland även för NVDB-ID.

Det finns i princip två möjliga alternativ för generering av gid, de kan genereras lokalt direkt vid källan eller centralt i NVDB-centralen.

Central generering innebär att systemet genererar gid i samband med att leverantörens levererans av data ”checkas in” in den centrala NVDB-databasen. Genererade gid:ar måste därefter skickas tillbaka till leverantören tillsammans med leverantörens ”preliminära” identiteter så att leverantören kan föra in gid i sitt eget system. Detta är enligt vår bedömning en kostsam, svår och sårbar hantering.

Lokal generering innebär att objekten tilldelas sin gid direkt vid källan. Dataleverantören kan direkt föra in denna identitet i sitt eget system. Enligt detta alternativ behöver leverantören endast få en kvittens på att leveransen har ”checkats-in”. Detta alternativ är enligt vår bedömning en billigare, enklare och mindre sårbar hantering.

Mot bakgrund av ovanstående resonemang har vi därför valt en lokal generering av gid i NVDB-systemet.

Vid leverans av information måste en leverantör kunna identifiera sina objekt med en gid för att kunna leverera information till NVDB-systemet.

3.2 Uppbyggnad av gid

En gid är sammansatt av ett primär-id (pid) och ett sekundär-id (sid).

En pid och en sid är båda ett positivt heltal (signed longword) och har utfallsrummet 1<= n <= 2147483647. En gid består alltså av två stycken positiva heltal (pid+sid) och skrivs vid behov i textform som ”pid:sid” t.ex. ”111:222”.

För att möjliggöra en lokal generering av en gid tilldelas en uppdaterare i NVDB-systemet en eller flera unika primär-id:n. I NVDB-systemet finns en webb-applikation
som en uppdaterare använder för att reservera/registrera ett eller flera primär-id:n.

När uppdateraren tilldelats en pid kan uppdateraren börja generera gid lokalt genom att se till att varje sid är unik inom ramen för tilldelad pid. I NVDB-systemet finns inbyggd funktionalitet för automatisk generering av gid som håller reda på nästa lediga sid för aktuell pid. Denna funktionalitet ingår i till exempel Slussen.

3.3 Exempel på användning av gid

En referenslänk, en nod eller en företeelseförekomst tilldelas ett NVDB-ID (dvs en gid) i samband med att objektet föds. Detta NVDB-ID följer sedan med objektet under hela dess livslängd ”från vaggan till graven”. Detta över tiden stabila NVDB-ID som unikt identifierar ett visst objekt kallas även för OID eller objekt-id.

4 Versioner och hantering av förändringar

4.1 Versioner av objekt

För att särskilja olika versioner av objekt (med olika tillstånd) används ytterligare en gid. Denna gid benämns VID eller versions-id.

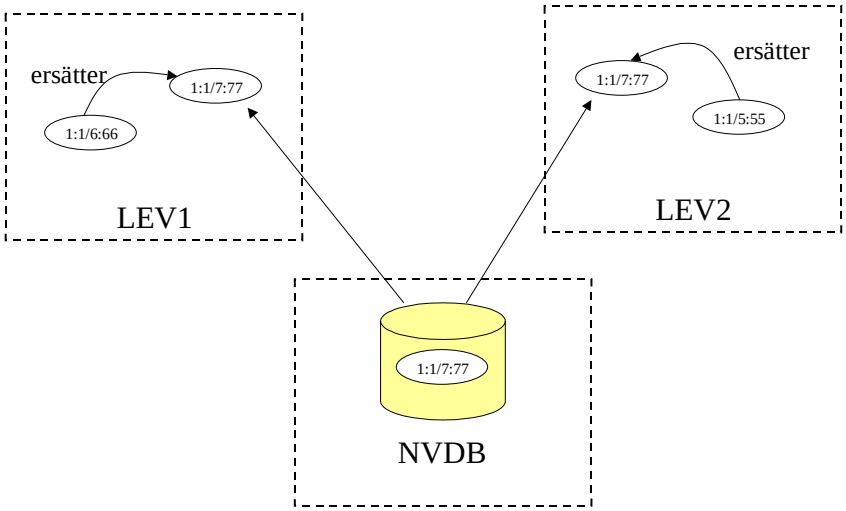
Ett globalt unikt versions-id krävs för att förhindra att flera versioner av ett och samma objekt med samma versions-id uppstår på olika håll. Detta är en följd av en optimistisk ansats där det finns en teoretisk möjligt att flera leverantörer samtidigt uppdaterar ett och samma objekt. För att kunna upptäcka sådana uppdateringskonflikter är det viktigt att varje ny version av ett objekt tilldelas en unik gid.

För en viss version av ett NVDB-objekt finns alltså både ett OID och ett VID. Om man i textform vill skriva ut både OID och VID skrivs detta som ”OID/VID” t.ex. ”111:222/333:444” Observera att pid för objektets OID inte behöver vara det samma som pid för objektets VID. En leverantör kan uppdatera ett objekt som någon annan skapat. Detta innebär att en ny version med unikt VID skapas med hjälp av uppdaterarens pid.

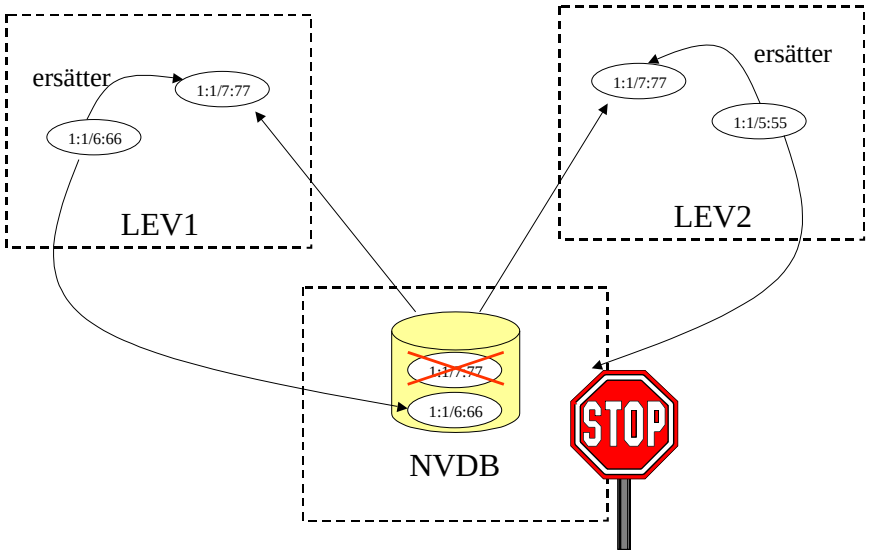
Det är viktig komma ihåg att alla referenser till ett specifikt objekt skall göras via stabila OID’n eftersom det endast är ”state-of-the-art” eller senaste versionen av ett objektet som är intressant. Ett exempel på en sådan referens är en utbredning som pekar på en specifik länk med hjälp av länkens OID.

För närvarande lagrar NVDB-systemet även alla gamla verkställda versioner för alla objekt. Detta innebär att det går att se de tidigare tillstånden, det vill säga se hur det såg ut i NVDB-databasen vid olika historiska tidpunkter.

I nedanstående bild visas hur två leverantörer båda hämtat ut ett och samma objekt. Båda leverantörerna har sedan ovetande om varandra uppdaterat objektet och skapat varsin ny objektversion för ett och samma objekt som ersätter den gamla versionen.



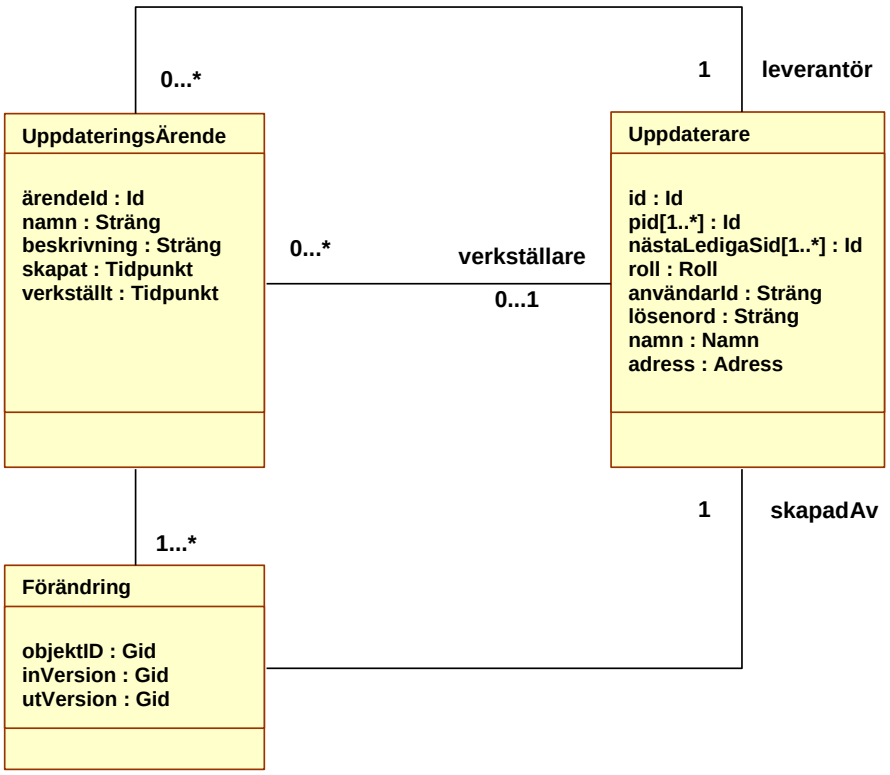
I denna bild visas att leverantören LEV1:s uppdatering är verkställd i NVDB vilken innebär att versionen ”6:66” ersätter versionen ”7:77” för objektet ”1:1”. Sedan kommer leverantören LEV2 med sin uppdatering vilken innebär att versionen ”5:55” skall ersätta versionen ”7:77” för objektet ”1:1”. I samband med verkställning kommer då systemet att upptäcka att versionen ”7:77” redan har ersatts av en ny version vilket medför att denna leverans förkastas.



Observera att mekanismen för versionshantering som beskrivs här inte skall förväxlas med hanteringen av t.ex. vägnätshistorik. Versionshanteringen är endast till för en säker och robust hantering av olika versioner för NVDB-objekten i samband med uppdatering och informationsutbyte.

4.2 Förändrings objekt

En förändring innebär att en förekomst av ett enstaka objekt skapats, modifierats eller tagits bort. För varje förändring finns det ett ”förändringsobjekt” som innehåller information om vilket objekt (OID) som avses, vilken objektversion som är ”input” och vilken objektversion som är ”output” för förändringen.



I ovanstående figur visas att det för ett uppdateringsärende kan skapas en eller flera förändringar. För varje förändring finns det information om:

- OID för objektet som förändringen avser
- Vem som skapat förändringen
- VID för version som gällde före förändringen
- VID för version som gäller efter förändringen

Om en förändring avser ”skapa” sätts inVersion till ”0:0” för att markera NULL. Om en förändring avser ”borttag” sätts utVersion till ”0:0” för att markera NULL.

Förenklat kan förändringar se ut enligt nedanstående enkla exempel för fallen skapa, modifiera och borttag av ett nätelement.

Skapa ett objekt

objektID:	1:1	pid=1, sid=1
inVersion:	0:0	pid=0, sid=0, dvs NULL
utVersion:	1:2	pid=1, sid=2

Exempel: En ny länk har skapats.

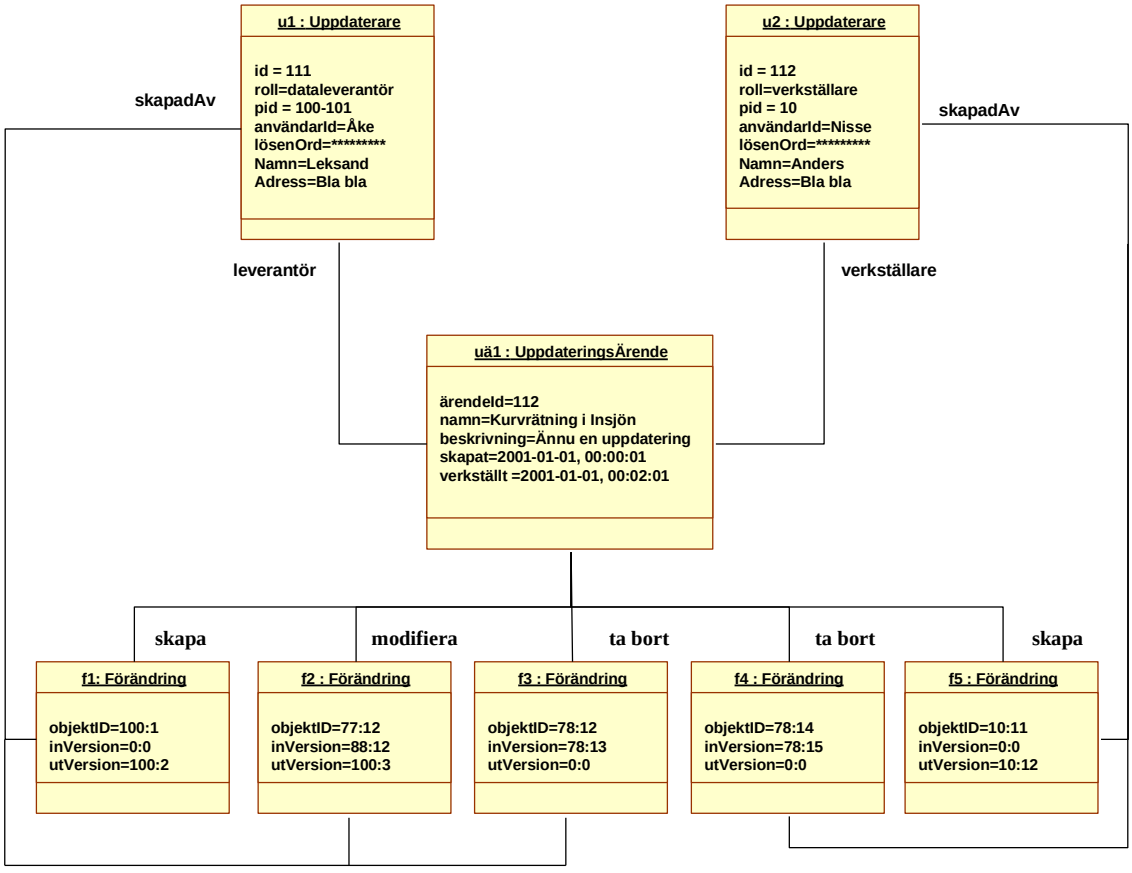
Modifiera objekt

objektID:	1:1	gid för objektets OID oförändrat
inVersion:	1:2	
utVersion:	1:3	

Exempel: Rättning av en oönskad länklängd eller att sätta tom-datum för en nod.

Borttag av objekt
objektID: 1:1
inVersion: 1:3
utVersion: 0:0 dvs NULL

Exempel: En rättning som innebär borttag av en oönskad länk.



I ovanstående figur visas ett uppdateringsärende där en leverantör u1 skapat, modifierat och tagit bort objekt (f1-f3), vid incheckning har sedan verkställaren u2 kompletterat ärendet och skapat samt tagit bort objekt (f4 och f5).

4.3 Leverans av förändringar till huvudsystemet

Ett uppdateringsärende resulterar normalt i en leverans av data till huvudsystemet. En leverans kan göras med hjälp av NVDB-utbytesformat eller NVDB-internformat. EN leverans består av en eller flera förändringar (förändringsobjekt) plus de förändrade objekten.

Har leveransen förändringar av typen skapa och modifiera måste leveransen innehålla den objektversion som är utVersion från förändringen.

Den objektversion som utgör inVersion vid modifiering och borttag behöver ej finnas med i leveransen utan förutsätts då finnas hos mottagaren och behöver endast identifieras genom objektId och inVersion.

Observera att för ett borttaget objekt räcker det att skicka ett förändringsobjekt av typen borttag som ett meddelande om att objektet tagits bort. Detta innebär att objektversionen som tagits bort ej behöver ingå i leveransen.

En leverans av förändringar får endast innehålla ett förändringsobjekt per objektID. Om t.ex. ett och samma objekt först skapas av en leverantör och sedan modifieras en eller flera gånger inom ett och samma uppdateringsärende skall leveransen ändock bara innehålla ett förändringsobjekt av typen skapa som utgör en sammanfattning av flera förändringssteg.

Följande exempel visar hur fyra förändringsobjekt kan sammanfattas av ett:

	Resultat	
ObjektID=”1:1”, InVersion=”1:0”, UtVersion=”1:2”	”1:1/1:2”	(skapa)
ObjektID=”1:1”, InVersion=”1:2”, UtVersion=”1:3”	”1:1/1:3”	(modifiera)
ObjektID=”1:1”, InVersion=”1:3”, UtVersion=”1:4”	”1:1/1:4”	(modifiera)
ObjektID=”1:1”, InVersion=”1:4”, UtVersion=”1:5”	”1:1/1:5”	(modifiera)

Sammanfattas med ett förändringsobjekt:

	Resultat	
ObjektID=”1:1”, InVersion=”1:0”, UtVersion=”1:5”	”1:1/1:5”	(skapa)

4.4 Utcheckning av förändringar från huvudsystemet

Från huvudsystemet kan man hämta alla verkställda förändringar sedan ett visst datum. Man kan få resultatet från en sådan utcheckning i NVDB-utbytesformat eller NVDB-internformat.

I NVDB-databasen kan det för aktuell tidsperiod finnas flera verkställda förändringar från flera olika uppdateringsärenden för ett och samma objekt. Vid utcheckning av förändringar sammanfattas dessa förändringar enligt ovanstående principer.

4.5 Leverans från Slussen till huvudsystemet

När man gör en leverans till huvudsystemet från Slussen skickas en fil i NVDB-internformat till huvudsystemet. Filen är en kopia av tillståndet i Slussen när leveransen skickades innehållande förändringar och aktuella objektversioner för både förändrade och oförändrade objekt i den utcheckade datamängden.

4.6 Att påföra förändringar i samband med verkställning

En leverans för ett uppdateringsärende till huvudsystemet kan göras med hjälp av NVDB-utbytesformat eller NVDB-internformat.

Innan incheckning skall leveransen kontrolleras av verkställaren. Först utför verkställaren en ny utcheckning enligt ärendets utcheckningsparametrar. Efter detta påför verkställaren levererade förändringar. En påföring av förändringar innebär här att både förändringar och deras nya objektversioner infogas i verkställarens aktuella ”dataset”.

Efter att verkställaren kontrollerat leveransens förändringar tillsammans med aktuell status i NVDB-databasen initieras incheckningen. Incheckningen innebär att NVDB-databasen uppdateras med en transaktion innehållande både förändringsobjekten och de nya objektversionerna från aktuell leverans.

4.7 Konflikthantering vid incheckning

Inom NVDB används en optimistisk metod för hantering av ut- och incheckning. Detta innebär att inga konfliktkontroller görs förrän data skall checkas in. Om NVDB-systemet hanterar förändringar som beskrivits ovan blir incheckningskontrollen enkel. Konflikt föreligger till exempel om en förändring som skall checkas in har en input VID som redan existerar som input VID i en redan

incheckad förändring. Om en sådan förändring finns så har objektet redan förändrats av någon annan.
För leverantörer som inte använder Slussen-applikationen krävs INTE att de har checkat ut data. Däremot krävs att man skapat ett uppdateringsärende och att de objekt som berörs av en uppdatering är up-to-date i leverantörens databas. Om inte så uppstår konflikt vid incheckningen av leveransen till NVDB och leveransen måste förkastas.

5 Unika länkidentiteter

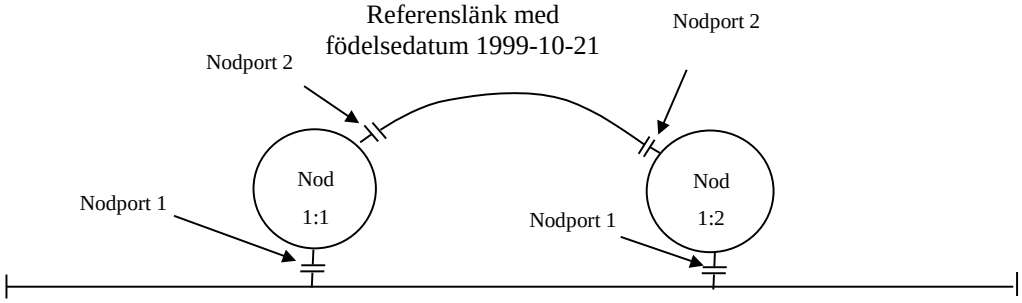
5.1 Bakgrund

Företeelser i NVDB knyts till referenslänkar, som är stabila över tiden. Det har också diskuterats ett behov av att skapa unika och stabila identiteter även för traditionella länkar, dvs. länkar enligt TK126. Eftersom länkar ej lagras i NVDB utan endast genereras vid behov måste stabila identiteter för länkar genereras i flykten. Eftersom det finns en eller flera länkar per referenslänk och det kan finnas mer än en länk mellan två noder använder vi i stället en metod som bygger på identiteter hos noder och nodernas port nummer för identifikationen av en länk.

5.2 Unika länkidentiteter

Följande entiteter används för att bygga upp länkens identitet:

- Startnodens id
- Internt löpnummer för startnodens nodport
- Slutnodens id
- Internt löpnummer för slutnodens nodport
- Födelsedatum



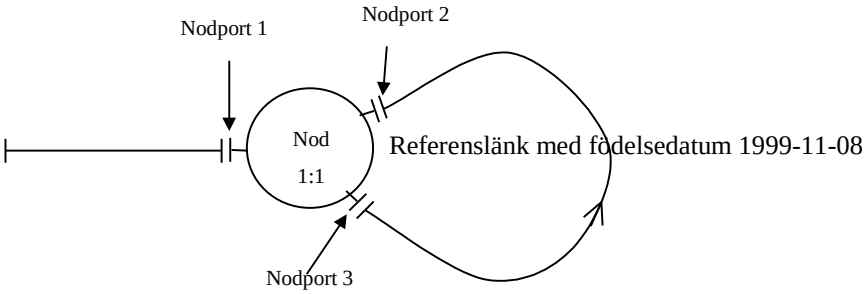
Länkarna som förbinder noderna ”1:1” och ”1:2” kan ges följande unika identitet med hjälp av entiteterna beskrivna ovan: ”1:1-1/1:2-1/1999-09-09” respektive ”1:1-2/1:2-2/1999-10-21”.

I figuren ovan visas hur de båda länkarna som finns mellan noderna ”1:1” och ”1:2” båda får olika unika identiteter. Observera dock att det för en människa fortfarande krävs en grafisk presentation för att kunna avgöra vilken länk som är vilken av de båda länkarna.

5.3 Identiteten för en länk som utgör en ögla

Ovanstående princip för att unikt identifiera en länk fungerar även för en länk som utgör en ögla. Detta innebär att ingen extra nod behövs i en ögla för att entydigt kunna identifiera länken. Om

referenslänken som utgör öglan har en riktning enligt nedanstående figur får motsvarande länk identiteten ”1:1-3/1:1-2/1999-11-08”.



En ögla som beskrivs med en referenslänk som startar i ”1:1” nodport 3 och slutar i ”1:1” nodport 2.

Observera att det för en människa är omöjligt att enbart utgående från länkidentiteten avgöra om öglans riktning, det vill säga om den går medurs eller moturs. För att avgöra detta krävs tillgång till en karta med t.ex. en pil som visar vilken riktning länken eller referenslänken har.

TRAFIKVERKET

Trafikverket, Röda vägen 1, 781 89 Borlänge.
Telefon: 0771-921 921, Texttelefon: 0243- 750 90

www.trafikverket.se