Facts

- Your application has a data model
- Your application shares its data with the outside world

  - Typical way of doing this is using the filesystem

  - This has not changed too much since old UNIX days

Problems to solve

- You can have problems with performance for larger documents
- Importing and exporting data creates multiple versions of the same document
- Multiuser access to the data is not available or custom solutions are developed

Possible solution:

database storage / connectivity

- Many applications are already database-like
- Do not afraid of databases - you do not need to:

  - deal with internals (SQL, backend specifics)

  - deal with database servers (installation, administration)

    because stable embedded backend is available (SQLite) codename: KexiDB

What can you get

- support for SQLite files, MySQL and PostgreSQL
- database abstraction layer, connectivity library (dialogs, widgets), strong introspection
- cost of data opening and saving - independent of total size (plus network overhead in case of remote connections)

  (unlike the case with big XML files)
- strong data typing (a general feature of databases)

What can you get (2)

- builtin data import/export facilities:

  - CSV, FixedWidthText formats for tabular data

  - MS Access import (!)

  - exporting database from a file (SQLite) to PostgreSQL or MySQL servers (ODBC in development)

  - dialogs and GUI-less functions

- advanced database widgets

  - tabular and form views

What can you get (3)

- designer tools for tables, queries and forms
- simple printouts
- scripting bindings (using KROSS interface)
    - for Python, Ruby and Javascript – allows you to write extensions “in minutes”
- macros (ala MS Access)

Plans (2.0)

- making more functionality Kexi-independent
- more SQL features
- full text search

  (currently you can implement it internally at your GUI level)
- reports (with designer like in case of forms)
- multithreading – optimization for large data sets
- live data sharing ala (MS) Dynamic Data Exchange

  with notifications implemented using DBUS, think about Qt-only and/or non-KDE solutions for it

Extensions for your apps and the whole desktop

- it could be possible to write a connector providing KDE PIM data as a table(s) of data in real time
- provide live r/w data for KSpread (using DDE)
  or even use KSpread as a regular data source
- database storage as an alternative for the filesystem
  (reasonable when most documents are smaller than 2MB)