

The design of a *Survival OLAP*

(3-way TANGRAM's Technical Choices)

Ref. TG17E - Revision 1.1

This document is for the OLAP expert who wants to place the 3-way TANGRAM product in perspective and assess its applicability to an application.

1 Introduction

The OLAP market is vast and fast-growing, to the point that it is hard to justify still another product. The very high complexity of the problem at hand has been compounded with the variety and sophistication of the software tools of the Internet era.

The most sophisticated OLAP products require an irreversible company commitment, large investments, a development phase measured in many months, permanent consultancy, upgrades, training courses ...

If all this is successfully completed, the company eventually enjoys the benefits of an OLAP application which is supposed to be stable over time. The project has produced a nice showpiece that testifies the company commitment to glossy state-of-the-art software technology.

3-way TANGRAM is instead a *survival OLAP* and unique in its class.

It caters for individuals or workgroups that have to handle the complexity of OLAP problems with impossible deadlines, minimum budget, nearly no company support. They have to be able to get a copy of very large files, install their OLAP tool and be up and running in a matter of hours. *No ifs and no buts.*

Problem analysts, business analysts, data-hungry managers, external consultants are examples of typical users.

The design choices of 3-way TANGRAM must be understood in this scenario. Reliability, transparency, reaction to change are the top priorities.

2 Philosophy

- TANGRAM must understand as much as possible of the meaning of data, so as to shoulder a part of the user's responsibilities.
- Data labelling and nomenclature, for example, are mostly TANGRAM's responsibility.

- TANGRAM’s dialogue must be verbose and complete, to make sure that the user is aware at all times of what is happening.
- Emphasis is on mass operations, large cubes, complex data structures, high cube rank.
- The set of basic cube operations must be complete, with special emphasis on all typical needs in handling company data. This means that TANGRAM can be useful to handle a scientist’s or a statistician’s multi-way tables—but the company analyst has an extra guarantee that all his needs are catered for.

3 Data model

The basic data model is an N-dimensional numerical cube with the associated labels on each dimensions (for example, the names or codes of Measures, Customers, Products, Months).

Labels can be either free text, like “Cost of Goods” or multi-faceted text like “Marseille France Europe”. Facets are not formally declared to 3-way TANGRAM but simply pointed to when required.

This arrangement is very handy in most cases and in principle handles the full complexity of a relational table on each cube dimension. It shows its limits if you end up with too many long facets that you identify by inspection (not by field name) — in such cases you may want to redesign your labels in a more compact form or consider the use of a relational tool.

The number of supported cube dimensions is from 3 to 15. (Cubes of company data typically require much less than 15 dimensions!)

Some basic operations on cubes are meaningful on all dimensions (for example a roll up or a computation). Other operations are dimension-specific (for example the extrapolation of a time series applies to a time-dimension only).

This is solved by 3-way TANGRAM with the characterization of three specialized dimensions (hence the name 3-way) that stand for Measures, Time Periods, Items of some kind.

This situation (of a 3-way cube) is simple to visualize, display, print. With a 3-way cube it is understood that a time operation applies to the second dimension, (instead of declaring the nature of the axes of a 5- or 6-way cube and pointing to the axis that represents time).

If the Items dimension of a 3-way cube contains both Customer- and Product-codes you can regard these facets as two logical dimensions that share a physical cube dimension, possibly with a view to avoid sparsity. These facets may later become two dimensions of their own right, so that this arrangement loses nothing in generality.

All modules act on cube cells and labels in a concerted way.

4 Basic objects

One design goal has been to reduce the number of objects and concepts that a user has to cope with. The TANGRAM objects are:

- A cube (complete with data cells and labels). A database is simply a cube stored on file.
- A namesystem (a character table). Namesystems are reference labels that store element facets or aggregation trees and that are later used as labels of a cube axis.
- A procedure (a TANGRAM macro)

5 Outputs

TANGRAM's results can be delivered in three ways:

- via Excel
- via traditional framed printouts (sent to screen, printer, file)
- via export in source code format (XML, LaTeX, HTML, Acrobat, ...)

6 Missing values and Sparsity

A cube cell that contains no number is termed a Missing Value (alias, a Null). This could be due to sparse data, data entry in progress or a number of exceptions in cube manipulation (such as division by zero).

3-way TANGRAM's assumptions are:

- missing values must be handled in all phases: they must be distinguished from genuine values (for example, they are printed as blank) and must be propagated by all computations (for example, a result is missing where one of the operands is missing).
- the existence of a missing value in a cube is the only warning of the impossibility to calculate the cell. Mass operations on large cubes proceed without explicit warnings.
- the user may decide to force missing values to a genuine value (for example, to zero). See the keyword FORCE.

A cube with a high fraction of missing values is called a sparse cube.

Most OLAP systems devise sophisticated techniques for handling sparsity: basically this consists in software layers that maps a virtual sparse cube to a compact file structure that reserves disk space only for genuine values.

3-way TANGRAM's approach is to avoid the overhead of this sparsity-handling layers and treat all cubes as complete.

The real remedy for huge sparse cube is a cube design that reflects the actual relationship between dimensions.

If a single customer buys only a tiny fraction of existing products AND if this fact produces impossibly large cubes, you had better place the double customer-product key on a single cube dimension. In other words you place on cube axes element populations that are (nearly) in an all-to-all relationship.

Modules like CROSSOVER allow you to split double keys (like customer-product) to independent dimensions at a later stage (for example, when a rollup has reduced the cube to a manageable size).

A consequence of this choice is that the cube size (both in RAM memory and on disk) is always 8 times its cell count (since a floating point value requires 8 bytes).

7 Cube size - Virtual or real

The basic principle is that, up to a limit, the same processing must work irrespective of cube size. In other words, 3-way TANGRAM has little use for toy applications that prove correct on tiny cubes but are not scalable to full size.

The design goal is to handle the largest possible cube sizes for a given computer (for a given RAM memory and disk space). See a different document (SysLim.htm) for detailed system limits.

1. **Workspace-size** If your cube is small enough to fit into RAM memory (say, below 300 Mbytes on a typical PC) then 3-way TANGRAM allows for maximum flexibility: you can transpose, reorder, grow, restructure your cube and then save it to a physical database.

3-way TANGRAM uses a thumb rule to decide the maximum cube size that can be loaded into RAM memory with a reasonable expectation that further processing is successful.

If, however, a module fails for lack of RAM memory, the error is reported and the original cube is unaffected.

2. **Disk-size** 3-way TANGRAM also supports the processing of cubes that are much larger than the available RAM memory but reside on disk (up to about 2 Gbytes).

Administration modules achieve this by iterating the processing on the physical database.

For example, you can recompute a large database on file. Dimensions of disk-size databases can also grow in size, first to a maximum defined at database creation then by copying them to a larger disk space.

Analysis modules use a different technique for large cubes. The current cube is initially virtual, meaning that the labels are in RAM memory, the cells are not. A number of basic operations work on a virtual cube (typically SELECT and label operations) and reduce the current cube size

to a volume compatible with RAM memory. At this point the cube is read into memory and all modules are available. All subsequent operations act on the real cube, until a RESET command is issued.

Processing can thus be iterated on any cube dimension. For example, a roll up on dimension 3 can be performed iterating on a collection of elements on dimension 1 or 2.

8 Interoperability

The basic choice is that TANGRAM must handle the OLAP aspects only and interface with other applications of your choice for other aspects (for example: business graphics, data mining, statistical packages, ...)

The Dyalog system that plays host to TANGRAM offers a number of API (Application Program Interfaces) for the dynamic interaction with other software: **DCOM, ODBC, OLE, DDE, ActiveX, .NET, TCP/IP**

All these API are accessible to a developer with basic programming expertise.

TANGRAM, however, prefers interoperability via editable files: for instance, you may import/export a cube in XML format or as a flat file.

This choice has several advantages:

- The complexity of two software environments is not compounded. You don't need a server and a client to be running at the same time and you don't need to bet on their (idiosyncratic) interpretations of the API protocol. You may have no knowledge and control of all software components and layers that guarantee a connection across a network.
- You can process the exported file at a different time on a different computer.
- Problems are reproducible and traceable (e.g. by inspecting the exchange file).

TANGRAM's internal file format is documented and can also serve as an exchange format.

For example, a **TANGRAM reader** (an Excel add-on) makes TANGRAM cubes accessible from Excel so that Excel becomes a preferred front-end to deliver TANGRAM outputs or use Excel's own computation and styling features.

9 Programming architecture

A survival OLAP must cater for all needs, from those of a beginner to those of a fearless user and an application developer.

Increasingly complex requests must be handled with a more sophisticated use of existing tools.

This is achieved with general purpose modules and open-ended keyword languages.

The beginner uses a guided menu system and mouse choices, the demanding user can reply to each input request with keywords, the programmer can use the whole power of the underlying language (APL) in each reply (as opposed to writing stand-alone APL programs).

TANGRAM keywords constitute three cube-oriented languages for queries, computations and element selections (the “set language”). All these can be mixed with APL primitives.

The simplest way to build an application is to save a menu session as an editable procedure. Procedures can call other procedures (with unlimited nesting). They can run unattended or step-by-step, possibly with update of the original replies.

The expert application developer can choose his toolbox from:

- TANGRAM procedures
- front-end modules (with dialogue and validity check)
- back-end modules (without dialogue)
- GUI modules
- low-level utilities and APL
- Excel
- import/export to other data formats.

The basic support for teamwork is that conflicting requests for the same database do not produce Windows-level interrupts but are correctly enqueued and carried out when the database is freed. This allows for the unattended execution of time-consuming tasks.

10 Growth Path

OLAP systems are there to ad-lib. You certainly can't expect to have all answers a mouse-click away. You must think in terms of toolkits or open-ended languages.

Even if you find your OLAP just super, because you simply click options in a smart screen, you must be aware of what happens when you reach the ceiling of the canned solution.

(Alternatively, your OLAP *application* may be glossy and easy-to-use because your developers implement it from time to time. So *they* must worry about the growth path).

Possible growth paths:

- Learn more about the OLAP product. Read the fine print in the manual.

- Inquire if you can buy extra modules, add-ons, companion products that do what you need.

This is out of the question if you have to provide solutions within hours !
An *emergency OLAP* must entail a complete toolbox for all needs !

- See what the OLAP developer suggests for the very nasty, exceptional cases. The answer is normally to resort to some kind of programming (like SQL, OOP, Java, Visual Basic,...). But beware:
 - This means totally new concepts, new skills.
 - Progress is much slower.
 - Integration with the existing OLAP is a big issue.

TANGRAM's prescription is that *an interpreter mediate your answers to the OLAP product* and provide the necessary flexibility.

While this is a must, there are advantages in the choice of APL as the interpreter:

- The power of APL is way above all OLAP user needs (you will use an elementary subset).
- APL has an unsurpassed power/complexity ratio and emphasis is on N-way tables like hypercubes.
- Your ad-hoc solutions are seamlessly integrated with TANGRAM, since TANGRAM is written and embedded in APL.
- Most important, when you learn some APL basics, you are learning a smart general-purpose notation that you may use outside TANGRAM (capitalize on your learning effort).

11 Product scope

As a concluding remark, note that TANGRAM's ambition is mainly the *analysis layer* and that you may need complementary products to cover adjacent fields:

Cube structure, computation. Company specific applications.	Full support. Complete set of cube operations.
What if, simulation, prototyping, decision support.	Full support.
Reports, Reporting, Desk Top Publishing.	Well supported.
Business graphics.	Basic set included.
Time series analysis.	Basic set included.
Modelling (scientific, statistical).	Only linear models or do-it-yourself
Executive Information Systems (EIS).	Rough-and-ready support to build EIS logic.
Data mining (aka Knowledge discovery).	Marginally useful.
Data warehouse.	Marginally useful.
Management of non-business data cubes (scientific, econometric, statistical).	Perfectly viable.