Design Principles for the *Accounting Flexfield*

Dr. Volker Thormählen, German Oracle User Group (DOAG e. V.)

1. Summary

This paper outlines the questions you need to ask in designing your *Accounting Flexfield* when implementing *Oracle Applications*.

2. Definitions

The following Oracle terminology is used:

The *Accounting Flexfield* is the full account number you use to identify a general ledger account in an *Oracle Applications* implementation. It comprises multiple *Segments,* such as *company*, *account*, and *department*. A *Value Set* is the list of values for one segment, such as the list of companies, the list of accounts, or the list of departments. *Segment Values* are numbers or codes within each segment, such as companies:= {01, 02, ... 99}, accounts:= {010100, 010500, ..., 990000}, or departments:= {1101, 1102, ..., 9999}.

An *Accounting Flexfield Segment* is either *independent* or *dependent.* An independent segment has meaning on its own. A dependent segment is one that is linked to an independent segment. *Account_subaccount* is the most common use for independent or dependent segments.

A *Summary Account* is a combination of all segments and is a full *Accounting Flexfield* code combination. Summary accounts are physically stored and updated with each journal entry or budget posting.

The *Parent (Child)* of a segment value N is the node directly above (below) node N in a tree-structured hierarchy over a *single Accounting Flexfield* segment. A *Leaf* is a node in the tree with no children. Therefore, a leaf always represents the posting level. The height of a hierarchy (e. g. the number of nodes on the longest path from the root to a leaf of the tree) is limited only by the segment size.

The *vertical structure* of the *Accounting Flexfield* is defined by the parent/child relationships set up over the segments and/or by summary accounts. Section 4 contains a detailed comparison of both methods.

The *horizontal structure* of an *Accounting Flexfield* is defined by the *name, number, size, type and logical order* of its segments. The same definition is valid for the *formal structure* of an *Accounting Flexfield*, if (and only if) the *physical location* and the logical order of the segments are equivalent.

Cross-validation Rules control which full *Accounting Flexfield* code combinations you can create automatically.

For illustrated definitions see [VT96], page 235-257.

3. General guidelines for the *set up* of the horizontal and vertical structure of the *Accounting Flexfield*

General guidelines for determination of *Accounting Flexfield* segments include identification of the dimensions of the business that may affect processing and on which you want to report, *especially across applications.*

- **Avoid redundancies.** If a detail for a business dimension resides in one information system or a set of highly integrated applications (for example, product revenue in AR) and is not needed for processing, retain the detail in the system(s) and do not include it in the *Accounting Flexfield*. On the other hand, if you capture detail product revenue, cost of sales, and overhead in multiple information systems that all pass journal entries to GL and you want to budget, allocate and analyse profitability by product, include the product dimension in the *Accounting Flexfield*.

- **Keep each business dimension as a separate segment**. Do not combine into one segment as this may complicate processing and reporting. Examples: Do not combine *company* and *division* into one segment. Do not combine *project* and *country* into one segment. Do not combine *location* and *department* into one segment, *location* followed by *department*, as it may be difficult to default department number and to retrieve data for the same department number across locations.

- **Avoid having more than one meaning for one segment**, for example a *generic subaccount* for *account*, *product* and *project.* Having more than one meaning may make it difficult to default segment values, to isolate different data for reporting, and to logically assign numbers or codes to segments. Combining multiple dimensions into one segment also precludes you from using more than one in an individual transaction, such as entering both *produc*t and *project* on the same transaction in the *generic subaccount* example.

- **Maintain an appropriate level of detail within a segment**. For example, if you include a *product* dimension, determine if *product line* would be preferable. If you include *product*, you will have more detailed transactions and resulting information, more code combinations, and more GL balances. On the other hand, if you include *product line*, you will have fewer detailed transactions, more summary financial results, fewer code combinations, and fewer GL balances. However if *product line* in captured and the *product/product line* relationship changes, transaction history using old product lines may not be useful.

- **Strive to have vertical structures above a single segment for reporting**, rather than enlarging the *horizontal structure* of the *Accounting Flexfield* or defining a structure above two segments combined. For example, set up accounts as parents and subaccounts as children within a dedicated account segment instead of separating account and subaccount into two segments. Obviously, having two segments will result in more complex cross-validation rules and more complex Flexfield population logic.

- **Avoid embedding horizontal structures in the *Accounting Flexfield*** such as *company_component_division_department*. Instead set up a vertical structure using parent/child relationships, such as the following *multiple* level hierarchy:

company	level 1: company is a parent
component	level 2: component is a parent
division	level 3: division is a parent
department	level 4: posting level: department is a child

Using parent/child relationships will be most flexible if the company reorganises frequently. You merely change relationships to reflect the new organisation and do not need to reclassify balances to restate accounting history.
Using parent/child relationships may, however, turn out to be bad for reporting. You cannot use parents to run most standard reports in GL. You would need to customise to generate such reports or to use a decision support tool such as *Business Objects*. On the other hand parents are favourable for FSG reporting and Mass Allocations. Therefore the benefits of parent/child often outweigh the disadvantages, especially if you want to utilise the extremely powerful *Mass Allocations* features of the GL module.

- **Ensure your *Accounting Flexfield* will support *Mass Allocations*.** For example, if you want to allocate overhead cost to departments (rolling-up into divisions) you will need a distinct *department* segment in the *Accounting Flexfield*. If you want to allocate based on products, you will need a district *product* segment. If you want to allocate based on *projects*, you will need a distinct project segment.

- **Ensure your *Accounting Flexfield* is segmented properly to default segment values.** Make sure that set up defaults and other built-in features such as

    - AutoAccounting (AR),
    - FlexBuilder (PO, FA) or its successor in release 11 (Workflow), and
    - Shorthand Aliases (GL)

  are able to populate the *Accounting Flexfield* for all expected business transactions. Otherwise you have to construct a customised population technique such as a *table-driven* approach.

- **Check whether localisations and/or statutory accounting are forcing you to define additional segments**. For example, a distinct segment for VAT processing was (and probably is still required) in some countries, for example Norway or Germany, see [VT98], page 183.

  A common goal for international companies implementing in multiple sites and multiple countries is *one* Chart of Accounts used world-wide. Such companies may find they need to capture different information in different sites and countries based in business needs and statutory requirements. A frequent approach is to maintain a common list of accounts in the account segment with site or country-specific detail in an independent subaccount segment that is controlled in each location and not consolidated to corporate headquarters.

  Unfortunately not all Oracle Applications modules support the subaccount approach. For example, avoid using any other segments besides company, account and department segments for *Fixed Assets* accounting.

  In addition you cannot roll-up *account_subaccount* information using parent/child relationships. This feature is limited to a single segment. If you need cross-segment roll-ups you have to utilise summary accounts instead. (see section 4 for more details).

- **Check your true need for dependent segments.** The advantages of dependent segments are as follows:

    - You do not need to *set up* cross-validation for independent/dependent combinations since you *set up* each combination. Sometimes setting up dependent segments is preferable to maintaining extensive cross-validation rules for combination of two segments for which there are many values and no ranges or other logic for valid code combinations.
    - Dependent segments limit the values that appear in corresponding picklist *pop-up* window.
    - If you have repetitive values and descriptions in the dependent segment, you will need to *set up* each combination.

  The disadvantages of dependent segments are as follows:

    - You cannot have multiple levels of cascading dependencies.
    - You cannot *set up* parents over dependent segments, since they have no meaning themselves.
    - Since consolidations are based on rules for single segments rather than combinations of segments, you cannot easily map different groups of dependent segment values into different consolidated values.

- **Place segments with defaults at the beginning or end of the *Accounting Flexfield***. When the *Accounting Flexfield pop-up* window opens, the cursor will be in the first blank (non-defaulted) segment. Once users complete all blank segments, they can press the save/commit key to deactivate the *pop-up* window. For example it is always better to order segments with *account* before *department*, if *department* is not always required. *Department* can be defaulted, for example to '0000', but can be changed after completing the account number.

4. Difference between Parents and Summary Acccounts

Summary accounts and parent/child relationships are means to *set up* the *vertical Accounting Flexfield* structure. How do they differ?

Criterion	**Summary Accounts**	**Parent/Child Relationship**
Direct posting of journal entries and budgets possible?	No	No
Physical storage and update with each journal entry and budget posting?	Yes	No
Requiring a full *Accounting Flexfield* combination?	Yes	No
Cross-segment *set up* possible?	Yes	No (1 segment only)
Is **T** (= Total) a valid segment value?	Yes	No
Usage in recurring journal formulas or budget formulas feasible?	Yes	No
Usage in mass allocation and mass budgeting formulas feasible?	Yes (for constant segments only)	Yes (for summing and looping segments only)
Usage in FSG row sets possible?	Yes	Yes
Usage in FSG column sets possible?	No	Yes
Usage in FSG content sets possible?	No	Yes
Online inquiry in GL possible?	Yes	No
Hint: You can assign segment types to *Accounting Flexfield* segments. You can choose one of the following types: *Looping, Summing, Constant.* See GL reference manual for more details. FSG:= **F**inancial **S**tatement **G**enerator		

Conclusions:

- You can view summary balances online in GL. You cannot inquire on parent account balances as balances do not exist for parents.

- Use summary accounts if needed for online inquiry of summary balances and if needed for allocations and reporting. Otherwise use parent/child relationships. Use parents if you frequently reorganise.

The author can be contracted at volker.thormaehlen@doag.org or
volker@dr-thormaehlen.de

About the author:

Dr. Thormählen headed the Applications-SIG of the German Oracle User Group (DOAG) in 1995 and 1996. He was a member of the DOAG-board during the last two years. He has published more than 20 articles on Oracle Applications. Most of them are written in German. All articles can be downloaded as pdf-files from the internet: http://www.dr-thormaehlen.de

Literature:

[VT96] THORMÄHLEN, V, Die Kontierungsleiste eines mehrdimensionalen Rechnungswesens in internationalen Konzernen, in: Betriebswirtschaftliches Controlling, Planung - Entscheidung - Organisation, Hersg. Bernd Rieper, Thomas Witte, Wolfgang Berens, Gabler Verlag, Wiesbaden 1996, ISBN 3-409-12909-X

[VT98] THORMÄHLEN, V. Pitfalls of Value Added Tax in Oracle Applications, in: Vortragsband zur 11. Jahrestagung der DOAG-Konferenz Fellbach 1998, Proceedings, Hersg.: DOAG e.V. Stuttgart, ISBN 3-928490-09-5