InterBase im Internet

InterBase und Firebird in Konkurrenz mit MySQL

von Karsten Strobel

Wenn Sie beim Händler einen stromlinienförmigen Flitzer mit aufwändiger Speziallackierung und Designerfelgen sehen und daneben – zum gleichen Preis – ein eher kastenförmiges, solides Gefährt mit ausgeprägter passiver Sicherheit und langen Wartungsintervallen, dann ist es wohl Geschmackssache, welchem Angebot Sie den Vorzug geben. Wenn aber nur der Flitzer unter Scheinwerferlicht im Verkaufsraum steht und sein Konkurrent in einem verschlossenen Schuppen im Hinterhof, dann sind die Chancen nicht mehr ganz gerecht verteilt. Schnappen wird uns also eine Brechstange und drehen eine Proberunde.

Natürlich ist heute das Internet in der IT-Branche niemandem mehr fremd und kaum ein Programmierer kann sich seine Arbeit ohne die Möglichkeiten, im Web zu recherchieren, Software downzuloaden und Diskussionsforen zu benutzen, überhaupt noch vorstellen. Für Datenbankentwickler, die an klassische Client-Server-Architekturen gewöhnt sind bedeuten die immer häufiger nachgefragten Weblösungen mit Datenbankanbindung eine ganz neue Herausforderung. Lösungsansätze stehen gleich dutzendweise zur Verfügung und zwischen A wie ASP und Z wie Zope dürfte kaum noch ein Anfangsbuchstabe für ein Akronym einer neuen Web-Technologie frei sein. Eine große Gemeinsamkeit dieser vielen Möglichkeiten besteht darin, dass Sie alle ganz anders sind als die vertrauten Mechanismen aus der C/S-Welt.

Wenn Sie die allgegenwärtige Reklameflut der großen und kleinen Web-Hoster nach Angeboten mit Datenbankfunktionalität durchstöbern, stoßen Sie immer wieder auf die Kombination von Linux, Apache, MySQL und PHP, manchmal auch kurz mit „LAMP“ bezeichnet. Linux, der Shooting-Star der Open Source-Bewegung, der keiner Vorstellung bedarf, ist bei Internet Providern die verbreiteste Betriebssystem-Plattform. Das liegt nicht nur an dem besonders charmanten Umstand, dass für Open Source-Produkte keine Lizenzgebühren zu zahlen sind. Linux wurde und wird auch sehr intensiv gerade für die Anforderungen im Webserver-Markt entwickelt und mit Tools ausgestattet, weshalb hier – anders als im Desktop-Bereich – dieses System inzwischen eine dominante Rolle spielt. Was für Linux unter den Betriebssystemen hinsichtlich Verbreitung und Funktionalität gilt, zählt für Apache unter den Webservern mindestens genauso. MySQL ist der meistinstallierte Vertreter aus der Gattung der Open Source-Datenbanken und das trotz seiner spartanischen Funktionsausstattung. PHP schließlich ist eine in den letzten Jahren zu großer Bedeutung und technischer Reife gelangte serverseitige Skriptsprache für Internetanwendungen.

Das schwächste Glied in der LAMP-Kette ist MySQL. Wer die Ausstattung aktueller SQL-Server kennt und gewöhnt ist, kann sich vermutlich nur schwer damit abfinden, bei MySQL nicht nur auf Stored Procedures und Triggers verzichten zu müssen, sondern auch – je nach Version – keine oder eine nur sehr spärliche Transaktionskontrolle, Referenzielle Integrität und Isolationsmechanismen zur Verfügung zu haben. MySQL gilt daher als eine zwar schnelle und im Web mit Tools bestens unterstützte, aber für den sicheren Mehrbenutzerbetrieb nur sehr eingeschränkt geeignete Software.

Vor diesem Hintergrund drängt sich natürlich der Gedanke auf, das M aus der LAMP-Zauberformel zu entfernen und an diese Stelle eine andere für Linux verfügbare Open Source-Datenbank zu setzen, namentlich InterBase bzw. Firebird. Borland vollzieht mit InterBase bekanntlich den Rückzug aus dem Open Source-Sektor und die neuste Version (6.5) der Software ist nur noch als so genannte

„Certifed"-Version – sprich kommerziell und lizenzkostenpflichtig – verfügbar. Firebird stellt eine Abspaltung des ursprünglichen InterBase-Quellcodes dar und ist und bleibt Open Source. Es lässt sich viel diskutieren und spekulieren über die Zukunftsaussichten des Open Source-Zweiges und es liegt mir fern, grundsätzlich von der kommerziellen Variante abzuraten. Im Bereich der Web-Lösungen könnte, speziell in der oben beschriebenen Konfiguration, zukünftig aber durchaus Firebird das Rennen machen und sei es nur, weil in diesem Rahmen die Bereitschaft zur Aufwendung von Lizenzkosten notorisch gering ist.

Einzelhaus oder Mietskaserne

Wer bisher Firebird oder InterBase auf einem Webserver haben wollte, aber über keinen eigenen statischen Zugang ins Netz verfügte, musste sich schon einen dedizierten Server mieten bzw. kaufen und bei einem Provider unterstellen. Dieser Exklusivzugriff auf einen Rechner hat natürlich einen Preis, der weit oberhalb der gängigen Webspace-Angebote rangiert, bei denen der Provider nur einen begrenzten Festplatten- und Ressourcenanteil eines Hosts vermietet und sich daher viele Kunden einen Rechner teilen, einschließlich der Nutzung des Datenbankservers. Einzig die Firma IT-Reich hat kürzlich angekündigt, einen Probebetrieb mit Firebird im Webspace-Abo zu starten und im Erfolgsfall standardmäßig anzubieten. Die Firma sucht nach Pilotkunden für die Testphase (siehe Kasten).

Wenn Sie sich einen eigenen Server für Ihre Webanwendung leisten, dann sind Sie auch selbst verantwortlich für die Wahrung der Sicherheit vor dem Zugriff Unbefugter. Da praktisch jeder Host im Internet Angriffsziel von Hackern ist, hat dieser Punkt natürlich eine große Bedeutung und soll deshalb auch das Hauptthema dieses Artikels sein. Zunächst muss aber etwas Installationsarbeit erledigt werden. Wenn Sie, so wie ich, eine aktuelle Version von SuSE Linux verwenden, haben Sie bereits den Apache Webserver zur Verfügung. Bei vielen anderen Distributionen ist das auch der Fall. Ich möchte diesen Punkt daher ausklammern und als erledigt voraussetzen. Bei Firebird haben Sie die Qual der Wahl: Auf den Downloadseiten [1, 2] finden Sie die Linux-Version in der Classic- und in der Superserver-Architektur. Die „klassische" Variante arbeitet unter der Regie des Inet Daemons (einer Art Türsteher-Prozess für die TCP/IP-Dienste des Rechners) und wird bei Bedarf gestartet, sobald ein nicht lokaler Benutzer eine Datenbanksitzung beginnt, und zwar als separater Prozess für jeden Datenbankanwender. Der Superserver dagegen agiert als eigenständiger Daemon und arbeitet intern mit Threads. Ich entscheide mich für den Superserver und damit für die neuere Architektur, die sich mit etwas weniger Ressourcen bescheidet, allerdings mit dem Nachteil, dass im Falle eines Absturzes der Datenbanksoftware alle Sitzungen davon betroffen sind.

> **Tür und Tor für Angreifer schließen**

Firebirds sanfte Landung

Die Entscheidung zwischen den angebotenen Dateiformaten (*rpm* oder *tar.gz*) fällt leicht. *rpm* erlaubt eine stressfreie, automatisierte Installation mit einem Befehl; *tar.gz* ist ein Archiv, das manuell entpackt werden muss, um an die Programmdateien und Installationsskripte heranzukommen. Die zur Zeit aktuelle und als stabil bekannte Version von Firebird ist der Snapshot-Build Nummer 821, der volle Dateiname der favorisierten Version ist *Firebird-1.0.1.821-0-SS.i386.rpm*. Wie versprochen ist die Installation mit einem Einzeiler an der Konsole erledigt, wobei root-Rechte obligatorisch sind:

```
linux:~ # rpm –i Firebird-1.0.1.821-0-SS.i386.rpm
Starting Firebird  done
```

Firebird landet im Verzeichnis */opt/interbase*. Der Name dieses Pfades ist ein weiterer Beleg für die bekannte Weisheit, dass man seine Herkunft nicht verleugnen kann. Der Datenbankserver wird bei der Installation sofort gestartet. Wenn Ihr Rechner bereits am Internet hängt und der Zugriff auf den von Firebird verwendeten TCP-Port 3050 (symbolisch: *gds_db*) nicht verhindert wird, dann sind in diesem Augenblick einem Angreifer Tür und Tor geöffnet, die man schleunigst schließen sollte, um nicht ungebetenen Besuch zu erhalten.

Die allergrößte Sicherheitslücke ist natürlich immer ein unverschlossener Haupteingang. Bei Firebird bzw. InterBase ist darunter der für alle Datenbankaktionen berechtigte (Super-)Benutzer *SYSDBA* zu verstehen. Dieser hat nach der Installation das Default-Passwort *masterkey*, das natürlich unbedingt geändert werden sollte. Das muss erstens in der Benutzerdatenbank und zweitens im Startup-Skript geschehen, das nämlich zum Beenden der Datenbank das *SYSDBA*-Kennwort „kennen" muss. Um diese beiden Aufgaben zu erledigen, bietet Firebird unter */opt/interbase/bin/changeDBAPassword.sh* zwar ein Shell-Skript an, das aber leider – zumindest auf meinem Rechner – nicht funktioniert. Daher erledigt man die beiden Änderungen besser von Hand. Das *SYSDBA*-Kennwort wird wie gewohnt mit *gsec* geändert:

```
linux:~ # /opt/interbase/bin/gsec –user SYSDBA –pass
                                                masterkey
GSEC> modify SYSDBA –pw Mein_PW
GSEC> quit
```

Im zweiten Schritt müssen Sie mit einem Editor das Start-Skript */etc/rc.d/firebird* wie folgt ändern (suchen Sie die entsprechende Stelle):

InterBase/Firebird vom Internet Service Provider

Die Firma IT-Reich in Aichach – Systemhaus und Internet Hoster – startet einen Probebetrieb eines Web-Accounts mit InterBase oder Firebird als Datenbankengine. Die Programmierung mit PHP soll möglich sein.
Interessenten, die an einer Teilnahme an der Testphase interessiert sind, werden gebeten, sich zu melden. Nähere Informationen unter: www.it-reich.de/interbase/

```
: $(ISC_USER:=SYSDBA)
: $(ISC_PASSWORD:=Mein_PW)   #geändert,
                                bisher masterkey
```

Es versteht sich von selbst, dass diese Datei nur für privilegierte Benutzer sichtbar sein darf, was standardmäßig auch der Fall sein sollte. Die vor etwa einem Jahr in die Schlagzeilen geratene, vergessene Hintertür, die es bei InterBase und Firebird in Form einer fest kodierten Benutzer/Passwort-Kombination gab, ist in den aktuellen Versionen beider Anbieter beseitigt.

Das *firebird*-Skript wird vom Betriebssystem verwendet, um beim Booten bzw. Herunterfahren des Rechners den Datenbankprozess zu starten oder zu stoppen. Das kann man auch manuell machen:

```
linux:~# /etc/rc.d/firebird stop
linux:~# /etc/rc.d/firebird start
```

Damit die Datenbank beim Booten wirklich automatisch gestartet wird, muss in der Datei */etc/rc.d/rc.config* (gewöhnlich ganz am Ende) noch folgender Schalter von *no* in *yes* geändert werden:

```
START_FIREBIRD="yes"
```

Unter Beschuss

Warum ist es so wichtig, dass sich kein Unbefugter mit der Datenbank verbinden kann? Es liegt auf der Hand, dass Sie Ihre Daten vor Spionage und Manipulation durch unberechtigte SQL-Zugriffe schützen wollen. Aber leider bietet der Datenbankserver einem Hacker verschiedene Möglichkeiten, die noch weit darüber hinausgehen. Kritisch sind besonders Externe Funktionen (Funktionen einer Shared Library – bei Windows .DLL – die gelinkt und ausgeführt werden können) und Externen Tabellen, also Tabellen, die außerhalb der normalen Datenbankdatei für einzelne Tabellen angelegt werden können und deren Struktur sich beliebig bestimmen lässt. Besonders in ihrer Kombination sind diese beiden eigentlich sehr nützlichen Features der Albtraum eines Administrators. Nichts hindert nämlich einen fremden Benutzer, der über irgendeinen Datenbanklogin verfügt, eine neue Datenbank zu erzeugen, in dieser eine Externe Tabelle anzulegen und mit Inhalten zu füllen, die einer gültigen Shared Library entsprechen. Funktionen dieser auf das System geschmuggelten Bibliothek kann er dann als Externe Funktionen der Datenbank deklarieren und aufrufen. Auf diesem Wege lässt sich im Handumdrehen beliebiger Programmcode auf dem Server ausführen.

Um solche Horror-Szenarien zu verhindern, sollte man bei einem im Internet erreichbaren Server den externen Zugriff auf den TCP-Port der Datenbank am besten ganz unterbinden. Dafür gibt es verschiedene Möglichkeiten, von denen (bei SuSE) die Personal Firewall eine ist. Verboten werden sollten alle Zugriffe von fremden Rechnern auf den Port *gds_db*. Nur lokal und eventuell innerhalb des lokalen, abgesicherten Netzwerks darf dieser Zugang möglich sein. Wenn es um Sicherheit geht, gilt ganz allgemein, dass man nur als Profi diese Maßnahmen selbst durchführen sollte, weil die Fehlerquellen vielzählig und in Hackerkreisen bestens bekannt sind. Auf das Geheimhalten der Passwörter für *SYSDBA* und andere Datenbankbenutzer alleine sollte man sich auf keinen Fall verlassen, denn die Datenübertragung zwischen Client und Server wird nicht verschlüsselt. So ist es mit einem Sniffer-Programm und etwas krimineller Energie nicht allzu schwierig, diese Informationen auszuspähen.

Wenn aber doch der Fall eintritt, dass sich ein Eindringling Zugriff auf den Datenbankserver verschafft, sollten wenigstens die Konsequenzen gemildert werden. Nach der Standardinstallation läuft der Datenbankserver mit den Rechten des Benutzers *root*. In dem oben skizzierten Fall würde das auch bedeuten, dass eine illegal aktivierte UDF-Funktion mit uneingeschränkten Rechten auf dem System wüten könnte. Firebird braucht aber dieses Übermaß an Privilegien gar nicht, also ist es durchaus möglich, die Gefahrenquelle etwas einzudämmen und ein anderes Benutzerkonto zum Ausführen dieses Prozesses zu verwenden. Die Firebird-Entwickler haben für genau

Abb. 1: Erster Test gelungen

diesen Zweck unter */opt/interbase/bin/SSchangeRunUser.sh* ein Skript vorbereitet, aber leider funktioniert es unter Linux nicht einwandfrei. Ein angepasstes Scriptfile finden Sie im Web.

Beste Verbindungen

Nachdem wir nun etliche Löcher zugenagelt haben, stellt sich natürlich die Frage, wie denn ein Zugriff überhaupt möglich sein soll. Leider gibt es unter Linux weit weniger Administrationsoberflächen als unter Windows. Eine Ausnahme ist zwar Borlands IBConsole für Linux [3], aber die Kompatibilität mit Firebird ist eingeschränkt. Wenn ein sicherer Remote-Zugriff möglich ist, können Sie natürlich mit einer der vielen, zum Teil sehr ausgereiften Windows-Lösungen Ihren Linux-Server bearbeiten, etwa mit IBExpert [4], um nur ein Beispiel zu nennen. Zur Grundausstattung gehört natürlich auch das Konsolenprogramm isql. Das i steht dabei für interaktiv, wovon man sich aber nicht zuviel versprechen sollte. Immerhin können Sie in einen einfachen Dialog zu Ihrem Datenbankserver treten und Skripte ausführen:

```
linux:~ # cd /opt/interbase/bin
linux:/opt/interbase/bin # ./isql
Use CONNECT or CREATE DATABASE to specify a database
SQL> create database '/tmp/test.gdb' user 'SYSDBA' password
                                                    'Mein_PW';
SQL> create table T (t1 integer not null primary key, t2
                                                    varchar(20));
SQL> insert into T values (1, 'eins');
SQL> insert into T values (2, 'zwei');
SQL> commit;
SQL> quit;
```

Man kann auch eine existierende Datenbank als Backup-Datei auf den Server bringen, auch wenn Sie unter Windows erstellt wurde. Nach dem Restore (*gbak-create* ...) ist sie ohne jeden Portierungsaufwand wieder einsatzfähig, es sei denn, es werden besondere Externe Funktionen verwendet, die natürlich auf Linux angepasst werden müssen.

Das erklärte Ziel ist aber, die Datenbank mit einer PHP-Anwendung nutzen zu können. Auch PHP finden Sie in vielen – wenn nicht in allen – Linux-Distributionen ebenso wie den Apache Webserver bereits vor, aber selten in einer Form, in der das InterBase-Modul eingebunden wurde. Daher müssen Sie hier nochmals selbst Hand anlegen. Dokumentation und Tools zu PHP findet man reichlich im Internet und ich möchte hier nur einige wenige herausgreifen, die einen guten Startpunkt darstellen und von denen aus man sich weiterhangeln kann. Speziell für InterBase und PHP gibt es das deutsche Forum Open-Database [5] mit Artikeln, Links und Diskussionsforum zum Thema. Ebenfalls auf Deutsch sind das PHP-Center [6] und eine Portalseite für PHP-Anleitungen [7]. Downloads der jeweils aktuellen Version findet man natürlich auf der offiziellen PHP-Site [8].

Gleich zwei Artikel in den Foren beschreiben die Schritte für die Konfiguration einer InterBase-tauglichen PHP-Installation. Wichtig ist, beim Konfigurieren die Option *--with-interbase* mit anzugeben, um das entsprechende Erweiterungsmodul zu erzeugen. Ich gehe an dieser Stelle aus Platzgründen aber nicht weiter darauf ein und verweise auf die detaillierten Instruktionen im Web [9,10].

Sie können die InterBase-Funktionen Ihrer PHP-Installation mit einem einfachen Skript testen (Listing 1). Speichern Sie die Datei unter */usr/local/httpd/htdocs/test.php*, bzw. in dem für Ihren Webserver konfigurierten Dokumentenverzeichnis. Wie Sie sehen, ist das PHP-Skript in statischen HTML-Quellcode eingebettet. Die *ibase_*... Funktio-

Listing 1

```
<html>
<head>
 <title>Erster Test mit Firebird</title>
</head>

<body>

<h1>Inhalt der Tabelle T in der Testdatenbank</h1>

<?php
 $dbh = ibase_pconnect ("/tmp/test.gdb", "SYSDBA",
                                                "Mein_PW");
 $sth = ibase_query ($dbh, "SELECT * FROM T ORDER BY
                                                T1");

 echo „<table border=1><tr><th width=100>T1</th><th
                                    width=300>T2</th></tr>";

 while (list($t1, $t2) = ibase_fetch_row ($sth)) {
  echo "<tr><td>".$t1."</td><td>".$t2."</td></tr>";
 }

 echo "</table>";

 ibase_close ($dbh);
?>

</body>
</html>
```

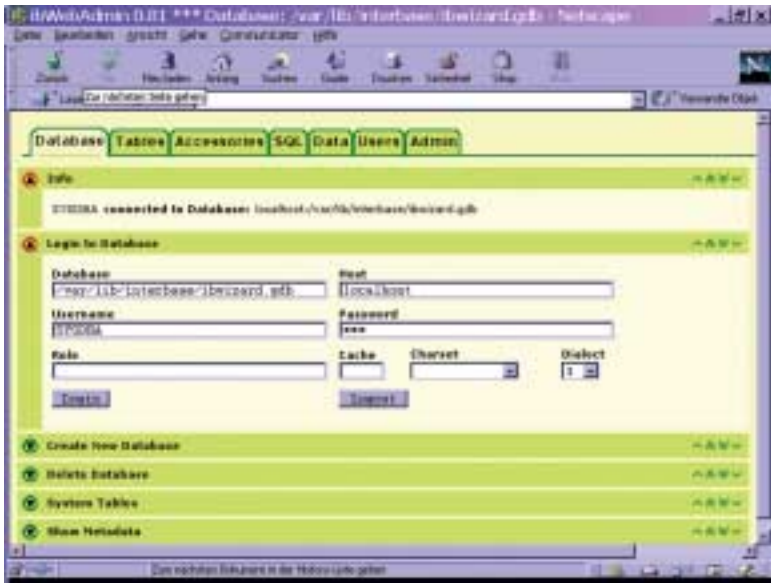

Abb. 2: ibWebAdmin

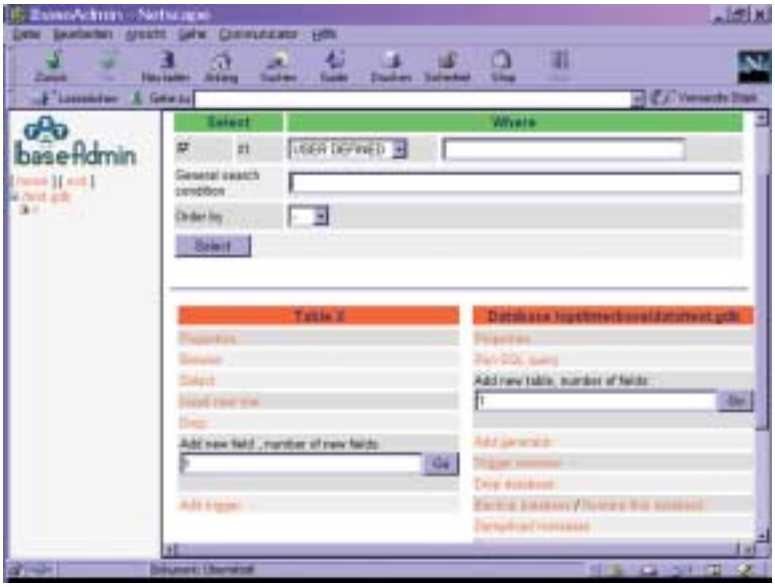

Abb. 3: IbaseAdmin

Firebird gibt es gleich zwei solcher Oberflächen, die ebenfalls in PHP geschrieben sind. ibWebAdmin (Abb. 2) und IbaseAdmin (Abb. 3). Beide Lösungen sind Open Source [13,14].

Beide haben den Anspruch, als webbasierte und universelle Managementoberfläche zu dienen und über den Browser Funktionen wie das Anlegen und Ändern von Tabellen, Durchsuchen der Daten, Starten von Abfragen, Löschen von Datenbanksätzen usw. zu erfüllen. Beide reichen leider bei weitem nicht an phpMyAdmin – das beim Open Source-Hoster Sourceforge.net in der Statistik der Entwickleraktivität an Platz 2 von über 43.000 Projekten geführt wird – heran.

Fazit

Obwohl Interbase/Firebird verglichen mit MySQL die ausgereiftere Datenbank ist, hat dieses Gespann in punkto Web-Tauglichkeit noch Schwächen, die weniger im Datenbankkern, als in der Ausstattung mit Tools zu sehen sind. Vermutlich ist das auch der Grund dafür, dass sich diese Lösung bisher bei Internet Service Providern nicht durchgesetzt hat.

Die Akzeptanz einer Datenbank hängt nun mal nicht nur allein von der Leistungsfähigkeit des Kernels, sondern ganz wesentlich auch von dem Komfort der Bedienoberflächen ab. Die laufenden Entwicklungen lassen aber hoffen, dass sich diese Situation bald verbessert.

nen sind im InterBase-Modul implementiert und im offiziellen PHP-Handbuch [11] beschrieben. Das Skript verbindet sich mit der mit isql angelegten *test.gdb*, greift auf die Tabelle *T* zu und gibt den Inhalt als HTML-Tabelle aus.

Wenn Sie nun die Seite mit einem Browser öffnen (Abb. 1) und sich den Quellcode anzeigen lassen, wird deutlich, dass die PHP Engine das Skript interpretiert und die programmierten Ausgaben an den Client gesendet hat. Das alles passiert auf dem Server. Der Client erhält auf seine Anfrage eine Datei, die sich rein formal nicht von einem statischen HTML-Dokument unterscheidet. Der PHP-Quellcode, einschließlich der darin kodierten Verbindungsparameter und Abfragen der Datenbank, ist für den Client unsichtbar.

Wer gerne ein etwas umfangreicheres Beispiel hätte, sieht sich am besten die mit PHP und InterBase geschriebene Mini-Adressverwaltung I-Man an, die als Open Source zur Verfügung steht [12].

Wer mit MySQL arbeitet, kennt höchstwahrscheinlich phpMyAdmin, eine Weboberfläche für die Verwaltung dieser Datenbank. Für InterBase und

Links & Literatur

[1] www.ibphoenix.com/ibp_download.html
[2] firebird.sourceforge.net/
[3] codecentral.borland.com/codecentral/ccweb.exe/author?authorid=102
[4] www.ibexpert.com/
[5] www.open-database.de/
[6] www.php-center.de/
[7] www.onlinetutorials.de/php-index.htm
[8] www.php.net/
[9] www.opendatabase.de/index.php?op=showdetail&id=6
[10] www.phpbuilder.com/columns/glodt20020212.php3
[11] www.php.net/manual/en/ref.ibase.php/
[12] freshmeat.net/projects/i-man/
[13] sourceforge.net/projects/ibwebadmin/
[14] freshmeat.net/projects/ibaseadmin/