# Table of Contents

JVx has very useful database support based on DBStorage and DBAccess. The DBStorage is very powerful and supports automatic foreign, primary, and unique key detection. It reads allowed (check constraints) and default values automatically from your database table.

But sometimes the recommended usage is not enough for your requirements. We recommend to use following:

```
DBStorage contacts = new DBStorage();
contacts.setDBAccess(getDBAccess());
contacts.setWritebackTable("CONTACTS");
contacts.open();
```

This code is enough to have full CRUD support and automatic linked cell editor detection.

It's also possible to read from a view and write back into a table:

```
DBStorage players = new DBStorage();
players.setDBAccess(getDBAccess());
players.setFromClause("V_STAT_PLAYERS");
players.setWritebackTable("PLAYERS");
players.open();
```

If you want to add additional columns, it's better to use a view. But if you use a view, the automatic link cell editor detection won't work because of missing metadata information.

However, it's trivial to add missing metadata information:

```
players.createAutomaticLinkReference(
     new String[] {"EXT_PLAYER_ID", "EXT_PLAYER_NAME"},
     "EXTINFO",
     new String[] {"ID", "NAME"});
```

The statement will add a link cell editor for the columns EXT_PLAYER_ID and EXT_PLAYER_NAME (from the view). The link cell editor will read data from EXTINFO (table, view, etc.) and use ID and NAME column as display and selection values (JVx hides ID columns per default).

If you have a prepared storage for your link cell editor records, it's also possible to use the following:

```
public DBStorage getExtInfo()
{
    DBStorage extInfo = new DBStorage();
    extInfo.setDBAccess(getDBAccess());
    extInfo.setFromClause("EXTINFO");
    extInfo.setRestrictCondition(new Equals("TYPE", "U18"));

    //not relevant for this storage
    extInfo.setDefaultValue(false);
    extInfo.setAllowedValues(false);
    extInfo.setAutoLinkReference(false);

    extInfo.open();
```

```
    return extInfo;
}

...

players.createAutomaticLinkReference(
     new String[] {"EXT_PLAYER_ID", "EXT_PLAYER_NAME"},
     getExtInfo(),
     new String[] {"ID", "NAME"});
```

The above usage was still very trivial, and DBStorage supports custom statements like this:

```
DBStorage users = new DBStorage();
users.setQueryColumns(new String[] {"u.user_id", "u.user_name",
                                    "c.company_id", "c.company_name",
                                    "d.department_id",
"d.department_name"});
users.setFromClause("users u, companies c, departments d");
users.setWhereClause("u.dep_id = d.id and d.comp_id = c.id");
users.setDefaultSort(new SortDefinition("company_name", "department_name",
"user_name"));
users.open();
```

or this style:

```
DBStorage users = new DBStorage();
users.setFromClause("(u.user_id, u.user_name, c.company_id, c.company_name,
d.department_id, " +
                       "d.department_name "+
                    "from users u, companies c, departments d " +
                    "where u.dep_id = d.id and d.comp_id = c.id " +
                    "order by company_name, department_name, user_name)
users");
users.open();
```

The last syntax is nice to copy/paste a statement from your DB tool. The DBStorage executes:

```
SELECT * FROM <fromclause>
```