SISU dokument

# WWW med databasstöd

## – en arkitektur för informationstjänster

Peter Johansson

nr
23

# 1 Sammanfattning

Mängden information på varje server på World Wide Web (WWW) är vanligen liten idag. Dagens hantering där informationen är lagrad i filer på servern passar varken för nya interaktiva tjänster eller för hantering av stora informationsmängder. Problem med bl a administrationen uppstår. För att lösa detta behövs någon form av stödsystem. Genom att låta WWW-servern generera dokumentet från en databas när en begäran kommer, kan en databashanterare användas för att hantera all information. WWW-servern kan också behandla värden som användaren fyllt i formulär, exempelvis för att använda dem som sökvillkor i en sökning eller för att lägga in dem i databasen.

Det finns flera tänkbara tillvägagångssätt att generera dokument i en WWW-server. Det idag vanligaste är att skriva program, s k gatewayprogram, som serverprogrammet på WWW-servern startar när en begäran från en WWW-läsare kommer. Istället för att skicka tillbaka ett dokument från en fil så genererar gatewayprogrammet dokumentet som skickas. Ett gatewayprogram har normalt en särskild uppgift. Ett program kan genererar en lista och ett annat lägger in värdena från ett formulär i databasen. Det finns även andra varianter än att ha fristående gatewayprogram.

Gatewayprogrammen kan styras med hjälp av parametrar. En användare kan t ex skicka med värden från ett formulär eller ett sökfält. Det går också att lägga in fasta parametrar i dokumentadressen.

Många interaktiva tillämpningar kräver att information om vad användaren gjort lagras undan. Ett tillstånd måste sparas medan användaren navigerar fram genom de dokument som utgör tillämpningen. Tyvärr så finns det inte något direkt stöd för detta i WWW-arkitekturen. Det går att lösa detta problem på olika sätt i en tillämpning. Två olika huvudprinciper finns där den ena innebär att tillståndet lagras hos WWW-läsaren, t ex i gömda fält, och det andra där tillståndet sparas på servern av gateway-programmen.

Till vissa databaser finns färdiga gatewayprogram som förenklar kopplingarna mellan WWW och databaserna. De döljer mycket av hanteringen av databaser och parametrar i WWW. Istället konfigureras de för tillämpningen på något annat sätt. Gatewayprogrammen har olika funktionalitet. De enklaste klarar enbart av att generera listor från fasta frågor som lagts in i konfigurationen medan de mest avancerade även klarar av att lägga in information i en databas. Men inte heller de mest avancerade räcker till för alla slags tillämpningar utan kräver programmering av egna gatewayprogram.

# 2 Introduktion

Användningen av World Wide Web (WWW) för informationshantering på Internet växer hela tiden. Fler och fler företag och institutioner väljer att göra sig sedda på Internet med WWW. Nya WWW-servrar tillkommer hela tiden.

Idag är mängden information på varje server i regel liten. Det handlar om få dokument vars innehåll inte uppdateras regelbundet. Oftast rör det sig om dokument med huvudsakligen text och kanske någon eller några bilder och ibland ljudsnuttar eller videoklipp. Dokumenten lagras i de flesta fallen som vanliga filer på servern. I och med att mängden är så liten så behövs ingen mer genomarbetad struktur för att underhålla systemet. Ändringar kan göras för hand utan några större problem.

Dagens hantering har däremot svårt att skala upp när större mängder information ska hanteras. Den passar inte heller för interaktiva tjänster, något som kommer att dyka upp mer och mer. Dessa typer av tjänster kräver mer arbete än att bara lägga upp dokument på en server. Problem som kan uppstå vid en uppskalning är bl a:

- Det blir svårt att hålla ihop alla länkar mellan dokumenten. Gamla länkar till ett obefintlig dokument kan ligga kvar eller ett dokument kan "tappas bort" om alla länkar till det försvinner.
- Inkonsistens mellan samma information lagrad på olika ställen. Ett exempel kan vara en intern telefonlista. Den ska dels distribueras på papper till de anställda, finnas tillgänglig via terminal i växeln och läggas upp i WWW. När listan ska uppdateras måste det göras på alla ställen för att informationen ska vara aktuell på alla ställen.
- Administrationen blir svårhanterlig. För att klara stora mängder information är en genomarbetad struktur av filerna nödvändig liksom ett system för namngivningen. Det är särskilt viktigt om det är många personer som arbetar med att lägga in dokument. Detta berör inte bara namn på dokument utan även länkar mellan dessa. När ett system väl tagits i bruk är det svårt att byta det i och med den stora dokument- och länkmängden.
- Alla de som jobbar med dokumenten måste kunna generera ett HTML-dokument. Antingen ska det göras med hjälp av något filter eller direkt i HTML-koden.
- Nya versioner av HTML kommer att innehålla nya finesser som t ex *style sheets* som väntas komma i version 3 av HTML. För att dra nytta av dessa nyheter så måste då gamla dokument förändras.
- Layouten skiljer sig mellan olika dokument. En enhetlig layout kan vara viktig för ett företags profil, men med många dokument och många personer som underhåller dem är risken stor att de får olika utseende.
- Vid interaktiv användning med formulär måste inmatade värden tas om hand och behandlas. Detta kan inte göras utan ett program på servern.

Dessa problem kan lösas genom att WWW kombineras med olika typer av stödsystem. I stället för att WWW-servern hämtar ett lagrat HTML-dokument från en fil så skapar serverprogramvaran ett dokument utgående från olika informationsbitar och skickar tillbaka det till användaren. Denna s k generering av dokument kan antingen ske direkt i serverprogramvaran eller via ett externt program. I botten kan t ex en databas finnas med informationen som ska presenteras i HTML-dokumentet. Ett genererat dokument kan innehålla allt ett vanligt HTML-dokument kan, inklusive länkar, olika slags rubriker, formulär, bilder etc.

## 2.1 Lagring och presentation av stora informationsmängder med hjälp av WWW och databaser

På samma sätt som databaser utnyttjas för att hantera stora informationsmängder i andra miljöer så kan de användas för att lösa samma problem i WWW. WWW kan i sin tur användas för att publicera de stora mängder data som finns i en databas. Exempel på fördelar med att använda WWW och databaser för att hantera stora informationsmängder är:

- Genererade dokument innehåller inga felaktiga länkar. Kontrollen att dessa är giltiga kan göras redan när dokumentet genereras.

- Samma information i databasen kan användas för parallellpublicering. T ex behöver en telefonlista bara uppdateras i databasen så kan informationen där användas för att generera både pappers-exemplar och ett WWW-dokument.
- WWW kan användas som gränssnitt mot databaser. Det går att bygga hela applikationer där WWW fungerar som det grafiska användargränssnittet. Applikationerna blir portabla och kan användas på alla plattformar där det finns WWW-läsare.
- De som producerar informationen i databasen behöver inte kunna HTML för att presentera den på WWW. Det räcker med att den som gör programmet som hämtar informationen från databasen kan HTML. På detta sätt går det också att få en enhetlig layout genom att bara ett fåtal program behöver uppdateras för att göra ändringar i utseendet.
- I databashanterare finns oftast goda möjligheter att göra sökningar. Detta kan användas så att en användare enkelt ska kunna navigera fram till rätt information.

Att använda en databas kommer dock inte att lösa alla problem i en handvändning. Fortfarande kommer det krävas en genomtänkt lösning med en bra databasstruktur.

## 2.2 Interaktiva tjänster baserade på WWW och databaser

Interaktiva tjänster är ett annat område där WWW och databaser utgör en bra kombination. Interaktiva tjänster är tjänster där användaren kan påverka vad som händer mer än att bara bläddra sig fram mellan olika dokument. Systemet anpassar sig efter vem användaren är och vad denna gör. Responsen från servern beror t ex på vad en användare skrivit in för värden i ett formulär.

Det vanligaste sättet att använda interaktiva tjänster i WWW är via formulär. Ett formulär är ett HTML-dokument som förutom text och bilder består av ett antal fält för inmatning av data. Det finns några olika typer av fält: en- och flerradig textfält, radioknappar och en- och flervalslistor. Ett exempel på hur ett formulär kan se ut finns i bild 1.

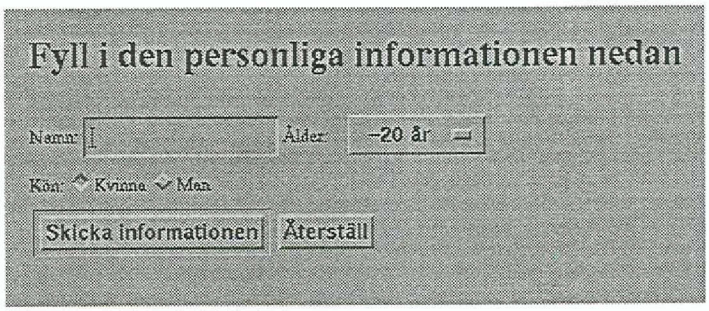


*Bild 1. En WWW-läsare med ett formulär uppe. Namnet skrivs in i ett textfält, åldern väljs från en listruta och personens kön väljs via de två radioknapparna.*

Några exempel på interaktiva tjänster är:

- WWW som försäljningskanal. Via formulär väljer användaren varor som ska köpas. Varorna summeras på en lista som användaren kan kontrollera innan köpet registreras. En order lagras sedan i databasen för vidare hantering.
- Support över Internet via WWW. En användare fyller i formulär med någon fråga. Detta kan antingen vara ett meddelande i textformat som skickas till en supportavdelning eller så kan det vara t ex ett expertsystem som ger lösningsförslag utifrån användarens val.
- Utbildning med inlärningskontroll. Information som användaren ska lära sig presenteras och sedan ställs frågor för att kontrollera om användaren har förstått. Vad användaren klarar avgör vad som sedan kommer att läras ut. Kanske måste något presenteras om igen eller så kan användaren gå vidare i studierna.
- Arbetsgrupper med stöd för distribuerad diskussion. Användarna kan dela på filer, adresser och föra diskussioner i forum. All information lagras i databasen och WWW används för att användare ska

kunna nå samma information var de än befinner sig så länge de har tillgång till nätverksförbindelse. Bild 2 visar hur en diskussionsgrupp kan se ut.

- Informationssökning i databaser. Med WWW kan användaren via formulär söka och navigera runt i databaser.

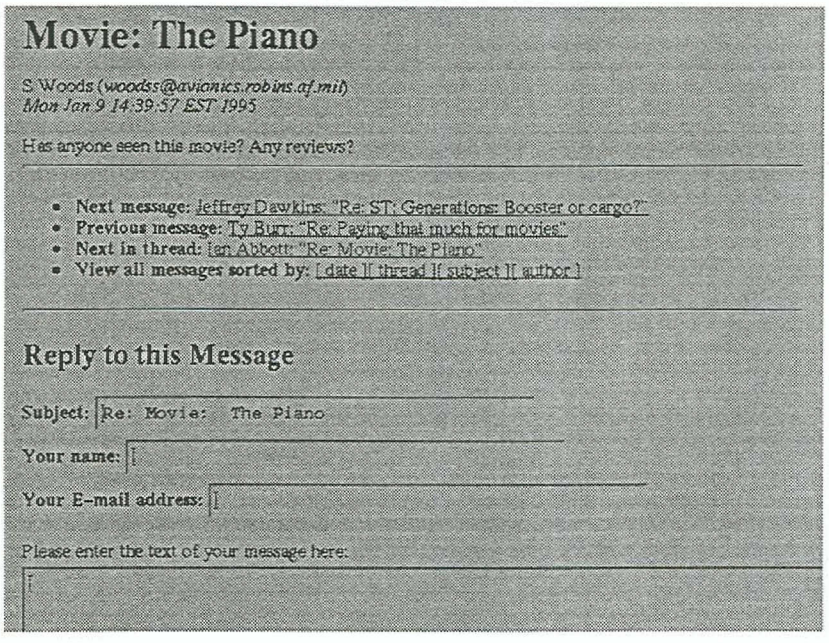


*Bild 2.* *Ett inlägg i en diskussionsgrupp. Användaren kan välja att fortsätta läsa svaren i inlägget eller att läsa inlägg om något annat ämne. Det är också möjligt att svara på inlägget.*

I vissa tillämpningar kan det vara nödvändigt att veta vem användaren är. Förutom att kontrollera behörighet kan en användaridentitet användas för andra ändamål. Ett exempel är elektronisk post där en användare har en egen brevlåda med inkomna brev. I WWW finns inbyggt stöd för en användare att "logga in" med användaridentitet och lösenord.

En interaktiv tjänst kan ofta behöva hålla reda på i vilket tillstånd användaren befinner sig. En användare kanske ska navigera sig fram bland flera olika dokument innan någon uppgift är avslutad och på något sätt måste informationen från t ex formulär finnas kvar mellan de olika dokumenten. Problemet är att en WWW-server inte sparar någon information om vad den fört över tidigare till läsaren, den är tillståndslös. Dokumenten skickas från servern till läsaren oberoende av varandra. Detta problem går dock att komma förbi och några olika sätt att göra det beskrivs senare i rapporten.

# 3 Tekniken – så fungerar det

En koppling mellan en WWW-server och en databas kan göras på flera sätt. Ett exempel på hur en arkitektur för detta kan se ut finns i bild 3. När serverprogrammet får en begäran från en användare så startar det ett program istället för att som vanligt skicka tillbaka en fil. Detta program, gatewayprogrammet, genererar i sin tur dokumentet som skickas tillbaka till användaren.

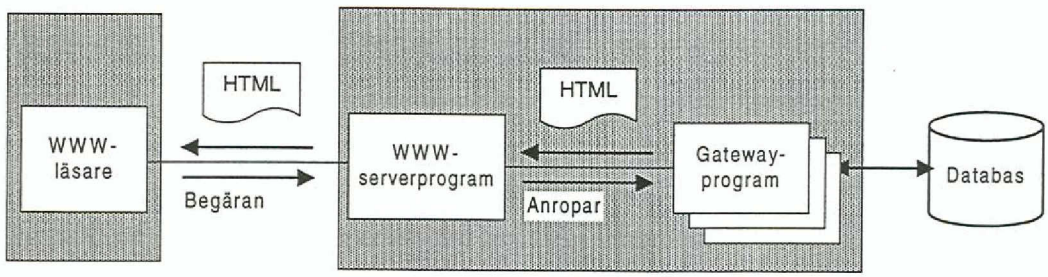


*Bild 3.* *Arkitektur med WWW-serverprogram, gatewayprogram och en databas. När serverprogrammet får en begäran från en WWW-läsare så anropar det ett gatewayprogram. Detta genererar ett HTML-dokument med information från databasen. Dokumentet skickas sedan tillbaka till läsaren via serverprogrammet.*

Varje gatewayprogram har normalt en specifik uppgift. T ex kan ett program generera en telefonlista och ett annat ett sökformulär. Utifrån en begärd dokumentadress (URL) väljer serverprogrammet vilket gatewayprogram som ska startas. Gatewayprogrammet kan även styras genom att parametrar skickas med, t ex värden från ett formulär. Mer om detta senare.

Bild 3 visar en typ av arkitektur, men andra är också möjliga. En variant är att mer funktionalitet läggs i serverprogrammet som på så sätt blir skräddarsytt för en viss tillämpning. Bland annat har Netscape några olika specialiserade serverprogram för olika tillämpningar och BASISweb-servern från Information Dimensions innehåller en dokumentdatabas. Mer information om dessa och andra liknande produkter finns i avsnitt 5.

Ytterligare ett alternativ till arktitektur är att bara ett enda gatewayprogram används. Detta måste då klara alla de funktioner som behövs för tillämpningen. Vilken arkitektur som man ska välja beror på de krav och resurser man har. Ett skräddarsytt serverprogram är troligen den lösning som är effektivast när det gäller att hantera många samtidiga användare. Dessa program åstadkommer man knappast själv utan de förutsätter att någon leverantör har en färdig lösning att erbjuda. Fristående gatewayprogram är enklare att utveckla då de kan testas var för sig. Det finns idag flera olika färdiga gatewayprogram som kan konfigureras på olika sätt. Ofta är de enklare att använda än att själv skriva gatewayprogram, men de brukar också ha olika slags begränsningar som gör att de inte fungerar för alla slags tillämpningar.

## 3.1 Gatewayprogram

Gatewayprogram har normalt till uppgift att generera HTML-dokument när de exekveras. För att göra programmen mer generella kan värden för att styra dem skickas med. Dessa värden kan t ex komma från formulär som användaren fyllt i eller så kan de hårdkodas i URL:en. Programmen får även annan information från WWW-läsaren och serverprogrammet som t ex vilka dokumenttyper läsaren hanterar och eventuellt användarens identitet.

Gatewayprogram är dock inte begränsade till att enbart skapa HTML-dokument. Det finns även program som genererar andra typer av dokument som t ex en bild med ett diagram. Ett sådant program kan användas för att lägga in aktuella kurvor i ett WWW-dokument, då programmet alltid använder de senaste värdena.

### Parameteröverföring

Genom att skicka med parametrar till ett gatewayprogram kan man styra det. Det vanligaste sättet är att en användare fyller i data i ett formulär. Data från formuläret skickas med i en begäran till WWW-servern där serverprogrammet vidarbefordrar dem till gatewayprogrammet. Ett formulär består av ett antal namngivna fält. Värdena från formuläret överförs kodade i en sträng med alla namn/värde-par. Som exempel visas HTML-koden för formuläret bild 1 nedan:

```
<form action="/cgi-bin/dosomething" method="GET">
Namn:  <input name="namn">
Ålder: <select name="ålder">
<option value=0>  -20 år
<option value=1>21-40 år
<option value=2>41-60 år
<option value=3>61-80 år
<option value=4>81-   år
</select>
Kön: <input type="radio" name="kön" value="kvinna" checked> Kvinna
     <input type="radio" name="kön" value="man"> Man
<input type="submit" value="OK"> <input type="redo" value="Clear">
</form>
```

Det här formuläret innehåller tre olika fält; *namn*, *ålder* och *kön*. *namn* är ett textfält, *ålder* är ett värde från en listruta och *kön* är ett värde från någon av de två radioknappar. Med de värden formuläret har i bild 1 skickas de tre värde/namn-paren nedan till gatewayprogrammet *dosomething* när användaren klickar på OK-knappen:

```
namn=Peter
ålder=1
kön=man
```

I den första raden i HTML-koden ovan står det bl a *method="GET"*. GET och POST är de två metoder för parameteröverföring som används från ett formulär. GET används för att skicka värdena från formuläret i URL:en, direkt efter den del som anger programmet. I exemplet ovan kan adressen se ut på följande sätt:

```
http://datornamn/cgi-bin/dosomething?namn=Peter&ålder=1&kön=man
```

Nackdelen med GET-metoden är att det inte går att sända hur mycket information som helst. Innehållet i ett textfält kan göra att adressen blir allt för lång. Alla läsar- och serverprogram klarar nämligen inte av hur långa adresser som helst. Därför har metoden POST tillkommit. Med denna metod skickas parametrarna över till servern separat från adressen.

Det finns även några ytterligare metoder som kan användas för att skicka med värden till ett gatewayprogram:

- Ett ISINDEX-fält som i stort sett är ett formulär med enbart ett fält och utan en sändknapp. När användaren skrivit in ett värde och tryckt på ENTER-tangenten så skickas värdet över till servern. Värdet skickas direkt i adressen på samma sätt som med metoden GET. Något fältnamn skickas dock inte med.
- Klickbara bilder (Clickable images). När användaren klickar på en sådan bild skickas koordinaterna för den positionen musmarkören har till servern. Klickbara bilder används oftast för att göra snyggare länkar. Ett gatewayprogram tar hand om koordinaterna och avgör vilket dokument som ska visas beroende på i vilken region av bilden användaren klickat. Koordinaterna överförs på samma sätt som i GET-metoden.
- Hårdkodade i URL:en. På så sätt kan gatewayprogram styras utan att användarna behöver mata in några värden. Parametrarna kodas på samma sätt som i GET-metoden.

Förutom parametrarna skickar serverprogrammet och läsaren ytterligare information till gatewayprogrammet. Detta är några av de saker som gatewayprogrammet kan ha nytta av:

- Om användaren har identifiera sig för servern så får gatewayprogrammet denna information från serverprogrammet.

- De dokumenttyper som WWW-läsaren kan hantera, antingen internt eller via något hjälpprogram. Gatewayprogrammet kan använda denna information för att bestämma vilka format t ex bilder ska skickas med. Om läsaren klarar JPEG-bilder så kan dessa användas istället för bilder i GIF-formatet om JPEG-bilderna är mindre och därmed överförs snabbare.

## 3.2 Tillstånd

När en WWW-läsare ska hämta ett dokument från en WWW-server etableras en förbindelse mellan dem som bryts när dokumentet är överfört. Vill användaren ha ytterligare dokument så måste förbindelsen etableras igen. När en användare avbryter en session, t ex genom att avsluta läsar-programmet, skickas ingen information om detta till servern. Om servern har sparat ett tillstånd för en användare så kommer den inte få veta att det inte längre är aktuellt. Därför är serverprogrammen i WWW normalt tillståndslösa.

För många interaktiva tillämpningar krävs det dock att ett tillstånd bevaras. I t ex fallet med försäljning via WWW så är detta nästan en nödvändighet. Jämför med en affär där kunderna lägger varorna i en kundvagn där de ligger kvar tills kunden når kassan. På samma sätt måste information om vad en kund valt från olika WWW-sidor sparas undan så att allt finns med vid en slutlig beställning och betalning. För att göra detta finns några olika tekniker som löser problemet trots att serverprogrammet är tillståndslöst.

Ett sätt att komma runt problemet är att låta WWW-läsaren hantera tillståndet genom att lagra det i gömda fält i ett formulär. På så sätt behöver inte serverprogrammet hålla reda på det. Gömda fält är fält i formulär som inte visas av läsaren men som överförs till servern tillsammans med de värden som användaren skrivit in. När ett gatewayprogram får värdena från ett formulär så överförs de till nästa formulär genom att de läggs i gömda fält i det nya formulär som programmet skapar och skickar tillbaka. Användaren fyller på nytt i detta formulär och proceduren upprepas tills en slutlig behandling av tillståndet görs, t ex så produceras en slutlig order och kunden debiteras. För att detta ska fungera så krävs att formulär används genom hela tillämpningen för annars kan inte tillståndet lagras.

En liknande lösning är att lägga in tillståndet som en parameter i länkarna. Denna lösningen kan inte hålla lika mycket information (se förklaringen av GET-metoden tidigare) men den kan användas i vanliga HTML-dokument när formulär inte används. En fördel med båda dessa metoder är att om användaren avbryter mitt i en transaktion så kommer inte servern lagra undan tillståndet utan det kommer helt att kastas bort.

En helt annan lösningsmetod är att spara tillståndet på servern. Detta görs inte av serverprogrammet utan av gatewayprogrammen. Tillståndet måste på något sätt kunna identifieras. Detta kan göras antingen via användarens identitet eller via någon tillfällig transaktionsidentitet. Om en tillfällig identifierare används så måste denna skickas med hela tiden. Det största problemet med att servern lagrar tillståndet är att en användare kan avsluta tillämpningen utan att meddela servern detta. Tillståndet ligger då och skräpar på servern. Ett sätt att undvika detta är att radera tillstånd som inte använts under en viss tid. Denna tid får inte vara för kort, för då kan tillstånd som användaren fortfarande anser vara aktiva rensas bort. Servern kan även lagra mer beständiga tillstånd som t ex en användares inställningar.

Vilken metod man ska välja för att hantera tillstånd beror på tillämpningen. Om den enbart består av formulär är lösningen med gömda fält den lämpligaste. Då behöver inte servern hantera tillståndet och på så sätt undviker man de problem som detta innebär. Om detta inte går kan den enda lösningen vara att hantera tillstånden på servern.

## 3.3 Programmering vs färdiga gatewayprogram

Gatewayprogram kan man antingen skriva själv eller använda något av de färdiga gatewayprogram som redan finns till olika databashanterare. Dessa färdiga program måste på något sätt konfigureras för den aktuella tillämpningen. Den tänkta tillämpningen får styra vilket alternativ som ska väljas. Färdiga program är att föredra men de är inte alltid tillräckligt kraftfulla. Då kan den enda utvägen vara att själv skriva programmen.

### Färdiga gatewayprogram för databaser

Fördelen med att använda färdiga gatewayprogram är att mycket av lågnivåhanteringen av databasen försvinner. Därför är det ofta enklare att skriva tillämpningar i WWW med färdiga program. I vissa fall går det att göra helt utan programmeringskunskaper. Möjligheterna och begränsningarna som finns

liksom tillvägagångsättet för att bygga tillämpningarna skiljer sig åt beroende på vilket gatewayprogram som används.

Normalt används ett program som kan konfigureras för att användas på flera olika sätt. Denna konfigurering kan t ex göras genom att använda olika konfigurationsfiler. För att avgöra vilken konfigurationsfil som ska användas så kan namnet på denna läggas in i URL:en. Denna arkitektur skiljer sig lite från den tidigare i bild 3. Bild 4 visar hur den skulle kunna se ut.

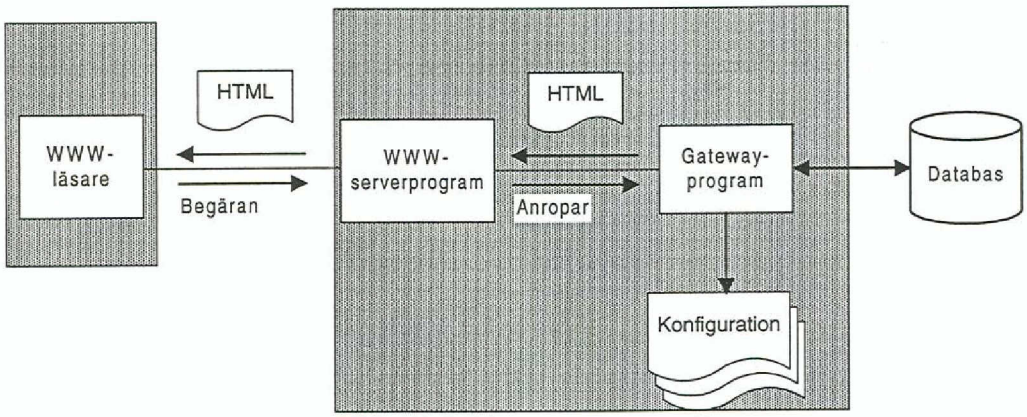


*Bild 4.* *När serverprogrammet får en begäran från WWW-läsaren så anropar det gatewayprogrammet. Detta läser in en konfigurationsfil som innehåller information, t ex vad den ska skicka för fråga till databasen, vad den ska göra med eventuella parametrar och hur det resulterande dokumentet ska se ut. Detta dokument skickas tillbaka till serverprogrammet som vidarbefordrar det till läsaren.*

De enklaste gatewayprogrammen tillåter enbart att fördefinierade databasfrågor används. Det går alltså inte att t ex specificera sökvillkor via formulär. Dessa program lämpar sig för standardrapporter som t ex att skapa någon särskild lista. Genom att uppdatera databasen kommer hela tiden rapporten att bygga på den senaste informationen utan att HTML-dokumentet behöver redigeras.

För mer avancerade tjänster behövs kraftfullare program som även kan hantera värden från formulär. Sökningar i databasen kan då begränsas av dessa värden. Det kan också finnas mer kontroll över genereringen av dokumenten. De mest avancerade gatewayprogrammen tillåter att man även kan uppdatera innehållet i databasen.

I avsnitt 5 finns exempel på färdiga gatewayprogram till några olika databashanterare.

### Programmering

De färdiga gatewayprogrammen räcker inte alltid till. Kanske stöds inte den databas man vill använda eller så är kraven på felhantering större än vad som erbjuds. Egenutvecklade gatewayprogram erbjuder mycket större kontroll över såväl databas som genereringen av dokument. Programmen skrivs i något vanligt programmeringsspråk. Om en databas ska användas så krävs naturligtvis att det ska finnas möjlighet att komma åt databasen från språket. Under Unix kan t ex något av språken C eller Perl användas.

Vad gatewayprogrammet verkligen ska göra beror naturligtvis på tillämpningen. I allmänhet ska det i alla fall ta hand om eventuella parametrar, koppla upp sig mot databasen och sedan utföra uppgiften. Som resultat genereras normalt ett HTML-dokument.

### Gatewayprogram på Unix-datorer

Under Unix-datorer finns en standard för hur gatewayprogram ska kommunicera med server-programmet. Denna standard går under namnet Common Gateway Interface (CGI). Detta innebär att samma gatewayprogram kan användas med alla serverprogram med stöd för CGI. De flesta

serverprogram gör det, bland annat NCSA httpd, Cern httpd, Plexus och Netsite. Standarden definierar hur parametrar ska vidarbefordas till gatewayprogrammet och hur formatet för det genererade dokumentet ska se ut.

För att överföra information till gatewayprogrammen används i huvudsak environmentvariabler. Dessa används i Unix för att ge program information om den omgivning de körs i, t ex sökvägen till körbara program, användaridentitet eller vilken texteditor som ska användas. CGI-standarden definierar ett antal environmentvariabler som gatewayprogrammen kan använda. Tabell 1 innehåller de viktigaste av dessa. När det gäller överföringen av parametrar så hanteras de två metoderna GET och POST olika i CGI-program. När GET-metoden används skickas parametrarna i environmentvariabeln QUERY_STRING. POST-metoden använder standardinmatningen (stdin) för överföringen. I båda fallen ligger alla parametrar tillsammans i en kodad sträng. För några vanliga programmeringsspråk finns färdiga funktioner för att avkoda denna sträng till ett mer lätthanterligt format. Mer information om CGI finns på URL:en http://hoohoo.ncsa.uiuc.edu/cgi/.

*Tabell 1. Några environmentvariabler som kan användas av CGI-program och vad de innehåller*

| Variabelnamn | Innehåll och användning |
|---|---|
| REMOTE _USER | Ett användarnamn om man begärt identifiering. |
| REQUEST_METHOD | Vilken överföringsmetod som används för parametrarna, antingen GET eller POST. |
| QUERY_LENGTH | Om POST används så antal tecken som ska läsas från 'stdin'. |
| QUERY_STRING | Innehåller söksträngen när GET används. |
| PATH_INFO | Resten av URL-adressen efter CGI-programmet. |
| ACCEPT_TYPE | De dokumenttyper som WWW-läsaren kan visa. Är kodade enligt *MIME*-standarden. Kan användas för att bestämma vilken typ av data som ska skickas över. |

Som alternativ till CGI finns Netscape Server API (NSAPI) i den senaste versionen av Netsite. Det är ett programmeringsgränssnitt för hur gatewayfunktioner kan länkas in i Netsite och på så sätt utöka funktionaliteten hos serverprogrammet. Fördelen med detta jämfört med CGI är att det går snabbare att exekvera då ett nytt program inte behöver startas och initieras. Det går även att låta egna rutiner kontrollera användarnamn och lösenord i stället för att använda filer som är vanligt idag. På så sätt kan även namn och lösenord lagras i databasen. Nackdelen med NSAPI är att det är Netscapes egen lösning och att inga andra använder den. Mer information om NSAPI finns på adressen http://home.netscape.com/newsref/std/server_api.html

### Gatewayprogram under Windows

Till Windows och Windows NT finns ett antal serverprogramvaror. Gatewayprogrammen hanteras lite olika beroende på vilket serverprogram som används. Några exempel på olika serverprogram finns nedan:

- Windows httpd kan exekvera gatewayprogram på två olika sätt. Det första liknar det sätt som CGI-programmen under Unix exekverar. Det andra, Windows CGI, sparar parametrarna på en fil som gatewayprogrammet sedan kan läsa. Gatewayprogrammen kan skrivas i t ex Visual Basic. Information om Windows httpd finns på adressen http://www.city.net/win-httpd. Där finns det även information om hur de två olika metoderna för att skriva gatewayprogram fungerar.
- Purveyor under Windows NT hanterar gatewayprogram på samma sätt som Unix CGI-program. Mer information om Purveyor finns på adressen http://www.process.com/prodinfo/purvdata.htm.
- Netscape har en version av Netsite för Windows NT. Förutom CGI-program enligt Unix-standarden klarar den dessutom av NSAPI på samma sätt som i Unix-versionen av Netsite. Mer information finns på Netscapes hemsida på adressen http://home.netscape.com.

### Gatewayprogram på Macintosh

Från serverprogrammet MacHTTP på Macintosh skickas AppleEvents till gatewayprogrammen för att överföra parametrar till dem. Funktionaliteten blir densamma som för CGI-programmen men tillvägagångssättet blir annorlunda. Gatewayprogrammen kan skrivas i AppleScript, C eller något annat programmeringsspråk som kan hantera AppleEvents. Mer information om MacHTTP finns på http://www.biap.com och exempelprogram för MacHTTP finns på http://edb518ea.edb.utexas.edu/scripts/cgix/cgix.html.

## 3.4 Säkerhet

Det finns många säkerhetsaspekter att ta hänsyn till när man bygger en tjänst på Internet. Denna rapport är dock inte inriktad på säkerhetsfrågor. Några saker ska bara nämnas för att belysa vad som kan vara viktigt att tänka på och var mer information finns.

Det gäller att skydda databasen mot obehörigt intrång. Gatewayprogrammen och användarna ska bara ha rättigheter att läsa de delar av databasen de behöver. Hur dessa saker ska göras beror på den databashanterare som används, så titta i den dokumentation som följer med. Detta är saker som man alltid måste ta hänsyn till när en databas ska användas.

WWW är långt ifrån säkert. Överföring av dokument sker i klartext liksom användarnamn och lösenord vid inloggning. Denna information skulle kunna avlyssnas av andra på nätverket. Dessutom kan man inte vara säker på att en användare är den han eller hon uppger sig vara. Mer information om dessa ämnen finns på adressen http://www-ns.rutgers.edu/www-security/index.html.

# 4 Exempel: Gemensam URL-hantering

En enkel interaktiv tillämpning visas här som exempel där WWW kombineras med en databas. Exemplet är skrivet för Unix-datorer och Oracle databashanterare, men samma tillämpning går att göra på andra plattformar på liknande sätt. Tillämpningen visas i två olika versioner för att visa på likheter och skillnader mellan de olika tillvägagångssätten. Den första versionen använder ett färdigt gatewayprogram från Oracle som heter WOW, den andra är skriven i programmeringsspråket C.

## 4.1 Tillämpningen

Tillämpningen handlar om en gemensam hantering av WWW-adresser, s k URL:er. Den är tänkt att användas för att flera personer ska kunna dela på URL:er som kan vara av gemensamt intresse. För att strukturera adresser så klassificeras de i olika kategorier. Tillämpningen består av två huvudfunktioner:

- En användare ska kunna lägga in nya URL:er i databasen. Förutom adressen ska även en kort och en lång beskrivning och en kategori läggas in. I en WWW-läsare kan detta se ut som i bild 5.
- En användare ska kunna lista alla URL:er i en viss kategori. Först ska en kategori väljas från ett formulär. Från detta ska sedan listan på alla URL:er i kategorin visas. Det är bara den korta beskrivningen som visas för varje URL. Denna beskrivning fungerar sedan som en länk för att hämta dokumentet som adressen pekar på. Dessutom ska en länk finnas till den längre beskrivningen av adressen. Hur detta kan se ut visas i bild 6.

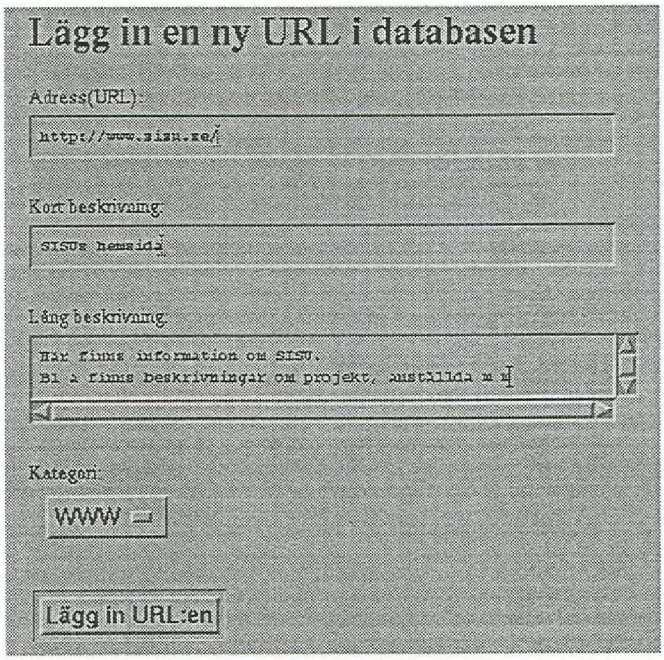


*Bild 5. Ett formulär där användaren kan skriva in en ny URL.*

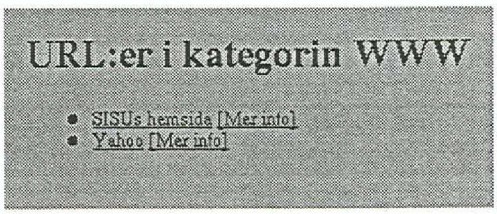


*Bild 6. Listan på alla URL:er för kategorin WWW. Genom att klicka på beskrivningen hämtas dokumentet som anges. Genom att klicka på texten* Mer info *visas den långa beskrivningen.*

### Databasstruktur

Databasen innehåller två tabeller för denna tillämpning. Den första tabellen innehåller information om kategorier och den andra information om länkar.

```
url (id, category, address, short_desc, long_desc)
categories (id, description)
```

*Id*-fältet i de båda tabellerna är primärnycklar. Fältet *category* i tabellen *url* är en främmande nyckel som pekar på en rad i *category*-tabellen.

### Programstruktur

Utseendet och strukturen på denna tillämpning kommer vara den samma i båda versionerna. Från en huvudsida kan någon av de båda funktionerna väljas.

#### Funktionen ny URL

Användaren ska få upp ett formulär med fält för adress, den korta och den långa beskrivningen och kategori, se bild 5. Adressen och den korta beskrivningen är enraders textfält, den långa beskrivningen ett flerraders textfält och kategorin väljs från en listruta. Kategorierna i denna lista hämtas från databasen. I och med detta så måste formuläret genereras av ett program. Detta program behöver inga parametrar. När användaren fyllt i alla värden och tryckt på knappen *Lägg in URL:en* så kommer URL:en att läggas in i databasen. Det görs av ett annat program än det första. Detta program tar värdena från formuläret som parametrar. Totalt behövs alltså två program för att lägga in en ny URL i databasen, ett för att skapa formuläret och ett för att lagra URL:en.

#### Funktionen lista URL:er i kategori

På samma sätt som i funktionen ny URL så behövs ett program som genererar ett formulär med kategorilistan. Inte heller detta program tar några parametrar. Sedan behövs ett program som skapar ett HTML-dokument med alla URL:er i den valda kategorin. Programmet får *id*-fältet för kategorin som parameter. Från listan kan användaren välja att visa detaljer om en URL. Det blir det sista programmet i denna tillämpning. Detta program tar URL:ens *id*-fält som parameter för att kunna visa rätt information.

## 4.2 Lösning med WOW

Gatewayprogrammet WOW kan hämtas hem gratis från Oracles WWW-server (se avsnitt 5). Funktionaliteten i WOW bestäms av databasprocedurer lagrade i databasen. Det enda WOW gör är att anropa dessa procedurer och sedan slussa tillbaka resultatet från dem till serverprogrammet. Databasprocedurerna skrivs i PL/SQL som är Oracles eget programmeringsspråk speciellt anpassat för att användas i databastillämpningar. Dessa procedurer kompileras in i databashanteraren innan de kan användas. Med WOW följer ett PL/SQL-procedurbibliotek för att skapa HTML-kod. Alla procedurer i detta bibliotek börjar med *htp* i exemplet nedan.

WOW använder en del av dokumentadressen för att avgöra namnet på databasproceduren som ska anropas. Denna del får WOW från environmentvariabeln PATH_INFO. I adressen

```
http://www.sisu.se/cgi-bin/wow/new_URL_form
```

är *new_URL_form* den databasprocedur som kommer att anropas. Eventuella parametrar vidarbefordrar WOW till proceduren när det anropas. Det är viktigt att namnen på alla parametrar stämmer överens med namnen på de argumentnamn man angett när databasproceduren skrevs, för annars kommer WOW inte att utföra anropet. Även parametrar direkt kodade i adressen måste vara namngivna för att det ska fungera.

Funktionen ny URL består av två databasprocedurer, *new_URL_form* och *new_URL*. Proceduren *new_URL_form* skapar ett HTML-dokument som innehåller formuläret för att lägga till en ny URL, se bild 3. En extra procedur, *category_list*, används också för att skapa en listruta med kategorinamnen.

```
create or replace procedure new_URL_form is
begin
  htmlhead('Ny URL');
  htp.header(1, 'Lägg in en ny URL i databasen');
  htp.formOpen('/cgi-bin/wow/new_URL');     -- Det program värden ska skickas till
  htp.p('Adress(URL):'); htp.nl; htp.formField('address', 60); htp.para;
  htp.p('Kort beskrivning:'); htp.nl; htp.formField('short_desc', 60); htp.para;
  htp.p('Lång beskrivning:'); htp.nl; htp.para;
  htp.p('<textarea name=long_desc rows=2 cols=60></textarea>'); htp.para;
  htp.p('Kategori:'); htp.nl; category_list; htp.para;
  htp.formDo('Lägg in URL:en');
  htp.formClose;
  htmlend;
end;


create or replace procedure category_list is
cursor cs is select * from categories;
begin
  htp.formSelectOpen('category');
  for c in cs loop
    htp.p('<option value='||c.id||'>'||c.name);
  end loop;
  htp.formSelectClose;
end;
```

När användaren klickar på *Lägg in URL:en* på formuläret så anropas databasproceduren *new_URL*. Denna lägger in URL:en i databasen och visar också en bekräftelse på detta för användaren.

```
create or replace procedure new_URL(address in varchar2, short_desc in varchar2,
                                    long_desc in varchar2, category in number) is
begin
  insert into url values (url_seq.nextval, category, address,
                          short_desc, long_desc);
  htmlhead('URL inlagd');
  htp.header(1, 'URL inlagd');
  htp.url('/', 'Tillbaka till framsidan');
  htmlend;
end;
```

För att komma åt URL:er lagrade i databasen används tre procedurer, *list_URLs_form*, *list_URLs* och *details*. Proceduren *list_URLs_form* skapar ett dokument med ett formulär som bara består av en listruta med namnen på de olika katagorier som finns.

```
create or replace procedure list_URLs_form is
begin
  htmlhead('Lista URL:er');
  htp.formOpen('/cgi-bin/wow/list_URLs');
  category_list;
  htp.formDo('Lista');
  htp.formClose;
  htmlend;
end;
```

Proceduren *list_URLs* visar alla URL:er som finns för kategorin som användaren valt (bild 4). Användaren kan från listan antingen hämta något av dokumenten som finns listade genom att klicka på beskrivningen av URL:en eller titta på mer information en URL. HTML-koden för en rad i denna lista kan se ut på följande sätt:

```
<li><a href="http://www.yahoo.com/">Yahoo</a>
<a href="/cgi-bin/wow/details?id=5">[Mer info]</a>
```

Den första raden här ovan innehåller en länk där den korta beskrivningen från databasen är det aktiva området för denna URL och adressen från databasen det dokument som hämtas när användaren klickar på länken. Andra raden beskriver en länk till mer information om URL:en i databasen. I detta fall skulle databasproceduren *details* anropas med parametern *id=5* där *id* är den identifierare i databasen som hör ihop med denna URL.

Koden till proceduren *list_URLs*:

```
create or replace procedure list_URLs(category in number) is
cursor urls is select * from url U where U.category=category;
cat varchar2(40);
begin
  select name into cat from categories where id=category;
  htmlhead('Lista på URL:er');
  htp.header(1, 'URL:er i kategorin '||cat);
  htp.ulistOpen;
  for l in urls loop
     htp.item; htp.url(l.address, l.short_desc);
     htp.url('/cgi-bin/wow/details?id='||l.id, '[Mer info]');
  end loop;
  htp.ulistClose;
  htp.url('/', 'Tillbaka till framsida');
  htmlend;
end;
```

Den längre beskrivningen visas av proceduren *details* som tar identifieraren för URL:en som parameter.

```
create or replace procedure details(id in number) is
begin
  select U.long_desc into d from url U where U.id=id;
  htmlhead('Detaljer för URL');
  htp.p(d); htp.para;
  htp.url('/', 'Tillbaka till framsida');
  htmlend;
end;
```

## 4.3 Lösning i C

Lösningen i C påminner mycket om den med WOW. Det finns två varianter att realisera exemplet i C. Den version som presenteras här använder separata gatewayprogram för varje funktion. I den andra varianten används bara ett program och istället används environmentvariabeln PATH_INFO för att välja vilken funktion i programmet som skall exekveras. Endast två program, *newURLform* och *newURL* beskrivs då övriga programmen är uppbyggda på liknande sätt. I programmen finns inte heller någon felhantering för att de inte ska bli för stora.

Som i alla C-program innehåller *newURLform* en huvudfunktion *main* (nedan). Uppbyggnaden påminner till stora delar om databasproceduren *new_URL_form* i WOW-versionen. Det som skiljer är bl a att programmet måste etablera en förbindelse med databasen för att kunna komma åt den. Detta görs i funktionen *dbLogin* och nedkopplingen görs i *dbLogoff*. En annan skillnad är att HTML-koden skapas direkt i programkoden och inte via några funktioner.

```
int main()
{
  /* Ett CGI-program MÅSTE skriva ut raden nedan innan något annat skrivs ut
   * annars ger tolkar serverprogrammet utdatat som felaktigt */
  printf("Content-type: text/html\n\n");
  htmlhead("Ny URL");

  dbLogin();

  printf("<h1>Lägg in en ny URL i databasen</h1>\n");
  printf("<form action=\"/cgi-bin/newURL\" method=POST>\n");
  printf("Adress:<br><input name=address size=60><p>\n");
  printf("Kort beskrivning:<br><input name=short_desc size=60><p>\n");
  printf("Lång beskrivning:<br>\n");
  printf("<textarea name=long_desc rows=5 cols=60></textarea><p>\n");
  printf("Kategori:<br>");
  dbListcategories();
  printf("<p><input type=submit value=\"Lägg in URLen\">\n");
  printf("</form>\n");

  dbLogoff();

  htmlend();
  return 0;
} /* main */


void dbLogin()
{
  /* ORAUSER innehåller användarid och lösenord på Oracleanvändaren som används
     för detta program. Ex "scott/tiger" */
  orlon(&lda, hda, (text *) ORAUSER, -1, (text *) 0, -1, 0);
  oopen(&cda, &lda, (text *) 0, -1, -1, (text *) 0, -1);
} /* dbLogin */

void dbLogoff(void)
{
  oclose(&cda);
  ologof(&lda);
} /* dbLogoff */
```

Funktionen *dbListcategories* har samma uppgift som proceduren *category_list* i WOW-versionen. Den är mer komplicerad i C-versionen eftersom stödet för databasfrågespråket SQL inte är lika bra i C som i PL/SQL.

```
void dbListcategories()
{
  text sql[2000];
  int id;
  char name[100];
  ub2 rlen;

  printf("<select name=category>\n");

  /* Bygg frågan och bind de variabler som ska hämtas från databasen */
  sprintf(sql, "select id,name from categories");
  oparse(&cda, sql, -1, FALSE, 2);
  odefin(&cda, 1, (ub1 *)&id, sizeof(int), 3, -1, 0, 0, 0, -1, 0, 0);
  odefin(&cda, 2, (ub1 *)name, 100, 1, -1, 0, 0, 0, -1, &rlen, 0);
  oexec(&cda);

  /* Så länge det finns mer data att hämta så läs in denna och lägg ut det som
     ett val i listrutan */
  for(;;) {
    if (ofen(&cda, 1))
      break;
    name[rlen] = '\0';
    printf("<option value=%d>%s\n", id, name);
  }
  printf("</select>\n");
} /* dbListcategories */
```

Detta är allt som behövs för programmet *newURLform*. Även programmet *newURL* har en *main*-funktion. Till stor del liknar den *main* i *newURLform*, fast den är ännu enklare. Först så ska parametrarna från formuläret avkodas. Detta görs i funktionen *parsepostinput*. Sedan ska länken läggas in vilket görs av funktionen *dbInsert*.

```
int main(int ac,char **av)
{
  printf("Content-type: text/html\n\n");
  htmlhead("Ny URL");

  /* Ta hand om parametrarna */
  parsepostinput();

  /* Koppla upp programmet mot databasen, lägg in URL:en och koppla sedan
     ner igen */
  dbLogin();
  dbInsert(address, short_desc, long_desc, category);
  dbLogoff();

  /* Generera sidan med informationen att databasen är uppdaterad */
  printf("<h1>Den nya URL:en är nu inlagd</h1>\n");
  printf("<a href=\"/\">Tillbaka</a>\n");

  htmlend();
  return 0;
} /* main */
```

*parsepostinput* avkodar parametersträngen som skickats från formuläret. Denna funktion hanterar enbart parameteröverföring enligt POST-metoden. Funktionerna *fmakeword*, *plustospace* och *unescape_url* som anropas ingår i NCSA:s serverprogram och är till för att avkoda strängen med parametrar. I filen *util.c* finns källkoden till dessa funktioner och denna går att hämta från bl a adressen http://ftp.sunet.se/ftp/pub/www/servers/ncsa_httpd/Unix/ncsa_httpd/cgi/cgi-src/util.c.

```
void parsepostinput()
{
  int i;
  int cl;
  char *value, *name;

  cl = atoi(getenv("CONTENT_LENGTH"));

  for(i=0;cl && (!feof(stdin)); i++) {

    /* Läs in ett fältnamn */
    name = (char *) fmakeword(stdin,'=',&cl);
    plustospace(name);
    unescape_url(name);

    /* Läs in värdet för fältet */
    value = (char *) fmakeword(stdin,'&',&cl);
    plustospace(value);
    unescape_url(value);

    /* Avgör vilken variabel det var och spara undan värdet i rätt globala
       variabel*/
    if (strcmp(name, "category") == 0)
      category = atoi(value);
    else if (strcmp(name, "short_desc") == 0)
      strcpy(short_desc, value);
    else if (strcmp(name, "long_desc") == 0)
      strcpy(long_desc, value);
    else if (strcmp(name, "address") == 0)
      strcpy(address, value);
  }
} /* parsepostinput() */
```

Funktionen *dbInsert* lägger in en URL i databasen. Den bygger först upp SQL-frågan och utför den sedan.

```
void dbInsert(char *address, char *short_desc, char *long_desc, int category)
{
  text sql[2000];

  /* Bygg upp SQL-frågan i en sträng */
  sprintf(sql, "insert into url values (url_seq.nextval, "
    "%d,'%s','%s',:1)", category, address, short_desc);

  /* Bind variabeln long_desc till frågan och utför sedan frågan. Egentligen
     skulle inte denna variabel behöva bindas på detta sättet utan skulle kunna
     ingå direkt i frågan. Detta görs bara för att visa hur det kan gå till. */
  oparse(&cda, sql, -1, FALSE, 2);
  obndrn(&cda, 1, (ub1 *)long_desc, strlen(long_desc), 1, -1, 0,
   (text *) 0, 0, -1);
  oexn(&cda, 1, 0);
} /* dbInsert */
```

# 5 Program för att underlätta databaskopplingar

Att bygga en interaktiv WWW-tjänst från grunden kan innebära mycket arbete. Det finns färdiga program som kan användas för att minska arbetsbördan och snabba upp utvecklingen, både några fria och några kommersiella. Generellt är de fria enklare och lämpar sig framförallt för att sätta upp tjänster för sökning i tabeller medan de kommersiella är mer kompletta och några av dem är helt färdiga tillämpningar.

## 5.1 Fria programvaror

Till några databashanterare finns färdiga gatewayprogram som kan användas för att bygga tjänster med WWW och databaser. De är gratis att hämta ner från Internet men är dock inte supportade. Programmen är av olika kvalitet och de har olika möjligheter och begränsningar. De enklaste kan bara användas för att visa innehållet i databasen medan de mer avancerade även klarar att lägga in ny information och uppdatera databasen.

Det gäller att sätta sig in i hur programmet fungerar. Framförallt i de programmen som tillåter att databasen uppdateras. Det kan nämligen finnas problem med säkerheten i dem. Genom att ge databasanvändaren för stora rättigheter kan en användare till och med ta bort hela tabeller.

### Oracle

Oracle har på sin WWW-server fler olika färdiga gatewayprogram att hämta. Adressen dit är http://dozer.us.oracle.com:8080.

- *WOW* finns att hämta på Oracles WWW-server. Det är ett gatewayprogram som gör det möjligt att anropa databasprocedurer i PL/SQL som är Oracles eget programmeringsspråk. Det ingår också PL/SQL-procedurer för att skapa HTML-kod. I och med att tillämpningarna måste skrivas i ett programmeringsspråk så krävs det att man kan programmera. Fördelen med att använda WOW och PL/SQL jämfört med andra språk är att det är enklare att komma åt innehållet i databasen. Annars är det som vilket annat procedurellt programmeringsspråk som helst.
- I *Oraywww* skapar man enkelt databaskopplingar från formulär. Kontrollen över frågor och utseende är mycket begränsat. Mer information om Oraywww finns på Oracles WWW-server.
- På Moulon WWW-server (http://moulon.inra.fr/oracle/www_oracle_eng.html) finns några olika gatewayprogram för att kombinera WWW och Oracles databas. Skillnaden mellan dem ligger i hur pass avancerade de är. Den enklaste kan i stort sett bara användas för att visa innehållet i en databas och den mest avancerade kan lägga till och ta bort information i databasen. På Oracles WWW-server finns ett av dessa gatewayprogram att hämta och går där under namnet *Decoux*.

### Sybase

Från Sybases WWW-server (http://www.sybase.com) finns det referenser till några olika gatewayprogram och några exempel på hur de används.

- *WDB* kan användas för att söka och presentera information i en databas. Det går inte att lägga in ny information i databasen från WDB. Val av tabeller och fält samt utseende på resultat görs i konfigurationsfiler. Det finns hjälpprogram för att skapa filer som man kan utgå ifrån. Mer information går att hitta på adressen http://arch-http.hq.eso.org/bfrasmus/wdb/wdb.html.
- I *GSQL* söker man från formulär i databasen. I konfigurationsfiler beskrivs vilka fält som ska tas med och hur formuläret ska se ut. Mer information om GSQL finns på adressen http://www.ncsa.uiuc.edu/SDG/People/jason/pub/gsql/starthere.html.
- I *Web/Genera* bygger man upp schemafiler som beskriver hur man kan söka och presentera databasen. Information finns på adressen http://gdbdoc.gdb.org/letovsky/genera/genera.html.

### Informix

- Stöd för Informix databashanterare i WDB finns i alfaversion. Se information om denna i avsnittet om Sybase.

- GSQL finns i en version för Informix också. Se avsnittet om Sybase för mer information om GSQL.

### Butler SQL

- Everyware har lanserat ett gatewayprogram för sin databashanterare Butler SQL på Macintosh kallat ButlerLink/Web. Det går att söka, uppdatera och ta bort information lagrat i databasen. Via ett program för att bygga upp formulär och träfflistor byggs tillämpningarna upp. Mer information om ButlerLink/Web finns på http://www.everyware.com.

## 5.2 Kommersiella programvaror

Det finns flera kommersiella produkter som har färdiga eller halvfärdiga koncept för att bygga informationstjänter på Internet.

### InterNotes Web Publisher från Lotus

InterNotes Web Publisher kan konvertera Lotus Notes vyer (views) och dokument till HTML-dokument som sedan kan distribueras av en WWW-server. Den hanterar länkar mellan olika dokument, tabeller, bilder och tillägg i dokument. Vid ändringar inne i Notes-systemet kommer automatiskt HTML-dokumenten att uppdateras. Mer information finns på adressen http://www.lotus.com.

### Topic WebSearcher från Verity Inc.

Med Topic WebSearcher kan man kombinera WWW med avancerade informationsagenter för att söka och hämta dokument lagrade i ett stort antal format från formulär. Dokument i olika format kan konverteras för att visas i en WWW-läsare. På Veritys WWW-server (http://www.verity.com) finns mer information.

### BASIS WEBserver från Information Dimensions

BASIS WEBserver bygger på BASISplus dokumentdatabas och lägger på funktioner för att söka efter dokument lagrade i databasen från WWW. Dokumenten kan vara lagrade i flera olika dokumentformat men de konverteras automatiskt till HTML när de ska visas. Hela dokumenten överförs inte heller utan enbart de delar som är intressanta för användaren. Mer information finns på Information Dimensions WWW-server på adressen http://www.oclc.org/oclc/idi/idihome.htm.

### PLWeb från Personal Library Software (PLS)

PLWeb är en fulltextdatabas som kan användas för att hitta relevant information snabbt. PLWeb har en distribuerad databasmodell och man kan transparent söka i flera separata PLWeb-servrar som befinner sig på olika ställen. I databasen kan HTML-, Acrobat- och ASCII-dokument indexeras. Det finns även inbyggt stöd för olika debiteringsmodeller och för att kontrollera behörighet. Det finns flera olika modeller för att ange sökvillkor som t ex naturligt språk och "Query by Example". För mer information se http://www.pls.com.

### Netscape Application Servers

Netscape har fyra olika WWW-servrar som är speciellt byggda för olika tillämpningar. De servrar som finns är Community System för arbetsgrupper, Publishing System för tidningsutgivning, Merchant Server och Istore för elektronisk handel. Mer information finns på Netscapes WWW-server (http://home.netscape.com).

### WebDBC från Nomad Development Corporation

WebDBC är ett CGI-program för Windows NT-baserade WWW-servrar. Det är gatewayprogram för att koppla ODBC-databaser till WWW. Nomad finns på Internet på adressen http://www.ndev.com/ndc.

### WWW DB2 från IBM

Med WWW DB2 kan data från DB2-databaser presenteras på WWW. Ett procedurellt språk används för att beskriva kopplingen mellan SQL och HTML. Det finns funktioner för att skapa grafik från innehållet i

databasen så att dokument där text blandas med grafik kan genereras. På IBM:s WWW-server (http://www.ibm.com) finns mer information.

### Surfboard från Fulcrum Technologies

Med Surfboard kan man snabbt söka och presentera dokument lagrade i olika format via ett WWW-gränssnitt. Förutom boolska sökningar kan frågor ställas i naturligt språk. Fulcrum har adressen http://www.fultech.com där mer information finns.

### WWW to Texis Bridge från Thunderstone Software – EPI, Inc

Thunderstone Software har en gateway mellan WWW och sin relationsdatabas Texis. Denna gateway gör det möjligt att söka och uppdatera databasen. Mer information om WWW to Texis Bridge finns på Thunderstones WWW-server (http://www.thunderstone.com).