

## Adatbázis-fejlesztés könnyedén, a Rekill segítségével

A Rekill segítségével rövid idő alatt fejleszthetünk ki Linux vagy Microsoft Windows alatt futó adatbázis-alkalmazásokat. Nézzük, következő munkánk során hogyan vehetjük hasznát ennek az ingyenes fejlesztőeszköznek.

**A** többféle géptípuson vagy operációs rendszeren is futtatható alkalmazások témaköre a következő néhány évben minden bizonnyal egyre nagyobb szerephez jut majd, ahogy a vállalatok – különösen a kis- és középméretűek – egyre nagyobb hányada ismeri fel a *Linuxra* való áttérés előnyeit. Ezek a vállalatok várhatóan fokozatosan, egyszerre mindig csak néhány géppel végrehajtani a váltást. Ha a piac vertikális lefedésére törekedve rövid idő alatt tudunk az ügyfél által használt alaprendszerek mindegyikén futó alkalmazásokat fejleszteni, akkor tanácsadóként fontos előnyre tehetünk szert. A *Rekill* és a *PostgreSQL* párosa itt és most biztosítja számunkra ezt a lehetőséget.

Tavaly a főbérőlm munkahelyéről felkértek, hogy segítsék lecserezni egy régi, Microsoft Access alapú alkalmazást. A hasonló megbízásokat korábban különféle eszközök segítségével teljesítettem, például *Glade*, *C++* és *Sybase ASE*, később pedig *Apache*, *PHP* és *PostgreSQL* alkalmazásával. Ez alkalommal többféle feltételt is teljesítenem kellett: a jelentéseket előállító programnak *Microsoft Windows* és *Linux* alatt egyaránt működnie kellett, illetve rendelkeznie kellett helyi „vastag” ügyféllel.

A háttérben futó adatbázis kiválasztásával nem sokat vacakoltam, a *PostgreSQL* mellett döntöttem. Az, hogy a *PostgreSQL* 8-as változata *Linux* és *Windows* alá egyaránt elérhető, nemcsak a szabad programok erejének nagyságát bizonyítja, de egyben kiváló alapot szolgáltat robusztus, többféle operációs rendszeren vagy géptípuson is futtatható alkalmazások készítéséhez. Megfelelő felülettel teljesen mindegy, hogy a kiszolgáló *Linux* vagy *Windows* alapú, vagy a munkaállomások mekkora hányadán fut *Linux* vagy éppen *Windows*. Mindez főként azoknak a vállalatoknak fontos, amelyek alkalmazottaik egy részénél vagy mindegyikénél az asztali munkaállomások *Linuxra* való átállítását tervezik.

Felületként olyan fejlesztői környezetet akartam, amellyel gyorsan meg lehet tervezni az űrlapokat, a jelentéseket és azokat az adatbázisokat, amelyekhez kapcsolódnak. A többféle környezet támogatása alapkövetelmény volt, ugyanis a megrendelő az

alkalmazottak egy részének számítógépét *Linuxra* akarta átállítani, míg a zárt, csak *Windowsra* elérhető alkalmazásokat futtató munkatársak továbbra is maradtak volna a *Windowsnál*.

Miközben keresgéltem, számos a *Microsoft Accesshez* hasonlítható, csak annál jobbnak nyilvánított alkalmazással találkoztam. Az eszközök mindegyike többféle adatbázis elérésére is képes volt, valamint támogatták az *ODBC* forrásokat is. Mindegyikük biztosított parancsfájlkészítési lehetőséget is, *BASIC*, *JavaScript* vagy *Python* nyelven. Néhány ezek közül az eszközök közül: *Kexi*, *OpenOffice.org Base*, *Kylix*, *Knoda*, *Rekill* és *Glom*.

A széles választék ellenére csak két eszközt nyilvánítottak termelésre késznek, a *Kylixot* és a *Rekillt*, így részletesebben csak ezekkel foglalkoztam. Miután rájöttem, hogy a *Kylix* önmagában nem támogatja a jelentések előállítását, egyedül a *Rekill* maradt.

### Ismerkedés a Rekillal

A *Rekillt* egy brit cég, a *Series One Consulting* fejleszti. *Mike Richardson*, a *Series One* vezető tanácsadója 2001-ben kezdte írni a *Rekillt* – egész egyszerűen zavarta, hogy nincsenek adatbázis-fejlesztői eszközök *Linux* alá.



© Kiskapu Kft. Minden jog fenntartva

Mike-hoz később csatlakozott John Dean is, aki a *Windows* és a *Macintosh* alatti fejlesztésért vállalta a felelősséget, illetve megírta az *Oracle* és a *DB2* illesztőprogramokat. A *Rekallt* eleinte a *TheKompany* terjesztette, mint keresztpatformos, kereskedelmi fejlesztőeszközeinek egyikét. 2003 végén a terjesztési szerződés lejárt, és a *Series One* úgy határozott, a továbbiakban maga terjeszti a *Rekallt*. Aki tehát a *Rekall* megismerése mellett dönt, az a *TotalRekall* vagy a *RekallRevealed* webhelyről töltsse le, ezek ugyanis a tevékenyen támogatott változatok. A *Rekall* kétféle, *GPL* és zárt szerződés hatálya alatt érhető el. Ennek oka az, hogy *Qt* alapú, és amikor első változatai megjelentek, a *TrollTech* még nem *GPL* hatálya alatt kínálta a *Qt* windowsos változatát. A *Rekall* linuxos változatát tehát forráskód formájában szabadon letölthetjük a *RekallRevealed* webhelyről, de dönthetünk úgy is, hogy fizetünk 25 fontot (körülbelül 9200 forintot), ekkor a linuxos, a windowsos és a *Macintosh* alá készült futtatható változatot egyaránt megkapjuk. A *Series One ODBC*, *Oracle* és *DB2* adatbázis-illesztőprogramokat és futási idejű csomagokat is kínál – felár ellenében. A *Rekall* következő kiadása a 2.4-es lesz, várhatóan több különböző változatban lesz elérhető. A *GPL*-es változat mindig forrás formájában lesz beszerezhető. A *Series One* úgy tervezi, hogy egy *Professional* változatot is megjelenít, mely két további szolgáltatást lesz képes biztosítani:

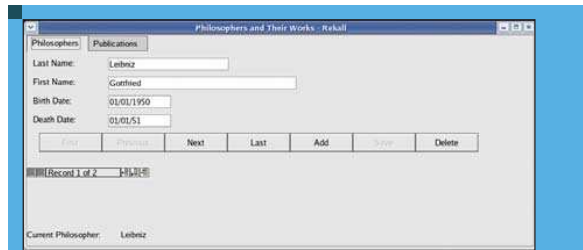
**Titkosítás** – Segítségével úgy terjeszthetjük alkalmazásainkat, hogy nem kell forráskódjuk lemásolásától tartanunk. A *Rekall Professional* változata által kínált titkosításnak köszönhetően alkalmazásainkat az ügyfelek számára személyre szabott védelemmel adhatjuk majd át.

**Webalkalmazások készítése** – Segítségével bármilyen *Rekall* alkalmazásból *LAMP* alapú webalkalmazást készíthetünk – lásd a *RekallRevealed* webhelyét.

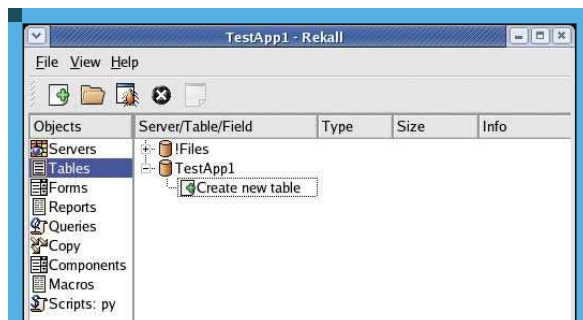
## A Rekall hátrányai

Használata során felfedeztem a *Rekall* néhány hiányosságát is. Először is, nem biztosít egyszerű eljárást menüsorok létrehozására, másodsor pedig az általa előállított alkalmazások nincsenek titkosítva.

A *Rekall*-alkalmazásokban található egy szabványos menüsor és eszköztár, amellyel a végfelhasználó – egyéb műveletek mellett – például lekérdezéseket indíthat el. Mivel szolgáltatásai meglehetősen széles körűek, ha korlátozni akarjuk a felhasználók számára elérhető lehetőségeket, akkor teljesen le kell tiltanunk a menüt és az eszköztárat. Aki nem fél az *XML* fájlok kézi szerkesztésétől, az korlátozni tudja a megjelenő gombok és menük körét, illetve sajátokat is létrehozhat. Ugyanakkor ez a használat egy nem támogatott módja, tehát jó volna, ha a szerzők a következő változatot kibővítenék ezzel a lehetőséggel. A kód teljes egésze, illetve az alkalmazásokban használt űrlapokat leíró *XML* szöveges formátumban tárolódik a fájlrendszerben vagy az adatbázisban. Ha tehát olyan alkalmazást fejlesztünk, amelytől valamilyen bevétel várunk, akkor érdemes megvárunk a titkosítási és webalkalmazás-készítési lehetőség megjelenését; esetleg megismernünk más megoldásokat, mint az alkalmazás



1. ábra A kész filozófusnyilvántartó program



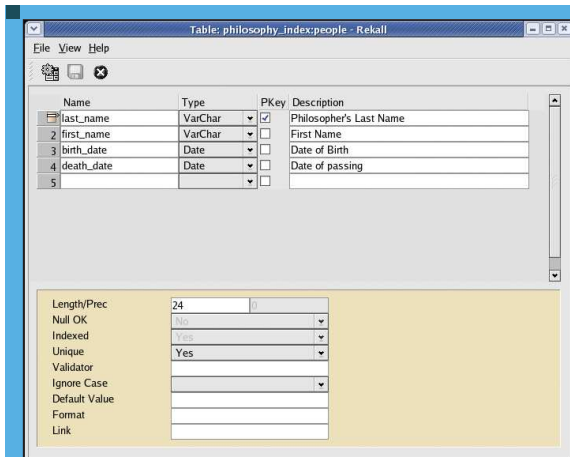
2. ábra A tervezet főablakáról lefelé lépkedve juthatunk el a táblák összeállításáig és az űrlapok és a jelentések létrehozásáig

szolgáltatásként való rendelkezésre bocsátását lehetővé tévő *FreeNX*. Egy az alkalmazással együtt terjesztett futási idejű könyvtárat megvásárolva korlátozhatjuk a végfelhasználók által végrehajtható műveletek körét, ugyanis a futási idejű könyvtárak nem tartalmazzák a fejlesztői eszközöket. Ezzel a megoldással azonban nem akadályozhatjuk meg a hozzáértőbb felhasználókat abban, hogy letöltsék a *Rekall* teljes változatát, kimásolják kódunkat az alkalmazásból, majd felhasználják saját céljaira, esetleg a fájlok módosításával saját szolgáltatásokat valósítsanak meg. A *Rekall Professional* változatának kiadása után az ügyfelek számára személyre szabottan, titkos kulcsra alapuló eljárással tudjuk majd elfedni az alkalmazások belső világát.

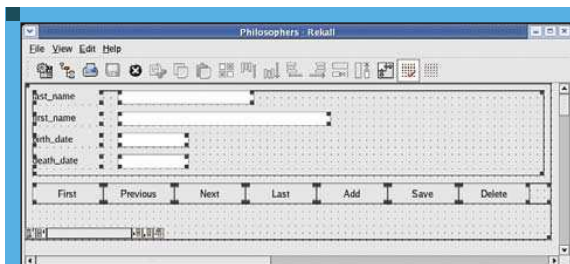
Természetesen annak is vannak előnyei, ha mindent nyílt *XML*-ben hagyunk. Nemrég észrevettem, hogy van egy sorozatnyi hibás beállításokkal ellátott összetevőcsoportom, más néven blokkom. Szerencsére semmi szükségem nem volt arra, hogy kivág-beilleszt módszerrel javítsam ki, esetleg újratervezzem az egyes blokkok mind a 11 mezőjét, ehelyett egyszerűen átírtam az *XML*-t, a blokkokat a lekérdezések helyett a megfelelő táblákra irányítva, és minden gond nélkül működött.

## A Rekall beszerzése

Miután egy ideig dolgoztam a *Rekall* forrásként letöltött linuxos változatával, a windowsos telepítőkészletként vásárolt változattal is kipróbáltam az alkalmazásomat. Az eredmény meggyőző volt, az alkalmazás mindkét környezetben azonos módon működött. Az átvittetés *Windows* alá gyerekjáték volt: egyszerűen készítettem egy biztonsági mentést a linuxos gépemem futó *PostgreSQL* adatbázisról, majd átmásoltam ezt a fájlt,



■ 3. ábra Új tábla létrehozása a Table ablakban



■ 4. ábra Az űrlapvarázsló a munka túlnyomó részét elvégzi helyettünk, mégis hatalmas rugalmasságot biztosít az űrlap tervezésében

valamint a *ReCALL* tervezetfájlját a windowsos gépre. Ott visszaállítottam a mentési fájlt a *PostgreSQL 8.0.2* windowsos változata alá, majd a *ReCALL* windowsos változata alatt futtattam a tervezetfájlt. A windowsos telepítés mérete nagyjából 7 MB. Ügyeljünk arra, hogy a *ReCALL* telepítése előtt a *Python* megfelelő változatát is telepítsük. A *ReCALL 2.2.3*, tehát a jelenlegi kiadás használatához a *Python* a 2.3-as sorozatba tartozó változatainak valamelyike szükséges. A *Linux* alatti lefordítás roppant egyszerű. *CentOS 3* alatt a fordítás és a telepítés egyaránt hiba nélkül zajlott le. *CentOS 4*-re frissítés után ugyan fordítási hibákat kaptam, ám az alkalmazás telepítése így is lezajlott. Viszonylag régi, 900 MHz-es *Athlon* processzorral és 512 MB memóriával el látott gépemen a fordítás körülbelül egy órát igényelt. Első futtatásakor a *ReCALL* jó néhány párbeszédpanelt jelenít meg, ezek segítségével a fejlesztői környezet különféle beállításait adhatjuk meg, ilyen például a rekordfrissítések és a törlések ellenőrzése, illetve a fejlesztői környezet elrendezése. A beállításokkal kezdő megadásuk után többé nem kell foglalkoznunk, hacsak le nem töröljük a *ReCALL* beállító fájlját.

### Példaalkalmazás

Szemléltetni szerettem volna a *ReCALL*ban végzett fejlesztői munka könnyűségét, ezért összeállítottam egy apró, filozófusok publikációinak nyilvántartására használható alkalmazást.

Segítségével mind a fejlesztés egyszerűségét, mind a *Python* alapú parancsfájlkészítési lehetőségeket be tudom mutatni. Célom az 1. ábrán láthatóhoz hasonló alkalmazás készítése volt, egy kisebb adatbeviteli ablak két lappal, alul pedig egy jelentés- és egy állapotsor, mely az éppen kiválasztott filozófust adja meg.

Első lépésként módosítanunk kell a *PostgreSQL* beállításait, hogy megfelelően együttműködjön a *ReCALL*al. Ehhez a helyi alkalmazások által indított kapcsolatok támogatására van szükség, amelynek engedélyezését az A *PostgreSQL* beállításai című szelvényzet foglalja össze. Ha ezzel végeztünk, adjuk hozzá a *philosophy\_major* felhasználót, adjunk neki jelszót, valamint hozzuk létre a *philosophers* adatbázist. Mindezeket a lépéseket még az előtt kell elvégeznünk, hogy a *ReCALL* először csatlakozzánk, ugyanis a *ReCALL* nem képes felhasználók és adatbázisok hozzáadására. A *PostgreSQL* kiszolgáló újratöltése után létrehozhatjuk az első kapcsolatot.

A következő teendőnk a tervezet létrehozása. Nyissunk meg egy terminálablakot, és hozzunk létre egy könyvtárat a tervezet számára, majd a *reka11* parancssal indítsuk el a *ReCALL*t. Telepítés után a *reCALL* futtatható fájljának a */usr/bin* könyvtárban kell lennie.

A tervezet létrehozására szolgáló varázsló első képernyőjén adjuk meg az adatok tárolására kiválasztott könyvtárat és a tervezet nevét, majd adjuk meg az űrlapok, a jelentések és a többi *XML* adatszerkezet tárolási helyét. Ezek az elemek egy különleges *ReCALL Objects* táblában az adatbázisban is tárolhatók, de fájlok formájában a fájlrendszerben is elhelyezhetők. Miután kiválasztottuk a tárolási helyet, adjuk meg, hogy milyen adatbázist használunk. Külön-külön illesztőprogramok segítségével több adatforrással is dolgozhatunk, de a most látható párbeszédpanel a fő adatbázisra vonatkozik.

A következő két párbeszédpanelen az adatbázist futtató állomást, a kapcsolatok fogadására használt kaput és a kapcsolatok létrehozásánál használt felhasználónevet és jelszót kell megadnunk. Ha a csatlakozás sikeres, kiválaszthatjuk, hogy melyik adatbázist kívánjuk használni.

Az adatbázis kiválasztása után a 2. ábrán láthatóhoz hasonló képernyő jelenik meg. Ezen a ponton megkezdhetjük az alkalmazás összeállítását. Ennek első lépése a táblák létrehozása, melynek elvégzéséhez kattintsunk az *Objects (Objektumok)* fa *Tables (Táblák)* elemére. Válasszuk ki a megfelelő kiszolgálót, majd kattintsunk duplán a *Create new table (Új tábla létrehozása)* parancsra. Megjelenik a 3. ábrán is látható táblaépítő.

A *ReCALL* a már meglévő táblákat nem tudja használatba venni – érdekes feladat volna egy adatbázissémákat *ReCALL* definíciós fájlakká alakító segédprogramot írni. Ha a *Tables* fában rákattintunk az egér jobb gombjával a tervezetünk nevére, akkor megjelenik egy menü, melyből módunk nyílik táblameghatározás beemelésére. A meghatározásnak *XML* fájlban kell lennie, ilyet viszont *SQL* táblameghatározásból hozhatunk létre.

Itt az ideje, hogy létrehozzunk néhány űrlapot a korábban megadott adatbázissémák alapján. Szerencsére a *ReCALL* erre is biztosít néhány könnyen használható eszközt. Ha rákattintunk az *Objects* fa *Forms (Űrlapok)* elemére, majd kibontjuk a tervezetünk után elnevezett

elemet, akkor varázsló segítségével vagy a nélkül hozhatunk létre űrlapot. Még ha lapokra osztott oldalakat tartalmazó, összetett alkalmazásokat készítettünk is, érdemes először mindegyik lapot külön űrlapként létrehozni, aztán másolni ezeket az objektumokat, majd beilleszteni őket az űrlap megfelelő blokkjaiba.

A 4. ábrán követhető, hogy a varázsló segítségével hogyan hoztam létre egy űrlapot a filozófusok táblája alapján. Az űrlapok felett széles körű ellenőrzést gyakorolhatunk, még a varázsló használatakor is. Nemcsak a használni kívánt táblát és mezőket választhatjuk ki, de megadhatunk oldalanként több rekordot, valamint meghatározhatjuk a mezőformátumokat és az oldalhoz önműködően hozzáadandó eszközöket is. A 4. ábrán látható képernyőn a varázsló a képernyő alján látható gombokat és a legalul megjelenő, apró navigációs eszközt adta hozzá. A *Nav. Toolnak (Navigációs eszköznék)* nevezett apró kiegészítő segítségével rekordról rekordra lépve navigálhatunk az adatbázisban.

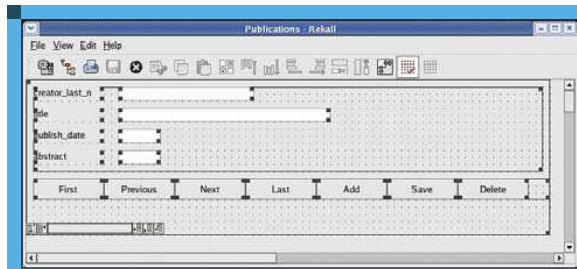
A *Philosophers (Filozófusok)* és a *Publications (Publikációk)* űrlapját létrehozásuk után (5. ábra) közös ablakba kell egyesítenünk. Itt mutatkozik meg az űrlapok kézi létrehozásának előnye, így ugyanis kívánságunk szerint tudjuk hozzáadni az összetevőket. Mielőtt rátérnénk erre, meg kell ismerkednünk a blokk fogalmával.

A *Rekall* űrlapjaiban az adatokat a blokkoknak nevezett egységek képviselik. Számos blokk típus létezik, a legegyszerűbb a tábla blokk, de van lekérdezés és *SQL* blokk is, utóbbiak egy adott lekérdezésből származó adatokat jelenítenek meg. Egy űrlap tetszőleges számú blokkot tartalmazhat. A blokkok a képernyőn gyakorlatilag tetszőleges elrendezésben megjeleníthetők, illetve noteszablakba, lapokra elosztva is elhelyezhetők. Mivel csak egy ablakot akartam, én is ilyen megoldást választottam.

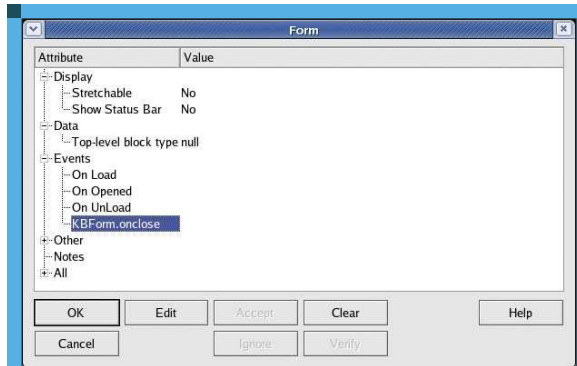
A noteszablak létrehozásához készítettem egy új űrlapot, de a varázsló nélkül. A parancs kiválasztása után a 6. ábrán látható *Form Attribute (Űrlap jellemzői)* ablak jelent meg. Mint látható, alapszintű ablakot választottam. Ezt nem lehet átméretezni, nincs állapotsora, valamint legfelsőbb szintű blokkjának típusa null – a kiválasztási párbeszédben ez a *menu (menü)* blokk nevet viseli. A legfelsőbb szintű blokk típusaként menüt, avagy nullt választva válik lehetővé, hogy a blokkokat és a vezérlőelemeket tetszőleges módon rendezzük el az űrlapon.

Rájuk kattintva menjünk végig az összes jellemzőn, tegyük meg a szükséges választásokat, illetve kattintsunk rá a párbeszéd alján található *Accept (Elfogadás)* gombra. Ezt követően a *Form Attribute (Űrlap jellemzői)* oldalhoz sokban hasonlító *Block attribute (Blokk jellemzői)* ablak jelenik meg. Mivel most egyszerű menü blokkról van szó, az alapértelmezett értékek túlnyomó részét nyugodtan elfogadhatjuk, a legtöbb érték ugyanis csak a tábla vagy a lekérdezés blokkokra érvényes. Ezzel egy üres lapot kapunk, amit a kívánt nagyságúra méretezhetünk, majd hozzáadhatjuk a lapozó vezérlőelemet.

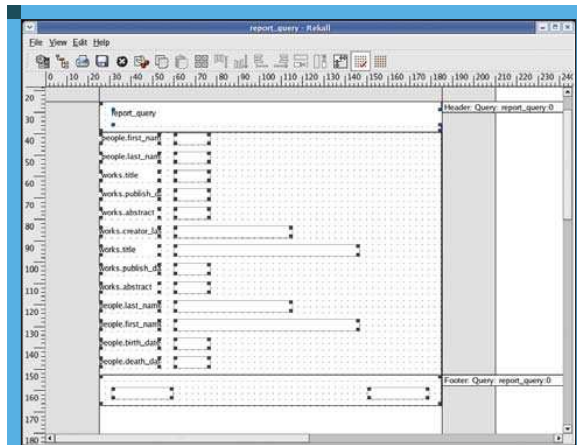
Mindegyik lapon létre kell hoznunk egy a megfelelő táblához tartozó tábla blokkot. Ez után másoljuk át a *Philosophers* űrlapon található blokk tartalmát a lapokra osztott *Philosophers* ablak megfelelő blokkjaiba. A *Publications* lappal ismételjük meg a műveletet. Ha ez is



5. ábra A publikációk űrlapja egy noteszablak része lesz, a 4. ábrán látható fő *Philosophers* űrlappal együtt



6. ábra *Rekall* űrlap létrehozása null típusú legfelsőbb szintű blokkal



7. ábra A jelentéskészítő ablakban a jelentések összeállítása legalább olyan egyszerű, mint az űrlapok megtervezése

megvan, nevezzük el a mezőket, módosítsuk a *Publications* oldal *Abstract (Kivonat)* mezőjének méretét – és végeztünk is, legalábbis ami a noteszablakot jelenti.

Most át kell váltanunk a *Philosophers* oldalra, és meg kell adnunk a publikációk kereséséhez használt kulcsot. Erre a fő űrlap alján, a lapozó vezérlőelem alatt látható állapotcímke használható. A címke szöveges értéke minden adatbázisbeli keresésnél vagy új bejegyzés létrehozásánál beállításra kerül. A beállításra akkor kerül sor, amikor az 1. kódrészletben található kódot a *Philosophers* blokk *On Display (Megjelenítéskor)* eseményével, visszahívással futtatjuk. A *Publications* blokk megjelenítésekor

## 1. kódrészlet Philosophers blokk:

```
def eventFunc (block, row) :
    someMainForm = block.getForm();
    currBlock = block;
    dataLabel =
someMainForm.getNamedCtrl("current_philosopher");
    dataLabel.setText(currBlock.getNamedCtrl
↳ ("last_name").getValue());
```

## 2. kódrészlet Publications blokk:

```
def eventFunc (block, row) :
    mainForm = block.getForm();
    currBlock = block;
    dataLabel = mainForm.getNamedCtrl
↳ ("current_philosopher");

currBlock.setUserFilter(dataLabel.getValue());
```

megtörténik az *On Display* esemény meghívása a blokkhoz, amely egy felhasználói szűrőt állít be a blokkban megjelenő adatokra. A felhasználói szűrő kódja a 2. kódrészletben található.

Végül kell valamilyen megoldás az adatbázisban szereplő filozófusok kilistázására. A 7. ábra a *Rekall* jelentéskészítő szolgáltatását szemlélteti.

A *Rekall* számos további összetevőt ismer, például jelentéseket, lekérdezéseket és adatmásolókat. Mindegyik összetevő hasonló könnyedséggel hozható létre, és ugyanolyan sokoldalúságot kínál, mint az úrlapok.

## Összefoglalás

Mint remélem, sikerült bemutatnom, a *Rekall* és a *PostgreSQL* párosával – *Linux* alatt – bármilyen adatbázis-programozási feladatot rövid idő alatt elvégezhetünk, miközben a számos tanácsadó által igényelt több rendszeren való működés képességét is biztosíthatjuk. Ahogy a vállalatok egyre nagyobb számban térnek át a *Linux* használatára a munkaállomásokon, a *Rekall*hoz hasonló termékek iránti igény nagyságrendekkel fog bővülni.

## A PostgreSQL beállítás

Ha alapértelmezett beállításokkal telepítjük, a *PostgreSQL 8.0.2* a felhasználókat linuxos azonosságuk ellenőrzésével hitelesíti. Ha biztonságosabb alkalmazást akarunk írni, állítsuk át az adatbázist jelszó alapú hitelesítésre. Ezt az alábbi lépéseket követve tehetjük meg. Először módosítuk a *postgres* adatbázis-felhasználó jelszavát, így akkor is be tudunk jelentkezni, ha jelszót kell megadnunk:

1. Írjuk be a parancssorba a `su` parancsot, majd adjuk meg a root jelszót.

2. Adjuk ki a `su postgres` parancsot.
3. A `psql template1` paranccsal indítsuk el a *psql* monitort.
4. A jelszót az `alter user postgres with password 'pgjelszo89'` paranccsal változtathatjuk meg, természetesen az általunk kívánt jelszót használva.
5. A monitorból a `\q` parancs kiadásával és az ENTER lenyomásával léphetünk ki.

Második teendőnk a *pg\_hba.conf* fájl módosítása, amelyet követően az adatbázis az *md5* jelszavakat is el fogja fogadni az összes kapcsolathoz. (Alapesetben az adatbázis-kezelő a hitelesítést az aktuális *Linux* fiók alapján végzi.) Alapértelmezett telepítésnél a fájl a `/var/lib/pgsql/data` könyvtárban található. Keressük meg benne az alábbiakhoz hasonló sorokat:

```
# "local" is for Unix domain socket connections
# only
local all all trust
# IPv4 local connections:
host all all 127.0.0.1/32 md5
# IPv6 local connections:
host all all ::1/128 md5
```

A jelszavak engedélyezéséhez a `local` kezdetű sorban a `trust` kulcsszót módosítsuk `md5`-re, majd mentjük el a fájlt. Indítsuk újra a *PostgreSQL*-t, a *Red Hat* jellegű rendszereken ezt a `/sbin/service postgresql reload` paranccsal tehetjük meg.

Ezt követően a felhasználókat és az adatbázisokat a *PostgreSQL* beépített eszközeivel vagy külső eszközökkel, például a *PgAdminIII* segítségével hozhatjuk létre. A *PostgreSQL* webhelyén mindezekkel a témakörökkel kapcsolatban kiváló leírásokat lehet találni.

*Linux Journal* 2005. július, 135. szám



**Joshua Bentham**

Hamarosan filozófiai bakkalaureátusi fokozatot fog szerezni az Ohio állambéli Bexleyben található Capital Universityn. A *Linux*ot az 1.2.8-as rendszermag megjelenése óta használja. Weblogja a `www.globalherald.net/jb` címen található, ő pedig a `jb42@globalherald.net` címen érhető el.

## KAPCSOLÓDÓ CÍMEK

Rekall: ➔ [www.totalrekall.co.uk](http://www.totalrekall.co.uk)

Rekall Revealed: ➔ [www.rekallrevealed.org](http://www.rekallrevealed.org)