VOLVO

UTGIVARE: Volvo Data AB
Avd 2540 Database Systems

NYHETSBREVET
DATAbasen

Nyhetsbrevet DATAbasen - Juni - 1989

GLAD SOMMAR!
Den här gången har vi med en lite längre artikel om distribuerade databaser. Ett ämne som blivit mer och mer aktuellt. Lämplig läsning för hängmattan kanske?

Än en gång, med risk för att bli tjatig upprepar jag bönen från tidigare nummer, vi vill gärna att Du bidrar med material att publicera i DATAbasen. Det kan vara allt ifrån artiklar till kortare frågor som har allmänt intresse. Och kom ihåg, några litterära mästerverk krävs inte, huvudsaken är att det går att läsa.

Planerade kvarvarande utgivningstillfällen för 1989 är:

- W939
- W950

Manusstopp för material att publicera är 14 dagar innan varje utgivning. Skicka ett memo till memoid **VD.DATABAS** och tala om var Ditt bidrag finns, så tar vi in det i nästa nummer.

DATAbasen distribueras till dom för oss kända IMS- och DB2-kontakterna. Är det någon som vi missat, sänd ett memo till **VD.DATABAS** så ordnar vi det till nästa gång.

För formens skull påpekas till sist att självklart står varje artikelförfattare för respektive artikels innehåll, vi på 'redaktionen' utövar ingen censur eller kontroll, möjligen lite redigering av insänt material.

Med vänlig hälsning

Henrik Holst / 2540 DATABASE SYSTEMS

Distribuerade databaser, vad är det?

Jag ska här försöka kasta lite ljus över ett aktuellt ämne. Alla talar mer eller mindre om distribuerade databaser, men vad är det egentligen? Det finns många uppfattningar och varianter har det visat sig. Först dock några vanliga förkortningar i sammanhanget:

DBMS DataBase Management System
Allmän benämning på databassystem, tex DB2 eller IMS/DB

RDBMS Relational DBMS, tex DB2 eller SQL/DS

DDBMS Distribuerad DBMS

Är DDBMS ett nytt exempel på principen 'en lösning i behov av ett problem'?

Nja, inte riktigt. Låt oss fördjupa oss lite i ämnet.

Först och främst: Distribuerad datakraft skall inte förväxlas med DDBMS. Det har länge funnits mer eller mindre distribuerad databehandling, men som Chris Date, känd dataguru, uttryckt saken: "Sömmarna syns", dvs användaren, eller applikationsprogrammet, är på något sätt medveten om att datat är distribuerat. Så är inte fallet med DDBMS; För användaren ser en distribuerad databas exakt likadan ut som om den inte var distribuerad.

Teknologin för att etablera DDBMS finns nu eller är på väg:

- Relationsdatabasen
- ett aktivt data dictionary för att "separera" användaren från datat
- standardiserat språk (SQL) för att manipulera data
- nätverk, utgörande de "motorvägar" på vilka datat skall transporteras

Trycket från den affärsmässiga sidan att etablera DDBMS utgörs främst av en allt mer tilltagande globalisering av företagens verksamhet. Som exempel på detta kan nämnas den 31 december 1992, då EG har rivit alla interna handelshinder. Detta kommer att innebära en hemmamarknad vilken på sikt kommer att bli större än USA.

Men vi är inte framme vid DDBMS än. Diskussionen för dagen förs till största delen i tekniska termer, och detta är lite grand av att spänna vagnen framför hästen. Fler stora frågor är fortfarande kvar att lösa innan DDBMS kan utnyttjas fullt ut.

De fundamentala frågor som måste ersätta nuvarande mer tekniska resonemang är:

- vem äger vilket data?
- hur struktureras applikationsövergripande data?
- vilken teknik skall användas för att bygga de nödvändiga datastrukturerna?
- etc. etc.

Ett annat sätt att uttrycka det är genom termen "corporate data structure", en datastruktur som innefattar hela företaget.

Migreringsfrågor: Det finns ingen enkel väg från nuvarande strukturer, dvs IMS-baserade systemlösningar, till RDBMS och DDBMS, trots att en del migrationshjälpmedel har sakta börjat se dagens ljus.

Vi kan inte räkna med någon massiv etablering av DDBMS på kort sikt, beroende på de tekniska och framför allt organisatoriska problem som finns. På kort sikt är det mer troligt att DDBMS implementeras delvis, typ read-only-access till distribuerat data.

Tekniska frågor: Ted Codd, "uppfinnaren" av relationsdatabasen, har definierat 12 regler som ett DBMS skall uppfylla för att kallas 'relational'. Hans kollega Chris Date har tagit fram ytterligare 12 regler som definierar en distribuerad databas (DDBMS). Date framhåller dock att "... det finns inget 'slutgiltigt distribuerat databassystem'. Olika användare kommer att framhålla olika kriterier som viktiga."

Bakom dessa 12 regler finns 'regel 0', som säger att ett distribuerat system skall upplevas av användaren som ett icke-distribuerat system.

De 12 reglerna för ett DDBMS är:

1. Lokal autonomi
2. Inget beroende av en central datormiljö

3. Avbrottsfri drift
4. Transparent dataplacering
5. Oberoende av fragmentering av data
6. Oberoende av replikation av data
7. Distribuerad query-hantering
8. Distribuerad transaktionshantering
9. Hårdvaru-oberoende
10. Operativsystem-oberoende
11. Nätverk-oberoende
12. RDBMS-oberoende

Beståndsdelarna i ett DDBMS: Följande grundläggande byggstenar utgör grunden för ett DDBMS:

1. Ett relationsdatabassystem (RDBMS)

2. Ett aktivt data dictionary

3. Språk, Structured Query Language (SQL)
   Det finns både för och nackdelar med SQL:
   + 'Set-at-a-time processing'
   + Inbyggd säkerhetshantering
   - Strukturen uppmuntrar till flera, redundanta sätt att ställa en fråga
   - Ej stöd för primary-key/foreign-key eller tillhörande integritetsproblem

4. Nätverk
   Peer-to-peer-kommunikation (APPC, LU6.2 kallas det också ibland) har accepterats bland sw-företagen som standard.

5. Mekanism för att hantera en distribuerad databas
   Olika tekniker finns för att 'dela upp' data:
   - kopior i olika former (snapshots, extrakt o.d.),
   - partitionering
   - horisontell, dvs uppdelning map rader i en tabell
   - vertikal, dvs uppdelning map kolumner i en tabell

6. Transaktionshanterare
   För att kunna säkerställa data-integriteten i en transaktionsomgivning krävs funktioner för bl.a synkronisering av accesser och backup/recovery.

Distributionsnivåer enligt IBM: Följande nivåer eller ambitionsgrader har definierats av IBM, och kommer att stödjas av 'de 4 databassystemen', dvs DB2 (MVS-miljön), SQL/DS (VM och VSE), OS/400 (AS/400) samt OS/2 EE (PS/2). Först ut är DB2 (i ver2 rel3) med begränsad support, när de övriga följer får vi se.

- Remote-unit-of-work
  Ett databassystem per SQL-anrop och COMMIT-scope
- Distributed-unit-of-work
  - 'single site' per SQL-anrop och
  - 'multiple site' per COMMIT-scope och
  - 'single-site-update'

  F.n. endast DB2 ver 2 rel 2, men allt eftersom stöd för s.k. 2-phase COMMIT läggs in i IBM's övriga DBMS (se ovan) kommer dessa att bli likställda med DB2 i denna betydelse.
- Distributed-request
  - 'multiple-site' per SQL-anrop och
  - flera SQL-anrop per COMMIT-scope

  Detta är svårt. Kommer ej i någon IBM-rodukt på länge.

Hårdvaruplattformar: DDBMS implementeras på stordatorer (framför allt IBM/370), mid-range (VAX o.d. dominerar), och PC-baserade LANs (lokala nät). Stordatormiljön domineras av IBM med DB2, men flera oberoende softvaruföretag förekommer här också. DEC gör inbrytningar på denna plattformen med VAX-clusters.

Mid-range med VMS som ledande sw-plattform (men även UNIX) domineras av företag som Oracle, RTI, Sybase och Tandem.

Den tredje plattformen, PC-baserade LANs, har haft liten betydelse än, men där kommer sannolikt PS/2 med OS/2 EE, Presentation Manager och MS-LAN att vara nyckel- produkter.

Stegvis utveckling av DDBMS: Utvecklingsvägen, hur DDBMS skall byggas, följer väl den modell över informations-systemens utveckling, som beskrivits av Nolan 1982. Sex stadier finns beskrivna:

1. Automatisera grundläggande affärsfunktioner
2. De flesta operationella funktioner har automatiserats var för sig med stor dataredundans som ett resultat
3. Försök att kontrollera det kaos som uppstått efter 1 och 2 ovan.

4. Existerande applikationer börjar struktureras om för att ta till vara databas-teknologin
5. Alla datasystem delar en central kärna av 'corporate data'
6. Total separation av data och de datoriserade processer de deltar i

Problemet är att komma förbi steg 4. Detta kan endast göras genom att etablera en övergripande funktion för data-administration, som skall implementera och sköta en 'corporate data structure'.

Slutsatser: En säker slutsats är att detta är ett mycket svårt ämne, det finns inga enkla avgöranden med i bilden. Den bästa rekommendationen som kan ges i detta skede är nog att alla försök att etablera en DDBMS-miljö skall övervägas **mycket** noga utifrån både tekniskt och framför allt organisatoriskt håll.

/HH

Några frågor till IBM

Följande saxat från DATA BASE NEWS-LETTER, maj/juni 89. Den som svarar på frågorna är Donna van Fleet, Director of Systems Products IBM, Santa Terasa Lab, San Jose.

Fråga: Kommer det att bli möjligt att köra existerande DL/1 applikationsprogram mot DB2 baser (online)?

Svar: Vi har undersökt användningen av DL/1-anrop mot relationsdata och det ser svårt ut att hitta en generell lösning. Även en inte helt komplett lösning kan innebära integritetsproblem. Men, vi har gjort utomordentliga framsteg vid användning av SQL för enbart läsning av DL/1-data.[1]

Fråga: Ser IBM någon chans att antingen Repositoryt ' eller DB2 kommer att bli en obligatorisk del av MVS?

Svar: Vi kommer inte att ta ett sådant beslut om inte våra kunder anser det vara det mest effektiva sättet att leverera produkten. Vi har inte annonserat Repositoryt än, så jag kan inte kommentera den delen av frågan.

(Ett till intet förpliktande svar kan man säga.)

Fråga: Ge exempel på problem som kan lösas med den nivå av distribuerade databaser som IBM nyligen annonserat.

Svar: Ett exempel på 'remote-unit-of-work' (se annan artikel) är en applikation som söker igenom en inventarielista i en lokal databas. Om den sökta artikeln inte finns i den lokala inventarielistan, kan applikationen söka i en extern databas och om artikeln finns där kan den bokas och skickas och den externa databasen uppdateras i enlighet därmed.

Ett exempel på 'distributed-unit-of-work': Överföring av medel mellan bankkontor. Tex vill användaren ta ut ett belopp från New York-kontoret och sätta in en del därav på Paris-kontoret och resten på Tokyo-kontoret. Användaren vill att både uttag och insättningar skall ske inom samma 'Commit-scope', dvs om någon del av transaktionen misslyckas skall samtliga tre delar backas ur.

/HH

---

[1] I andra sammanhang har det framkommit att IBM jobbar på och planerar att släppa SQL-gränssnitt, både från QMF och applikationsprogram, mot DL/1-data. (Claudia Baker/IBM St.Teresa Lab vid GUIDE/Basel juni-89)

Repository - följetongen

Från Gartner Group kommer följande karaktäristik av det ännu inte annonserade Repositoryt i ljuset av SAA.

"SAA är egentligen bara en uppsättning standards för gränssnitt mot maskinen bl a. Repositoryt kommer att bidraga med standards för att accessa, uppdatera och manipulera meta-data, som beskriver relationerna mellan och strukturerna på SAA-objekt, och specifiera var dessa objekt är placerade. Mer påtagligt kommer R att innebära introduktionen av ett nytt 'Common Programming Interface' (CPI) i SAA för access och manipulation av meta-data som lagras där.

Den övergripande designen av Repositoryt ser ut så här:
R kommer att ha tre komponenter:

1. Repository databas med ett '3-level-schema'[2] Denna kommer att bygga på DB2.
2. Repository Manager, som kommer att bli åtkomlig för omvärlden genom det nya gränssnittet 'the Repository Services Common Programming Interface (CPI)'. Genom detta gränssnitt kommer användare och oberoende leverantörer att kunna manipulera Repository-objekt.
3. Repository applikationer. IBM kommer att börja med fyra stycken: information management (problem/change control bl a), systemutveckling, nätverkshantering och 'systems management' (dvs konfiguration etc.)

Det finns ingen konceptuell invändning mot denna konstruktion. Problemet är IBM's möjlighet att leverera alla dessa funktioner, eller ens en användbar delmängd av dom, inom rimlig tid. Även om IBM annonserar Repositoryt i höst, vilket börjar se tveksamt ut, kommer det att dröja till 1992 som tidigast innan användare blir märkbart påverkade och viktiga oberoende softvaruleverantörer implementerar support för R. Användare kan inte vänta till dess för att formulera sin strategi för systemutveckling."

/HH

---

[2] '3-level-schema' innebär i samband med Repositoryt följande nivåer:
Fysisk lagrings-nivå som är databasadministratörens område, **logisk nivå**, som riktar sig först och främst till dom som tar fram verktyg (tools), och **koncept-nivån** som är en data-entry-nivå; den erbjuder ett enkelt sätt att definiera information för Repositoryt och är databasadministratörens område.

Nyheter i DB2-Handboken

DB2 handboken är nu översatt till engelska och finns på samma ställe som den svenska, dvs under val V.3.H. Du får upp den genom att skriva ENGLISH på kommandoraden och byter tillbaka till den svenska genom att skriva SWEDISH på samma ställe.

Printningen av handboken är omgjord. Vid print ALL fås numera alla kapitel, 1-9, och detta till en mycket lägre kostnad än tidigare. Dessutom är innehållsförteckningen omgjord och lättare att hitta i.

Den nya versionen av DB2 handboken finns i alla miljöer på Volvo Data och kommer att distribueras ut till MEXPACK kunderna i samband med installationen av DB2 Version 2.

/ Yvonne

VBACKUP - för tabellunderhåll i DB2

Ett verktyg, speciellt för slutanvändare, för att göra underhåll av tabeller. Det visar om IMAGE COPY har tagits och isåfall vilket datum den senaste IC'n är ifrån. Genererar och submittar JCL för IMAGE COPY eller för RECOVER av TABLESPACE och INDEX.

Man behöver här inte veta vare sig databas- eller tablespacenamn, utan anger bara CREATOR. En generisk sökning på tabellnamn är också möjlig,

Därefter kan man göra ett selektivt urval på vilka tabeller som det skall tas IC eller göras RECOVER på.

Denna funktion finns på V.3.O.9 under ISPF och som kommandot VBACKUP i QMF.

/Yvonne

Innehållsförteckning