Denne guide er oprindeligt
udgivet på Eksperten.dk

COMPUTERWORLD

Forskellige databaser

Denne artikel beskriver kort forskellige database typer, produkter og API'er. Målet er at give et overblik over hvad der findes af muligheder. Der gåes ikke i detaljer.

Den forudsætter ikke kendskab til databaser i forvejen.

Skrevet den **16. Feb 2010** af **arne_v** I kategorien **Databaser / Generelt** | ★★★☆☆

Historie:
V1.0 - 19/08/2004 - original
V1.1 - 16/02/2010 - smårettelser

Database typer

Relations database (RDBMS) = database baseret på den relationelle model (tabeller som joines med andre tabeller udfra fremmed nøgler som peger på primær nøgler)

Object Orienteret database (OODBMS) = database baseret på objekter i en form svarende til den der bruges i objekt orinteret programmering.

Hierakisk database = ældre form for database som jeg ikke vil komme nærmere ind på

De relationelle database slog igennem i 1980'erne og er idag totalt dominerende. De ældre database typer er udgået. Og objekt orienterede database har ikke slået igennem (på trods af at objekt orienteret programmering absolut har slået igennem).

Database server = database hvor applikationerne kommunikerer med en database server process som er ansvarlig for at læse fra og skrive til fil

Fil database = database hvor applikations processen selv er ansvarlig for at læse fra og skrive til fil

Embedded database = reelt et synonym for en fil database (man bruger bare mest fil database om database produkter som Access etc. og embedded database om ikke database produkter som inkluderer en database)

En database server performer normalt meget bedre end en fil database med mange samtidige brugere. Til gengæld er der også normalt meget mere administration med at installere, konfigurere og starte.

Relations database er tæt knyttet til SQL sproget med kommandoer som:

```
SELECT f1,f2 FROM t1,t2 WHERE t1.f1=t2.f1 AND t2.f3 > 100 ORDER BY t1.f1;

INSERT INTO t1 (f1,f2) VALUES (123,'ABC');

UPDATE t1 SET f2='DEF' WHERE f1=123;

DELETE FROM t1 WHERE f1=123;
```

Databaser

[note: nogle af nedenstående betragtninger om pris, udbredelse og funktionalitet
er mine subjektive vurderinger som man selvfølgelig ikke skal acceptere ukritisk]

Alle nedenstående databaser er relations databaser.

Oracle - kommerciel database server fra Oracle, kan klare virkeligt store datamængder og stor transaktions volumen, meget dyr, meget udbredt til store e-business løsninger og lignende

IBM DB2 - kommerciel database server fra IBM, kan klare virkeligt store datamængder og stor transaktions volumen, meget dyr, meget udbredt i bl.a. den finansielle sektor

Sybase ASE - kommerciel database server fra Sybase, professionel database, dyr, da MS SQLServer nedstammer fra Sybase så kan man også Sybase hvis man kan MS SQLServer, var tidligere kendt som Sybase SQLServer men ændrede navn for ikke at blive forvekslet med MS SQLServer, ikke særligt udbredt i Danmark

MS SQLServer - kommerciel database server fra Microsoft, professionel database, dyr i de fleste versioner men der findes dog en gratis version MSDE som kan downloades, meget udbredt til alt muligt hvor der er brug for kvalitet til en rimelig pris

Informix - kommerciel database server som idag ejes af IBM, professionel database, dyr, ikke udbredt i Danmark

Oracle RDB - kommerciel database server som idag ejes af Oracle, professionel database, dyr, kommer oprindeligt fra Digital og var kendt som DEC RDB, meget lidt kendt udenfor en snæver kreds(bl.a. fordi den kun kører på VMS styresystem)

MySQL - open source database server, meget udbredt specielt til web løsninger, har flere slags tabeller bl.a. MyISAM (default, understøtter ikke transaktioner) og InnoDB (understøtter transaktioner)

PostgreSQL - open source database server, ikke så kendt som MySQL (jeg plejer at sammenligne MySQL-PostgreSQL med Linux-FreeBSD)

Firebird - open source database server / fil database, bygger på Borland Interbase

SQLite - open source fil database, bruges f.eks. i FireFox og Thunderbird

MS Access - kommerciel fil database fra Microsoft, Access består af selve databasen (Jet) og et applikations værktøj (forme, reports, VBA), ikke dyr og mange har den i forvejen som en del af Microsoft Office Professional, utroligt udbredt

Hypersonic/HSQLDB/H2 - open source fil database, lavet i Java, HSQLDB er den underliggende database i OpenOffice Base

Cloudscape/Derby/JavaDB - open source database server / fil database, lavet i Java, kommer med Java SE

Database API'er

Så godt som alle databaser idag inklusive alle de ovennævnte bruger SQL som sprog. SQL er en helt universelt anerkendt standard for database sprog. Men selvom sproget er standardiseret så er de API kald man bruger til at eksekvere SQL sætningerne med ikke standardiseret.

Vær dog opmærksom på at de forskellige databaase har lettere forskellige SQL dialekter. Selvom man bruger SQL med API X mod database A så er det ikke sikkert at koden uændret kan køre mode database B selvom den også understøtter API X.

database specifikke API'er - mange database kommer med deres egne API'er. Eksempler er: Sybase ASE (DB lib og CT lib), MS SQLServer (DB lib), MS Access (DAO lib), MySQL (MySQL lib). De har dog aldrig været så meget brugt, idet der altid har været et stort ønske om brug af standard API.

embedded SQL - dette var i mange år det meste brugte og er stadigvæk udbredt på store maskiner (MVS, OS/400, VMS, Solaris, AIX, HP-UX, Tru64), men har aldrig slået igennem på Windows og Linux, fordi da de kom på banen var der masser af alternativer. Ideen er at man skriver sit Cobol/C/Fortran/PLI program og tilføjer linier med almindelige SQL sætninger prefixet med EXEC SQL midt i koden, så kører man en preprocesser som konverterer de embeddede SQL sætninger til et database specifikt API, hvorefter man kalder den almindelige compiler.

ODBC - kom frem tidligt til Windows (eksisterede allerede i 16 bit Windows). Det er et standard C API for database kald. Så godt som alle database som fåes til Windows platform leveres med ODBC drivere. Der er også blevet lavet ODBC til Linux, men det har ikke nær den samme udbredelse. Når folk snakker om ODBC mener de i virkeligheden tit diverse overbygninger oven på ODBC f.eks. ADO oven på OLE DB oven på ODBC.

Eksempel C kode ODBC API med DSN:

```
SQLHENV Environment;
SQLHDBC DataBaseConnect;
SQLHSTMT stmt;
SQLRETURN stat;
char *dsn = "MinDSN";
char *un = "mitbrugernavn";
char *pw = "mitpassword";
char *sqlstr = "SELECT * FROM T1";
int f1;
char f2[50];
int f1len,f2len;
stat = SQLAllocEnv(&Environment);
stat = SQLAllocConnect(Environment,&DataBaseConnect);
```

```
   stat = SQLConnect(DataBaseConnect,
                     (SQLCHAR *)dsn,(SQLSMALLINT)strlen(dsn),
                     (SQLCHAR *)un,(SQLSMALLINT)strlen(un),
                     (SQLCHAR *)pw,(SQLSMALLINT)strlen(pw));
   stat = SQLAllocStmt(DataBaseConnect,&stmt);
   stat = SQLExecDirect(stmt,(SQLCHAR *)sqlstr,strlen(sqlstr));
   stat = SQLBindCol(stmt,1,SQL_C_LONG,&f1,sizeof(f1),(SQLINTEGER *)&f1len);
   stat = SQLBindCol(stmt,2,SQL_C_CHAR,f2,sizeof(f2),(SQLINTEGER *)&f2len);
   for(;;)
   {
      stat = SQLFetch(stmt);
      if((stat!=SQL_SUCCESS)&&(stat!=SQL_SUCCESS_WITH_INFO)) break;
      f2[f2len] = '\0';
      /* f1 og f2 indeholder nu data */
   }
   SQLFreeStmt(stmt,SQL_DROP);
   SQLDisconnect(DataBaseConnect);
   SQLFreeConnect(DataBaseConnect);
   SQLFreeEnv(Environment);
```

OLE DB - er en Microsoft standard beregnet til at erstatte ODBC. Den er kun til Windows. Alle Microsofts databaser plus de mest betydningsfulde databaser leveres med OLE DB driver. Af kompabilitets årsager er det muligt at bruge OLE DB oven på ODBC. OLE DB bruges primært med en overbygning i form af ADO.

ADO - er en ActiveX overbygning til OLE DB som gør det nemt at bruge OLE DB fra sprog som MS Access VBA, ASP VBScript, VB etc..

Eksempel VBScript kode ADO API med DSNless mod MS Access database:

```
Set con = Server.CreateObject("ADODB.Connection")
con.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=MinDatabase.mdb;User
Id=mitbrugernavn;Password=mitpassword;"
Set rs = Server.CreateObject("ADODB.Recordset")
rs.Open "SELECT * FROM T1",con
Do While Not rs.EOF
   f1=rs("F1")
   f2=rs("F2")
   ' f1 og f2 indeholder nu data
   rs.MoveNext
Loop
Set rs = Nothing
Set con = Nothing
```

JDBC - er et standard Java API for database kald. Det er stort set det eneste low level API for databaser i Java (embedded SQL i Java slog aldrig igennem). Stort set alle database uanset platform leveres med JDBC drivere med en markant undtagelse MS Access. Ofte bruges der forskellige overbygninger oven på JDBC. Der findes en bridge så man kan bruge JDBC oven på ODBC

(den virker dog ikke specielt godt).

Eksempel Java kode JDBC API med MySQL database:

```
        Class.forName("com.mysql.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/MinDatabase",
"mitbrugernavn", "mitpassword");
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT * FROM T1");
        while(rs.next()) {
            int f1 = rs.getInt(1);
            String f2 = rs.getString(2);
            // f1 og f2 indeholder nu data
        }
```

CMP entity EJB - sværvægts komponent baseret persisterinsg framework som bygger oven på JDBC. Ideen er at man har en speciel Java klasse og at man via en XML fil beskriver hvordan felterne i klassen skal mappes til felter i en database tabel. Frameworket sørger så selv for at konvertere forskellige kald til de rette JDBC kald. Da det er en selvstændig komponent kan det tilgåes både lokalt og remote.

Hibernate - letvægts objekt orienteret persisterings framework som bygger oven på JDBC. Ideen er at man har en normal Java klasse og at man via en XML fil beskriver hvordan felterne i klassen skal mappes til felter i en database tabel. Frameworket sørger så selv for at konvertere forskellige kald til de rette JDBC kald.

JPA - standard letvægts objekt orienteret persisterings framework som bygger oven på JDBC. Ideen er at man har en normal Java klasse og at man via annotationer beskriver hvordan felterne i klassen skal mappes til felter i en database tabel. Frameworket sørger så selv for at konvertere forskellige kald til de rette JDBC kald. JPA har erstattet CMP entity EJB. Hibernate understøtter også JPA.

ADO.NET - Microsoft .NET indeholder et helt nyt satabase API som er lidt en blanding af standard API'er som ODBC/JDBC og de database specikke API'er. Ved hjælp af standard objekt orienteret teknologi har man database specifikke klasser som implementerer standard interfaces. Folk kan så vælge at bruge de konkrete database specifikke klasser eller bruge interfacene. Microsoft leverer .NET providere for MS SQLServer og Oracle plus overbygninger til både OLE DB og ODBC ag hensyn til kompabilitet. Derudover leveres flere og flere andre databaser også med en .NET provider (mit indtryk er at .NET providere allerede er mere udbredt en OLE DB providere men stadig mindre udbredt end ODBC drivere).

Eksempel med C# kode SqlClient og MS SQLServer database med integrated security og datareader:

```
        SqlConnection con = new SqlConnection("server=MINPC;Integrated
Security=SSPI;database=MinDatabase");
        con.Open();
        SqlCommand sel = new SqlCommand("SELECT * FROM T1", con);
```

```
        SqlDataReader rdr = sel.ExecuteReader();
        while(rdr.Read()) {
            int f1 = (int)rdr[0];
            string f2 = (string)rdr[1];
            // f1 og f2 indeholder nu data
        }
        rdr.Close();
        con.Close();
```

NHibernate er en portering af Hibernate til .NET.

LINQ to SQL og LINQ to EF (Entity FrameWork) - letvægts objekt orienteret persisterings framework som bygger oven på ADO.NET. Ideen er at man har en normal C#/VB.NET klasse og at man via attributter / XML filer beskriver hvordan felterne i klassen skal mappes til felter i en database tabel. Frameworket sørger så selv for at konvertere forskellige kald til de rette ADO.NET kald. Kom med henholdsvis .NET 3.5 og .NET 3.5 SP1.

Kommentar af simonvalter d. 19. Aug 2004 | 1

Selv om oracle er kommerciel kan den hentes i fuld version og bruges under Developer License som siger noget i retning af at man må bruge deres produkter til at udvikle en prototype så der er ikke noget i vejen for at studere den lidt nærmere.. man skal bare være klar over at den bruger mange resourcer ;)

Kommentar af pajterma d. 09. Aug 2005 | 2

Kommentar af webcreator d. 16. Jan 2005 | 3

Rigtig god gennemgang - dog kunne du sagtens have gået mere i dybden med de forskellige databaser.

Kommentar af tigertool d. 24. Aug 2004 | 4

Rigtig god artikel.

Kommentar af optical d. 24. Aug 2004 | 5

hvad kan man sige - det er Arne_v kvalitet :)

Kommentar af alister_crowley d. 04. Apr 2005 | 6