# Implement special fields as separate fields

## Context and Problem Statement

How to implement special fields in BibTeX databases?

## Considered Options

- Special fields as separate fields
- Special fields as keywords
- Special fields as values of a special field
- Special fields as sub-feature of groups

## Decision Outcome

Chosen option: “Special fields as separate fields”, because comes out best (see below).

## Pros and Cons of the Options

### Special fields as separate fields

Example:

```
priority = {prio1},
printed = {true},
readstatus = {true},
```

- Good, because groups are another view to fields
- Good, because a special field leads to a special rendering
- Good, because groups pull information from the main table
- Good, because hard-coding presets is easier than generic configuration
- Good, because direct inclusion in main table
- Good, because groups are shown with color bars in the main table
- Good, because there are no “hidden groups” in JabRef
- Good, because can be easily removed (e.g., by a formatter)
- Good, because prepares future power of JabRef to make field properties configurable

- Bad, because bloats BibTeX file
- Bad, because requires more writing when editing BibTeX manually by hand

### Special fields as keywords

Example:

```
keywords = {prio1, printed, read}
```

- Good, because does not bloat the BibTeX file. Typically, 50% of the lines are special fields
- Good, because the user can easily assign a special field. E.g, typing “, prio1” into keywords instead of “`\n priority = {prio1},`”
- Bad, because they need to be synchronized to fields (because otherwise, the maintable cannot render it)
- Bad, because keywords are related to the actual content
- Bad, because some users want to keep publisher keywords

### Special fields as values of a special field

Example:

```
jabrefspecial = {prio1, printed, red}
```

- Good, because typing effort
- Bad, because handling in table gets complicated → one field is now multiple columns

### Special fields as sub-feature of groups

- Good, because one concept rules them all
- Good, because groups already provide explicit handling of certain cases: groups based on keywords and groups based on author’s last names
- Bad, because main table implementation changes: Currently, each column in the main table represents a field. The main may mark entries belonging to certain groups, but does offer an explicit rendering of groups
- Bad, because groups are more a query on data of the entries instead of explicitly assigning entries to a group
- Bad, because explicit assignment and unassigment to a group is not supported by the main table

---