DATA WAREHOUSING
GISG WORKING PARTY

1998 GENERAL INSURANCE CONVENTION
AND
ASTIN COLLOQUIUM

GLASGOW, SCOTLAND: 7-10 OCTOBER 1998

Data Warehousing

Working party

Richard Bland **Melanie Cooper**
Simon Harris **Roger Massey**
Geoff Perry

Summary

This paper is a discussion of data warehousing with an emphasis on actuarial and business approaches. The main section is an introduction to the concepts of warehousing and its business implications. The benefits and risks are discussed, along with the practical issues of structuring a warehousing project. Some guidelines for a successful warehousing project are suggested. To give some perspective, there is a short appendix with the history of one company's warehousing project.

A further three appendices are provided to give a full technical discussion of the IT issues in warehousing. These cover data models, some examples of these applied to retail and insurance, and a discussion of user tools. A number of warehouse projects fail because of inappropriate choices of data models and user tools. These appendices, although technical, are intended to provide insight into why these failures occur.

The views expressed in this paper are those of the working party as a whole and do not necessarily reflect the views of any one individual or any organisation with which any member has been associated. Whilst the working party has used its best endeavours to ensure accuracy, any person or organisation using this paper to make decisions should check the accuracy themselves and seek their own professional advice.

0. Introduction

The introduction of computers into many industries enabled the storage of large quantities of data. As both hardware and software have developed it has become less costly to analyse this data and to put it to constructive use. The number of uses to which data is put has grown markedly, which has led many companies to re-visit the manner in which their data is stored, retrieved and analysed. It was the users' need for up-to-date, accurate data which has instigated this business process re-engineering, and data warehousing has provided a possible solution.

1. Theory

1.1 What is a data warehouse?

Data warehouses have a variety of definitions, since they are often put to different uses, and therefore have different structures, and employ different methods, in different industries. Two such definitions are given below.

"A database designed specifically to provide data supporting decision making, as opposed to one used to run a business. Data comes from standard operational systems, but is checked, cleaned and reformatted to allow easier analysis by business users"

"A store of integrated information, available for queries and analysis. Data/information is extracted from heterogeneous sources as it is generated. By placing data from different sources in one place, it makes it easier and more efficient to query the data."

More generally, a data warehouse is a database which is regularly updated. It is fed from, but is separate from, a company's operational systems. The main purpose of the data warehouse is as a source of data for analysis purposes, and to enable the provision of management information. The warehouse may reside on a PC, a mid-range computer or a mainframe.

The data is made available to many levels of users, from inputting clerks to senior executives. Since data warehouses commonly hold vast quantities of data, with each item having many different parameters and levels within each parameter, users query the information held in the warehouse using software designed specifically for this task. These business tools can give quick and

accurate analysis, and can be constructed in such a way as to enable all users, whatever their level of computer literacy, to easily access the data.

Data warehouses, as well as helping to improve data systems, are often quoted as leading to tangible financial gains. For instance, Capital One Financial Services (a credit card company), claim that its warehouse strategy produced customer growth of more than 40% in 1995-97, compared with around 27% for its nearest rival. Certainly, in the case of retailers, data warehouses have allowed businesses to extend and improve the information they have on their customers' purchasing habits. This, in turn, allows them to maximise their revenues and profits.

However, there are many pitfalls in correctly establishing a data warehouse. Many options in terms of size and structure of warehouse are available and unless they are chosen correctly to match the business needs, the warehouse may not succeed. Recent surveys support this, showing that 60% of the 65% of large European companies adopting data warehousing strategies found that the project failed or did not meet their requirements.

1.2 Why do I need one, and why is it different to existing systems?

The main drivers for implementing a data warehouse are user demand, changes in the business environment and technology availability. Users' demand for more accessible, up-to-date information, companies' requirement to react more quickly to external changes and technology improvements have all contributed to making data warehousing a viable option for many companies.

Many individual reasons are put forward as to why companies should establish data warehouses: usually by those who are selling the concept. There are, however, many reasons which will be common to many industries.

- Inconsistent internal data systems

Many companies will have IT systems which have been developed piece-meal as the company has expanded. This can mean that different systems hold the same records in different ways or formats, making it difficult to integrate the different systems and provide an overall picture of the business. Similarly, users in different areas (using different systems) may produce similar reports showing different results. A data warehouse can provide an integrated, data-cleansed, consistent source of information that should avoid such problems.

- Legacy systems

As the millennium approaches, wholesale review of existing IT systems is occurring. This work often reveals that the existing (legacy) systems, sometimes developed tens of years ago, are either incorrect or are incomplete. Establishing a data warehouse gives developers a clean slate, allowing correction of such errors. Alternatively, the warehouse may be used to consolidate various legacy systems, ensuring that company-wide data sources are consistent.

- Over-reliance on IT support departments

Traditional IT systems produce outputs either on demand or regularly which usually require assistance from IT support staff, since the users who need the information to make business decisions do not have the technical knowledge required to run or maintain the requests. Data warehouses are constructed and accessed in such a way that users with little or no IT knowledge can easily and quickly obtain the information they require. Cutting out the IT support requirement should enable decision-makers to assess information and react to changes more speedily.

- Customer knowledge

Increasingly, companies in all sorts of areas are keen to extract as much information as possible about their existing customers, enabling the effect of future marketing or product launches, for example, to be maximised. Traditional IT systems are unlikely to be able to cope easily with this sort of data linking, but a data warehouse can easily cope with such data mining concepts.

Also, since data warehouses usually have a fairly open structure (in that their future development is not restricted), it may be easier to bolt on external data sources (for instance, market information or census data) to enhance customer knowledge.

1.3 Why might a warehouse be unsuitable?

A balanced argument should also include a discussion of areas or situations where a data warehousing approach is not suitable, or more specifically, the reasons why a data warehouse may not achieve maximum results.

- Poor set-up

Essentially, any data system is only as good as its design. In the case of data warehousing, the system is not usually designed to do one particular task, e.g. count stock or produce trial profit/loss accounts. Since it should permit wide-ranging queries of data, system design is potentially even more important. If the database is constructed in such a way that such queries are limited, the data warehouse will be of little or no use.

- Lack of education

Some organisations' culture will not suit the data warehousing system approach, since it relies on the users being more pro-active than in the traditional systems environment. For instance, some organisations do not encourage individuals to generate their own queries. This is one of the key ideas of data warehouse access. Individuals must be encouraged to generate their own queries, both ad-hoc and regularly. The use of OLAP tools and sufficient re-education means that users do not have to become IT experts, but they must move away from the culture of IT reports being delivered. If this does not happen or cannot happen due to the organisation's structure, then the best data warehouse in the world may become unused and end up as a costly white elephant.

- Impatience

Data warehousing is a young concept in IT, and therefore still has a relatively long development span. It will also require a substantial investment in maintenance, particularly if the system is taken up by users and usage increases significantly. Organisations must realise that this sort of system is not a short-term fix, but a permanent change in systems approach, which may need a lot of time and money expended before tangible benefits appear. Starting a data warehouse only to drop it a few years later would be a very expensive waste of money.

Note that many of these reasons are a consequence of inadequate or inappropriate design, as opposed to the data warehousing concept being of little use. Due to the high cost of developing such a system, it is vital that any organisation following the data warehousing path is clear in terms of its objectives, and in the way it is to achieve these objectives.

Developing a warehouse that is to be used as the data source for a comprehensive information system may be a long process, particularly if the company has little experience of the work involved. The delay in getting the

data to the users means that it may be difficult to maintain the momentum of the project, and to retain the continued co-operation of all the departments involved.

An alternative to attempting to build the complete warehouse in one stage is to develop a data mart which fulfils the needs of a particular department. If successful, this may help in maintaining other departments' interest in the project. The exercise may also be used as a learning exercise for the IT department. Additional data marts can be developed to meet the needs of other departments, and then the main data warehouse may be constructed by linking the information in each of the data marts. This "think big, start small" approach means that the project may also deliver business benefits in a much shorter time.

1.4 How is a warehouse different to existing systems?

Many managers, perhaps now convinced that a data warehouse is a worthwhile project, may not be convinced that they do not already have such a system in place. However, there are some key differences between a data warehouse and a traditional IT system that is used for decision support.

- Integration of data

Data warehouses are not process oriented, e.g. F&A claims file, Motor policy file, as in a traditional IT environment, but are subject oriented, e.g. customer, sales outlet, product etc. All data that comes from process-specific operational sources is integrated before entering the warehouse, so that queries can be easily performed.

- Time stamping

All data in the warehouse was current at some point in time, but many records are held which are now historical, unlike some operational systems which only hold current data. Practical differences are

- The data warehouse holds lots of old data: say 10 to 15 years, as opposed to the most recent snapshot in an operational database.
- The time-stamp of each record in a warehouse is a key element of its structure.
- Transaction data in the warehouse should not be updated: though it can be replaced by a more recent snapshot. Transactions themselves are not

overwritten or deleted, but many observations of their amounts, volumes or other quantities may be held as the warehouse is updated over time.

- Separate entity

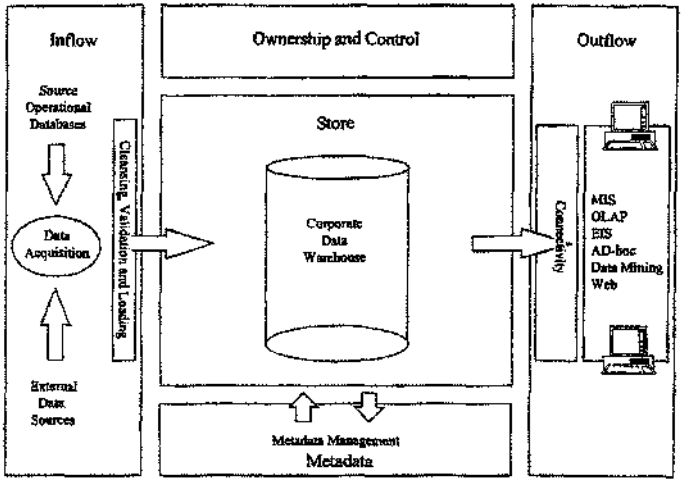
In all cases, a data warehouse is a completely separate entity to the systems that feed it, and usually the warehouse and its users cannot write back to the operational systems. Changing the operational systems should pose fewer problems. It should be possible to modify the structure of the warehouse to accommodate the new system. The data from the old system may then be re-coded to fit in with the new warehouse structure, where needed.

Since it is separate, the warehouse must be updated on a regular basis, unlike operational systems which will be real time. Provided the warehouse is updated frequently enough this does not cause problems and can actually be an advantage. Users requiring decision support can query the warehouse at will and are not restricted in size and time as may be the case with reports run against operational systems.

1.5 How is one built and used?

As mentioned in the previous section, the structure and design of a data warehouse is vital to its success. There are various ways in which data can be stored in a data warehouse, in terms of database style (relational, sequential) and data format (star, cascade), all of which are suitable for various purposes. Again, the key is to structure the warehouse and the data in it so that users can maximise their knowledge from the data, and so that false structures or relationships are not imposed on the data. These concepts are explored and explained in Appendix 2 (data models).

In terms of how a data warehouse as a whole should be constructed, the following diagram is a broad idea of a typical layout.



Existing operational systems provide inputs to the data warehouse. At the same time, external data sources may be used to enhance the quality of data or to provide additional data, e.g. census data. All of the relevant data should be copied from these sources and loaded to an intermediate database. At this stage, the data should be converted for use in the warehouse.

Data conversion is complex and time-consuming, but unless it is performed properly, the data warehouse will be of little or no use. Also, data conversion will be problematic since it usually has to be done in a short space of time or else the data available to decision-makers will be out of date. It is the one process most likely to reveal problems with the original source data.

Data conversion actually involves three different steps, specifically data cleansing and validation, data transformation and loading of the data. Data cleansing ensures the accuracy of the data, for example, making common sense checks on data values such as ages and birthdates. Errant data values can be corrected or excluded, and correction of the original source data can be considered. Data validation may include overall checks on data, e.g. policy counts, or the use of external data sources to verify data items produced from operational systems.

Once cleansed and validated, data must be transformed for use within the warehouse. Data from various operational sources will not always have the same identifying codes, even though they share the same characteristics. For example, gender may be coded numerically, e.g. 1 or 2, on one operational system, but alphabetically, e.g. M or F, on another. In the example of the Star data model (see Appendix 2), data transformation generates a consistent variable known as a key for each item of information from each source, meaning that summaries can be created within the warehouse by each key. This data key only exists within the data warehouse: it will usually be meaningless in the form in which it is stored within the warehouse. Such data, along with details on what all other items of data are, where data comes from and what can be done with it, are known as Metadata. The management of this data needs to be controlled to ensure that the warehouse operates correctly.

Usually data keys will be automatically generated by a dedicated application, and, due to the importance of key generation, tight specifications and procedures should be created that ensure the consistency of treatment of data items. The best way to achieve this is to involve users in the conversion specification process, ensuring that they are happy with the relationships implied between data from various operational sources.

This data cleansing, validation and transformation should be performed at an atomic level, i.e. for each record, a series of keys is created that indicates the various attributes of the record, known as dimensions. The data is then aggregated by dimensions, and loaded into the warehouse. The warehouse then holds all the inserted data, cleansed, validated and summarised by the pre-defined keys.

In some organisations, the data warehouse is then copied into smaller data marts, to reflect the fact that some users will only require access to a fairly limited amount of data, e.g. input clerks only need to access the customer data mart. Restricting their access only to the data that they require will speed up query times. Many data marts can co-exist within the warehouse.

In either case, end-users access the warehouse/mart through the use of OLAP tools, which are usually PC-based applications. MIS (Management Information Systems) may also access the warehouse, and Data Mining analysis can be performed on the warehouse data.

One key factor in the creation of a data warehouse is the speed with which the warehouse is refreshed. This is discussed in more detail in an insurance context in section 2.3.

1.6 Hardware and budgets

It is no surprise that data warehousing will, in most cases, require substantial storage capacity and processing power. A typical data warehouse may hold many gigabytes of data, meaning that the warehouse usually exists on either a mainframe or a mini-computer. If this cannot be achieved with existing hardware, the ideal solution would be to obtain the most advanced hardware possible. It should be technically capable of expansion at a later date since, if the warehouse is a success, the volume of data retained in it will increase over time. The warehouse hardware should be selected with consideration not only of technical issues, but also of practical factors, such as the location of input data sources and of users, any resulting strains on network systems that may occur, and, of course, cost.

In terms of overall budgets, purchase of hardware for a warehouse will not usually be the largest item, since storage capacity and processing power are now relatively cheap. W. H. Inmon (Information management: Charting the course) suggests that, for a typical organisation, budgets for a data warehouse will be split into one-off (i.e. set-up costs) and recurring costs. Set-up costs will be split into 60% on hardware and 40% on software (database management systems, access tools, systems management tools and data transformation/cleansing). Recurring costs will be split into 60% on data uploading (from operational systems), 6% on end-user education and 34% on warehouse administration/maintenance.

Recurring costs could be expected to grow by around 30% p.a., for a successful and frequently used warehouse, whilst hardware, if correctly selected, may only need replacement or enhancement periodically. However, it should be noted that advances in technology may quickly make warehouse hardware obsolete. One way to limit losses due to obsolete hardware could be to lease hardware rather than purchase it, allowing more frequent replacement.

Usually software and maintenance costs are likely to form the majority of expenditure, although this may be offset to some extent by the savings on old reports that are no longer required. Nevertheless, this re-emphasises the point

that a data warehouse is a medium to long-term commitment, and it will require significant investment for some time.

1.7 Control issues

Once the warehouse has been completed, an owner department must be assigned. It should control future developments of the warehouse, authorisation to use the warehouse, the access that different users are given, and ensure that sufficient maintenance is performed.

One of the good features of a warehouse is that it provides a single, consistent data set for all users. To be of use, this data must correspond to the data on the source systems. For example, if the management accounts of a subsidiary company are produced using periodic reports from its operational systems rather than the warehouse, the information should be verified by corresponding reports from the warehouse.

If a company actually uses a data warehouse as the source of information for its financial accounts, then the extract of the data from source systems, and any changes made to the data, should be covered by any external audit.

2. Practice

2.1 The insurance perspective

The insurance industry was one of the first large industries to adopt computer power in a major way. The need to process large volumes of policies and claims information, and to provide useful and timely information to managers led to widespread use of mainframe systems. These systems evolved piecemeal, limited by the technological constraints at the time, for example, using only 2 digits for years to save space. Many of these systems are still in place, basically unchanged from their inception.

The result is that, in many companies, outputs that influence managers' decisions are using inconsistent data definitions, and rely heavily on IT support, which typically means that changes to reports are costly and time-consuming. The reports cannot evolve quickly enough to match changes in the marketplace. At the same time, the need to provide frequent, accurate and relevant company information has increased, as the insurance market has become more competitive. In particular, in personal lines, the influx of new

insurers, with state-of-the-art IT systems, has put many well-established insurers at a competitive disadvantage. They are unable to analyse premium and claims statistics quickly enough to act against selection or business losses.

Ironically, most insurers do hold the information that they need to respond quickly to external changes: it is the manner in which the data is stored that makes such analysis difficult. Data warehousing may offer a means to solve this problem, and also to allow further analysis of customer behaviour, e.g. through use of external data sources, or analysis of effects of pricing strategies, that will allow insurers to maximise sales from existing customers.

This is certainly the conclusion that many insurers have already reached. Known users of data warehousing methods in the UK include Sun Alliance, Cornhill and GAN. Data warehousing is also in common use by US-based insurers. One example often quoted is of an unnamed Californian workers' compensation insurer which wanted to be able to respond quickly to problems using up-to-date information, as opposed to relying on monthly and quarterly reports.

The company wanted a system that could be accessed by any computer in the company and by all grades of staff, from non-actuarial, non-computer literate executives to senior business analysts. The system had to be able to give accurate and frequent information, with the ability to assess risk overall on a daily basis. Specifically, the system had to identify problem accounts, track individual claims, identify procedural problems and backlogs, monitor trends in losses with real-time frequency and alert potential problems to executives in minutes, not months. The company chose a product by Platinum Technology, called "Forest & Trees", to access a data warehouse specifically created for this purpose. The company states that it has benefited from the introduction of the warehouse through reduced operating costs, speedier processing of claims and increased the detection rate of loss control problems (leading to improved profitability for poorly performing lines of business).

A case study detailing one insurer's experience of a data warehousing project is given in Appendix 1 (warehousing experience).

Data warehousing is far more developed and accepted in the retail industry, where many applications of data warehousing are easier to envisage. However, there could be areas where the insurance industry could learn from

retailers' experience, and apply them through the use of data warehouses in insurance.

2.2 How to construct a business case for an insurer

Before a data warehouse project is embarked upon, it needs to be justified. There needs to be a demand for the data and the information that analysis of the data may provide. This demand is usually from the business, due to existing information systems not meeting the needs of users. The demand may also come from the IT department, wishing to improve its efficiency in supplying the business with information.

There may be gains from a successful data warehousing project, but actually putting a value on them before embarking on the project is extremely difficult. For example, some possible benefits might include:

- Consistent, cleaned data

Accurate and consistent data is vital to the assessment and pricing of risks, and often existing systems in an insurer will not automatically provide this. Constructing a data warehouse will provide a centralised, consistent source of data covering the whole business. Risk premiums can be priced more correctly, helping to remove the risk of selection.

- Year 2000

Many insurers are spending a lot of time and effort adjusting existing systems to be year 2000 compliant, often involving wholesale changes to reporting suites. This is an ideal opportunity to consider introducing a data warehouse, to provide more accurate and frequent information, although if you haven't started solving year 2000 by now it may be too late!

- Time and cost of changing IT reports

Existing weekly or monthly IT reports may have been created some time ago, and be difficult or costly and time-consuming to change. For example, a report producing weekly counts of in-force policies used to analyse the overall level of risk in the portfolio over time will need to be changed when extra rating factors are introduced. In a data warehouse environment, the only changes required would be to the application producing keys for each data item.

- Fraud detection

Fraudulent claims may be difficult to spot using existing operational databases, since it may not be easy to link between different systems relating to different time periods or classes of business. Such queries would be relatively easy to perform in a well-constructed data warehouse.

- IT savings

Existing management reports may need considerable IT assistance in their production, for example, in creating batch job submissions. In a data warehouse environment, there should be minimal involvement of IT staff in the production of reports used to support decisions. IT staff will instead concentrate on warehouse maintenance and uploads.

- Speedier reaction to changes

Similar to the above, decision-makers, once educated in the use of OLAP tools, will be able to create ad-hoc queries to address sudden changes and propose corrective action, without the need for another level of staff to be involved. Response times should improve, and the insurer should be able to react to adverse trends more swiftly, resulting in smoother and higher underwriting profits.

- Merged systems

Insurance conglomerates created as a result of past mergers may have several different operational systems in existence that all perform the same task, but in different ways and using different definitions, making it difficult to achieve an overall picture. A unifying data warehouse will make this much easier.

- Use of external data / Data Mining

Insurers are increasingly using external data, such as postcode verification software, census data by postcode and geographical information systems to improve their segmentation and analysis of risk. Often this is complicated by the need to combine data from several internal data systems that have inconsistent data fields, e.g. claims file, customer address file, policy file. Creating a data warehouse that holds all this data on a consistent basis will make using such external data easier. Data mining methods will also be easier to use, since the data will be in a useable format.

Given the fairly unquantifiable nature of these benefits, any cost-benefit analysis of such a project is very difficult. In practice, such an analysis rarely takes place, or one particular benefit is focused upon and used to justify the

project. For example, a 1% improvement in underwriting performance as a result of better information available to the underwriters may more than cover the cost of the project. Any additional benefits may be considered as perks of the warehouse. The costs which need to be taken into account should include the costs of the hardware and software required, training costs, consultancy fees and the cost of human resources.

2.3 Practical issues

Once sponsorship for the project has been received, ownership needs to be assigned. Since the demand for the warehouse usually comes from the business, the project owner is usually from the business and not the IT department. Actuaries are ideally placed to manage such projects, since they are heavy-duty users of data, and should be able to communicate users' requirements to the IT department effectively.

Any data warehousing project will require input from many departments. The responsibilities of these departments need to be clearly defined, and resources need to be committed to the project.

Before the project received sponsorship, an outline plan of the warehouse and what it will provide the business with will have been considered. Once allocated a budget, a detailed plan needs to be put together. One of the first steps is to clearly define the data that should be stored in the warehouse. Each department that will be a user of the data needs to outline its requirements. When defining requirements it is useful for all users to be familiar with any inherent coding structures within the operational systems. Issuing a set of corporate definitions for terms such as gross premium written, revenue period, accident year etc., with which users can specify their requirement, will help prevent misinterpretation of the users' needs by the IT department.

The users' requirements will determine a great deal about the warehouse, including:

- What data is stored in the warehouse (summary level or detailed information)
- How much data is stored (how many years' history)
- How the warehouse is structured
- How regularly the warehouse is updated
- How the warehouse is updated

2.3.1 What data is stored in the warehouse?

At the most detailed level, the data warehouse would contain information relating to every single transaction which has taken place. This may be more detailed than is required by many users, but would enable more detailed analysis than if summary data was stored. Using summarised data will however speed up the processing times when the data warehouse is accessed.

If data is required at transaction level by some users, but the majority only require summarised data, it would be possible to have the main (transaction level) warehouse pass the summarised information into a separate data mart, which users may access for the less detailed requirements. A good example of this may be the claims department only requiring claims information and basic policy information. They would be unlikely to be interested in rating factor information.

2.3.2 How the data is structured

As discussed in Appendix 2 (data models), there are various types of database and data models that are suitable in various circumstances. Unfortunately insurance data is not organised in the way that is suitable for many data model structures: for example, there is no obvious hierarchical structure to insurance data, unlike retailers' sales data. Many data warehousing products and solution providers would suggest a Star as the default model structure for a data warehouse, although this is unlikely to work well with insurance data, as described in appendix 3 (data model examples). The key is to know in advance the range of data to be held in the warehouse, the interrelationships between different items and the sorts of queries that will be asked of the data. Once you have all this information, it is likely that the choice of data structure will be more obvious.

2.3.3 How the warehouse is updated

Users will have different requirements for the frequency with which the data in the warehouse is refreshed. For example, management accounts will probably not be required more often than monthly. Underwriting departments are not likely to require information to be updated more than monthly, in order to monitor the portfolio mix of business, or the performance of accounts.

Administration and processing departments may wish to monitor volumes of business processed on a weekly or even daily basis.

There are various methods by which the data in the warehouse may be updated. The larger the volumes of data involved, the more important it is to choose the most efficient way of transferring data from operational systems to the warehouse.

If the warehouse were small, it would be possible to perform a complete download of the information each time the warehouse is updated, overwriting the existing data in the warehouse. Alternatively, it may be more efficient to append the information relating to transactions which have taken place since the last update, onto the existing information in the warehouse.

It is during the transfer process, when the warehouse is being updated, that the data is usually checked, cleansed and modified in any way necessary.

2.4 How to ensure that the project succeeds

1. Ensure that the users' requirements are clearly stated before the project gets under way.

2. Make sure that the IT department and the business both understand the aims of the warehouse, and work together towards a common goal.

3. Make it as easy as possible for users to use the warehouse. This may involve using a software package with which the users are already familiar.

4. Ensure that the structure is flexible enough to adapt to changes.

5. Adopt a "start small, think big" policy, or develop a prototype before embarking on the main project.

6. Ensure that controls are in place to make sure that the information in the warehouse is accurate, maintained, and that data used as the basis of reports is protected and cannot be overwritten by users.

3. Conclusion

Actuarial training and guidance state that a lot of time should be spent thinking about data: in particular, getting enough data to calculate premiums and making sure that this data is correct. Actuaries were involved initially in designing the early IT systems, and actuaries, along with other users, are commonly involved in the formulation of new IT systems in insurance companies. The introduction of a data warehouse is one way of changing systems to provide clean, consistent and accurate data company-wide, and so actuaries at all levels should be aware of the potential benefits of this approach.

Appendix 1 – Warehousing Experience (Company A)

Company A is a direct personal lines insurer with around ¼m policies running back over more than 5 years.

1. History of the project

The company's warehousing project began in 1995. The business principles were:

(a) It would be justified on the basis of the improved underwriting performance resulting from the actuarial analysis made possible by the warehouse.
(b) To support this business case there would be a pilot programme based on quotation data: assessments of the benefit would be based on benefits arising from the pilot.

Quotation data was chosen for the pilot on the basis that this was the area least well covered by existing actuarial reports. This pilot was developed rapidly (in the space of 1-2 months) and was delivered in April 1995. It was a hybrid data model: mostly flat with several minor lookup tables in an Oracle relational database. For analysis work this was converted to a fully flat data model in SAS. The presence of two copies of the data was inefficient from a storage point of view, but it was easy to maintain the Oracle copy because it was derived from an Oracle production environment; this was too slow for analysis work, so the SAS copy was necessary for a useful working environment.

Although never planned, this pattern of dual copies in Oracle and SAS was to be carried throughout the project, for similar operational reasons.

The pilot data contained postcode information which, while not sufficient for a full rating exercise, did suggest that there were enough geographical rating anomalies for there to be a significant benefit from rezoning of postcodes, for which full policy data would be required.

The full warehousing project began in September 1995 with a launch workshop which defined the contents and structure of the policy warehouse. Consultants from company X, who have warehousing experience, led the project. However, when specifying the project, three basic assumptions were made: (1) Hardware from company X would be used, (2) data would be held

in an Oracle database, and (3) a star data model would be used. The consultants did not explain that this last assumption was being made, and would appear to have made it before having any knowledge of the data concerned. It turned out that all three assumptions were interdependent and were all flawed because the star model was inappropriate.

The star model for policy data was constructed and tested over a period of three months and became operational in December 1995. A full worked example of policy data derived from a similar operational system, used in both star and cascade models, is shown in Appendix C, in which the problems with using a star model for insurance data are shown. In this case, despite some desperate measures to avoid proliferation of dimension tables, the model still contained more than twelve dimension tables, all to be joined to the central fact table, and all the joins were made using artificially generated keys. The consequences of this were:

(a) Actuarial queries using all available data took some hours to resolve.
(b) Fully loading the warehouse took more than two weeks. Most of this time was required to generate the artificial keys.

Despite these problems this warehouse remained in use. However, for provision of claims data, an alternative model was used. The consultants were not retained and further development continued in-house. The new approach was:

(a) An Oracle copy of production data was created in the warehouse (because this had the same table structure as the production system, it could be easily maintained).
(b) This was copied into SAS and converted into a cascade model. The cascade model was used because (i) the data contained natural cascades, (ii) this provided the most efficient storage, and (iii) it provided optimum performance when used with a sequential database: a query performance tens of times quicker than queries on the original production format.

This was delivered in April 1996 and was found to be wholly satisfactory in terms of performance.

Some time after this, toward the end of 1996, it became increasingly impossible to load the policy section of the warehouse, due primarily to its bad design, and it was replaced with a model similar to the claims model: an

Oracle copy of production, copied into a cascade model in SAS. This finally proved to be capable of efficient maintenance.

2. Conclusions drawn

The project finally delivered a usable and efficient warehouse, but mistakes were made: the original policy data model was so badly designed that it had to be scrapped and rebuilt. Much time and effort was lost in the creation and attempted maintenance of the wrong design. The best contributions to the project were made by those developers wholly familiar with the business and the design of the production system. The contributions of the consultants were almost entirely negative. The conclusion from this is that familiarity with the data environment is absolutely essential.

The wrong data model was used to begin with: this alone created an unmaintainable warehouse. It was not easy to use either. Moreover, this then dictated the use of a relational database: eventually a sequential database was used primarily for warehouse query work. The choice of database also dictated the choice of hardware, although this was also forced by the use of consultants linked to a hardware vendor.

This choice of hardware was significant, because it was an expensive part of the project and a major investment. Unfortunately, in the time which has elapsed from the purchase of this machine (more than two years) it has been substantially surpassed in performance by much cheaper hardware. This now puts significant limits on the use of the warehouse. It should have been assumed in the costing of the project that no hardware would last more than two years in service: as a result, only standard, easily available hardware should have been used.

The cost of the project was approximately £½m. This was divided roughly half into hardware costs and half into consultancy fees and internal programmer resource. Software was not a significant cost. The cost of the project could have been cut to less than £100k if (a) cheaper, more effective hardware had been used and (b) the consultants had not been employed and the successful in-house design had been implemented directly.

The business benefits of the project, though largely unquantifiable, can be thought of as having been achieved in the sense that all the intended actuarial work became possible, along with many other projects.

Appendix 2 - Data models

1. Databases

1.1 Relational databases

Before discussing data models, the types of databases must be considered. There are essentially two types of database: relational and sequential.

Relational databases include most of those used for transaction processing, e.g. Oracle, Informix. MS Access has some of the characteristics of a relational database, but is not truly relational.

Relational databases store data in tables in databases (the storage area may be compartmentalised into tablespaces). Although the table is treated within the database as a single entity, it is not stored as one within the database. The columns in the table, each representing a field such as "car group" or "driver age" are stored as separate entities with links between them which tie all the columns together to form a table.

Indexes are a vital feature of relational databases: these take the values of a column and arrange them in a sort order which enables a search algorithm to find a particular value quickly. For example, one of the simplest index algorithms is binary chop:

Take 1,000 records
Arrange them in order
Test the 500th entry: above or below the desired value?
Move to the 250th or 750th, etc.

It can be seen that with 1000 records, the binary chop will find the specified record within at most 10 search attempts: with a million records, no more than 20 attempts will be required. More sophisticated index algorithms are used which can deliver better search performance depending on the distribution of values within a column.

The presence of an index can enable a specified value to be found directly rather than by searching linearly through the column. This is particularly important where two tables are to be joined using a common column as the key.

The characteristics of relational databases arising from this construction are:

- Where operations are required on one record in a very large database, seek times can be extremely fast (this is a vital requirement for a transaction system).
- Retrieving one field from a table is much quicker than retrieving all the fields.
- Adding records or columns afterwards is not difficult.
- Joining two tables together does not slow down a query much compared to accessing the table alone.
- The speed of the join depends on how many fields and records the tables have, e.g. joining a small lookup table onto a big data table will not be much slower than a query on the data table alone.
- Indexing fields used in a join will dramatically speed up the process, e.g. minutes rather than hours.
- Because the table may consist of many bits and pieces spread out over a database, there is no scope for compression.

1.2 Sequential databases

Sequential databases are primarily used in archive and warehousing applications. SAS is the most common sequential database: many bespoke databases and archives use a sequential form.

In a sequential database, the table is a specific entity which is often held as an ordinary file within a file system, similar to a spreadsheet or a document. The fields are held in record order, i.e.

Record1 Field1
Record1 Field2
Record1 Field3
Record2 Field1
Record2 Field2

The simplicity of the storage system means that data recovery time can be very fast indeed, many times faster than a relational database, provided that all the fields and records are required. There is no performance gain from selecting fewer fields or fewer records: the access time is always the same. Indexes can

be used in the same way as for relational databases: in this case they enable direct access to individual records by providing the same efficient search algorithm. This can enable fast access where only a few records are required, but if more than a tenth, say, of the records are required then it may be more efficient to search the table linearly rather than use the index for random access.

Because the table exists as a single physical entity and is normally read in a linear manner, there are gains to be had from ensuring efficient storage. Defragmentation of the filesystem can ensure that tables are contiguous and striping of the filesystem across multiple disks together with read-ahead caching can also ramp up performance.

One technique which is only available on sequential databases is compression. At one level this consists of squashing the fields in a record but leaving the record structure intact. If there are many fields per record and many of the fields contain nulls or zeros this can achieve high compression ratios of 70% or more. Alternatively, the operating system can also provide filesystem compression: by viewing the table as stream of data and using one of the many stream compression algorithms, the operating system can reduce the amount of storage space required. This may produce a further 50% compression on top of the record compression.

Compression saves disk space, but more importantly can also improve performance. By reducing the amount of storage required, compression reduces the access time at the cost of increased processor usage. If multiple processors are available to handle the query and data decompression separately, then compression should substantially enhance performance.

Joins between tables are a particular problem for sequential databases. Indexes can solve the problem, but at the cost of turning efficient linear access into time-consuming random access. However, if it can be arranged that all the tables to be joined share one common join key and are sorted in the order of that key, then the join can be as efficient as accessing one large table.

In summary, the characteristics of a sequential database are:

- Performance can be much greater than for an equivalent relational database, even more so if compression, defragmentation and striping are used.

- But this is only true for queries which use most or all of the records in the table.
- Adding columns will require the table to be rewritten, as well as if records are added and the table is sorted.
- Table joins must be constructed with great care.
- Indexing should not be necessary if the tables have been correctly keyed and sorted.
- Storage volume may be many times smaller than with a relational database.

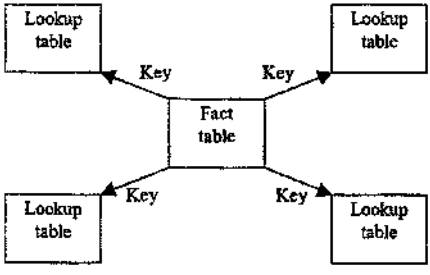
It would be wrong to say that either relational or sequential databases are best for warehousing; but they are very different and must not be viewed as interchangeable in usage. Almost certainly, for a specific application, one will heavily outperform the other for one of the reasons outlined above. Furthermore, some data models will only work properly on one type of database.

2. Data models

2.1 Relational data models

Star (normalised)

This is a very common model used only with relational databases. A central table (known as the fact table) holds quantitative variables such as volume, price, value, and lookup tables (known as dimensions) are joined to the central table with natural or artificial keys.



Example given in examples appendix.

Benefits:

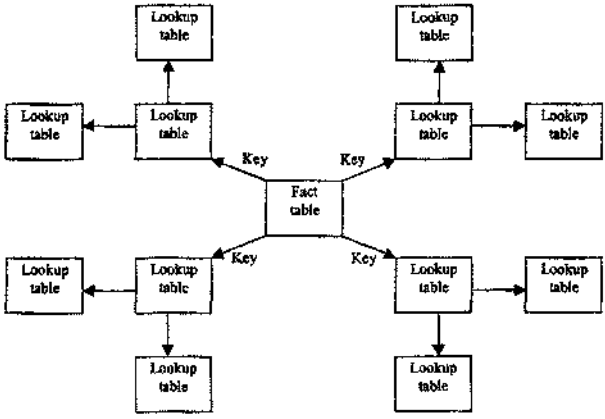
- Only the fact table has one record for every transaction (the lookup tables have relatively few records), so if there are few quantitative variables the data volume is minimised.
- If the dimensions can be arranged in a hierarchy, then one key can attach many classification variables, e.g. for a retail operation store → town → region → country is an appropriate hierarchy which derives from the store key.
- The model optimises drill-down tools which select levels from a hierarchy, e.g. select all the stores in the London area: the lookup table can quickly generate the keys to pull the appropriate records out of the fact table efficiently. Star models are best used for MIS applications where small amounts of highly selected data are used.

Problems:

- The model only works efficiently with highly hierarchical data structures. Retail is a good example, because all the classifications can be grouped into a few hierarchies, e.g. product line, location. Insurance is a bad example, because policy classifications are not hierarchical, e.g. age, car group, postcode, business use. If the number of dimension tables is greater than three or four the fact table will become too large (because there are too many keys in it) and the time taken to resolve the joins will be too long.
- If extra classification fields are introduced the model may have to be substantially rebuilt. If artificially generated keys are used to join the dimension tables to the fact table, then the key generating algorithm may fail if classification variables acquire new values.

Snowflake

The snowflake is an extension of the star, where the lookup tables have lookup tables. The data needs to be even more hierarchical, i.e. hierarchies within hierarchies.



The benefits and problems of the snowflake are similar to the star, but more exaggerated.

MDDB

The multi-dimensional database is a variation of the star with additional summary tables. Additional dimension tables are created with the lower levels of the hierarchy eliminated and supplementary fact tables are produced by summarising (in most cases adding-up) the main fact table to provide one record for each level of the new base of the hierarchy. For example, if the *location hierarchy goes postcode → post sector → post district → post area* (e.g. KT), then a summary table at the post area level would have only 124 values.

If the tool used is sufficiently intelligent to select one of the summary tables when the query permits and only use the full star model when required, then many queries can be substantially speeded up.

Benefits:

- In a MI application, a user will often start at a high level in a hierarchy (a query which can be resolved quickly using a summary table) and drill down to a small subset in a lower level of the hierarchy (a type of query which benefits from the ability of the star model to pull out small subsets

efficiently). The concept of MDDB fits well with this approach to MI systems and tools.

Problems:

- Summary tables can only be effectively created if they allow for permutations from only one or two hierarchies. If there are many dimensions, it will be necessary to create many summary tables allowing for all the likely combinations of one or two dimensions. The creation of these summary tables when the main fact table is refreshed will be time-consuming.

2.2 Sequential data models

Flat (denormalised)

The simplest possible data model: all data is arranged in one table. Each transaction appears as a record with all analysis and classification variables attached.

Record1	Field1	Field2	Field3	Field4	Field5
Record2	Field1	Field2	Field3	Field4	Field5
More records ...					

This data model is only really suitable for sequential databases, which can achieve optimum performance on a linear pass through one table. Relational databases can use this model, but tend to be slower because of the resolution of column links.

Benefits:

- Simple to create and refresh.
- Can achieve high performance with appropriate storage optimisation in a sequential database.
- If many fields have small integer values, then compression will be effective.
- Suitable for applications where many classification variables exist, but are not hierarchical, and where most queries use most of the records. Many insurance applications fall into this category, especially multivariate regression modelling.

Problems:

- Can be easily outperformed by the star model in cases where only a small number of records are normally read.
- If classification variables can be grouped into hierarchies, the star will take up less storage.

Split flat

This is a variation of the flat model, where the data is split into several tables. Each table has the same number of corresponding records, but the fields are divided into separate tables. The tables are used in the same way as with a flat model, but if queries use only the fields in one sub-table, then performance will be improved with a smaller table to read.

	Table 1	
Record1	Field1	Field2
Record2	Field1	Field2
More records ...		

Table2	
Field3	Field4
Field3	Field4

The model requires a smarter tool or application to detect which combination of tables is required to pick up the desired fields in the query.

Benefits:

- As for the flat model, but performance can be improved if queries can be regularly satisfied with only one sub-table.

Problems:

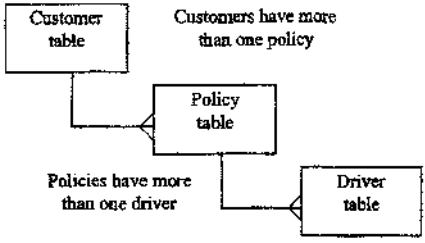
- Although the linear join is efficient, the performance when accessing all tables will not be quite as good as with a single table.

Cascade

A cascade model takes the idea of split flat a bit further, so that there are no longer equal numbers of records in each table. A useful example is that of insurance claims. In a flat model there would be one record for every claims transaction, e.g. a payment or a reserve change, with all claim details attached. However, many of these details are fixed throughout the claim, e.g. the date of loss or the type of incident, and it is inefficient to keep repeating these in the data. In a cascade, these would be separated into a table with one record per claim. Both tables would be sorted by a common key e.g. claim number,

which is unique in the smaller table and thus an efficient linear merge is created.

A cascade may have many levels: the characteristic of the cascade is that each step has a one-to-many relationship on a common key and that all tables are sorted in the order of the keys.



Example given in examples appendix.

Benefits:

- If many fields can be moved into the smaller table (at the top of the cascade), then there is a significant saving in space. This may also produce an improvement in performance. There is an overhead from the join, but since the join is linear it is efficient and may be more than offset by the reduced data volume to be read.
- Natural cascades are often found in transaction processing systems, especially insurance related ones, e.g. customer data → policy inception data → policy transaction data.

Problems:

- Only worth doing if a natural cascade is present in the data.

MDDB

The sequential form of multi-dimensional database is similar to the relational form. In this case, the summaries are built from a flat model instead of a star, but the objectives are the same: if the tool is sufficiently intelligent, it can satisfy queries using either the summaries or the entire flat table, whichever is appropriate.

Benefits:

- Essentially the same as for the relational model: if a query can be satisfied by a summary, then it will run much faster than it would have done over the whole table.

Problems:

- As before, maintaining the summaries requires time and resource, so the database must be well used to justify it.

3. Conclusion

The overall conclusions from the examination of databases and data models are straightforward but surprisingly often ignored:

- There is no data model or database which is always better than any other.
- But for a specific application there will probably be one model which substantially outperforms all the rest.
- This will depend largely on the nature of the data and the types of query to be run. For example, retail and insurance transactional data require wholly different models and the sort of drill-down tool a CEO will want to use will require a different data model from a multivariate regression application used by an actuary.
- If the data model is wrong, the warehouse will be more or less useless.

Appendix 3 - Data models examples

In these worked examples, two production systems (retail and insurance) will be considered and two data models (star and cascade, running on relational and sequential databases respectively) will be considered.

1. Production data

1.1 Retail

The basic unit of data is a transaction. This represents the sale of an item in a shop: it occurs at a particular location, involving one product, and is fixed forever after the event. All transactions are independent of other transactions.

The data collected for the transaction are:

Sale price	Item cost	Sales person	Manager	Personnel division
Store	Town	County	Region	
Product code	Supplier	Product category (e.g. groceries)		

1.2 Insurance (personal motor)

The basic units of data are periods of policy cover and claims transactions. A period of cover represents cover for one car from one point in time (start date) to another (end date). A policy may have a number of periods of cover, and a customer may have more than one policy. A period of cover may have more than one driver. Each claim belongs to one period of cover and may have a number of transactions.

Each entity described above has a number of attributes:

Customer:	Customer number	Age	Marital status
	Address	Postcode	
Policy:	Customer number	Policy number	Inception date
	Quotation date	Marketing source	
Policy cover:	Policy number	Start date	End date
	Premium	Cover type (e.g. TPFT)	
	Car group (and other car attributes)		Business use indicator
	Voluntary excess	No claims bonus	
Driver:	Policy number	Driver cover start date	
	Driver cover end date	Age	Main driver indicator
	Sex	Accident history, etc.	

Claim:	Policy number Notification date	Claim number Claim status	Loss date
Claim transaction:	Claim number	Amount paid	Revised estimate

The relationships between the entities are as follows:

Customers and policies are linked by their customer number, policies and policy covers are linked by policy number, drivers are linked to policy cover by the policy number and the dates of cover, claims are linked to policy cover by the policy number and whether the loss date fits in the period of cover, and claims transactions are linked to claims by the claim number.

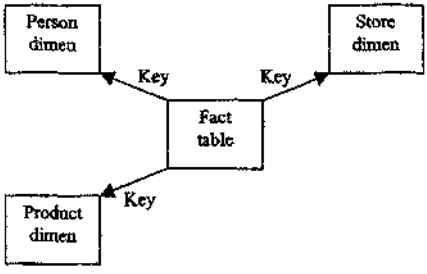
2. Data models

2.1 Star

2.1.1 Retail

There is one fact table and three dimension tables:

Fact:	Sale price Store key	Item cost Product key	Person key
Person dimension:	Person key Personnel division	Sales person	Manager
Store dimension:	Store key County	Store Region	Town
Product dimension:	Product key Product category	Product code	Supplier



The dimension tables are joined to the fact tables by the three respective keys, which are indexed in each table. The dimension tables only need to be refreshed if there are new personnel, stores or products. The fact table need only be refreshed by adding new transactions, since existing ones can never change. Even if new keys are introduced, these only affect new transactions.

Natural hierarchies exist in the dimensions: Each person belongs to one manager, who belongs to one division. Each store exists in only one town, which is in one county, which is in one region. Products come from one supplier, and only one category, although in this case there is a bifurcation in the hierarchy because a category has many suppliers, and a supplier may produce goods for many categories.

This model is efficient:

- There are only three dimension tables, so there are few joins.
- The fact table has few columns, so it is a lot smaller than a flat model would have been, for example.
- The dimensions have few rows: the number of personnel or the number of stores is tiny compared to the number of transactions.
- This probably makes the joins even more efficient because the keys have few values and the indexes will resolve easily.

2.1.2 Insurance

The insurance data will not actually fit a star in the normal sense, because there is more than one transaction stream present and so there will be more than one fact table. It is possible to construct a binary star model with two fact tables or even more, but a query which spans both fact tables could be very difficult to resolve. For the sake of simplicity, the drivers and the claims data will be ignored completely for the moment.

There will be one fact table and thirteen (sic!) dimension tables:

Fact:	Premium	Customer age key
	Customer marital status key	Customer address key
	Inception date key	Quotation date key
	Marketing source key	Policy start date key
	Policy end date key	Cover type key
	Car key	Business use key
	Voluntary excess key	No claims bonus key

Customer age dimension:	Customer age key
	Customer age

Customer marital status dimension:	Customer marital status key
	Customer marital status
Other dimensions ...	

It should be noted that almost all dimensions consist of a key and only one attribute: this is because the attributes are non-hierarchical, e.g. business use and voluntary excess are unrelated and so cannot appear in the same dimension. Many of these dimensions are large: the date dimensions have to contain one record for every possible date.

This model is a complete non-starter, because:

- There are way too many dimensions (and this is a very simplified example). The joins will take forever to resolve.
- The fact table will be enormous: as big as a flat model table would have been anyway, with all those keys.
- The dimensions have many rows too, and that will not help the joins either.

2.2 Cascade

2.2.1 Retail

No natural cascade exists in this transaction data, so effectively this is a flat model.

There is one table, just a list of transactions with all attributes attached.

Sale price	Item cost	Sales person	Manager	Personnel division
Store	Town	County	Region	
Product code	Supplier	Product category		

This model is viable, but probably not as efficient as the star because:

- There will now be twelve fields in the transaction table instead of five. In this example, an efficient sequential database using compression could probably close the gap in terms of performance, but in real life the ratio could be much bigger than this if there were more attributes sitting in the dimensions.

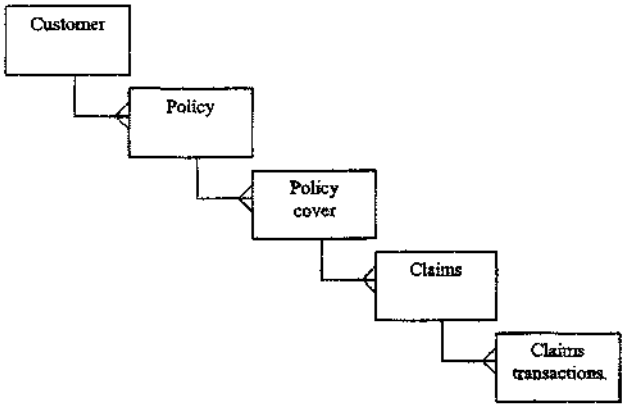
2.2.2 Insurance

There are five cascading tables: customer, policy, policy cover and drivers, claims and claims transactions.

In order to perform the cascade, each table must have customer number added (via the relationships described in the data) and the claims transactions must have the loss date added.

In order to prevent a bifurcation of the cascade, the drivers must be summarised into the policy cover data. If the claims data were not required, the drivers could become the bottom level of the cascade, without summarisation, but with claims there would be no clear cascade from drivers to claims.

Customer:	Customer number	Age	Marital status
	Address	Postcode	
Policy:	Customer number	Policy number	Inception date
	Quotation date	Marketing source	
Policy cover:	Customer number	Policy number	Start date
	End date	Premium	
	Cover type (e.g. TPFT)	Car group (and other car attributes)	
	Business use indicator	Voluntary excess	No claims bonus
	Number of drivers	Main driver age	Main driver sex
Claim:	Customer number	Policy number	Claim number
	Loss date	Notification date	Claim status
Claim transaction:	Customer number	Policy number	Loss date
	Claim number	Amount paid	Revised estimate



All tables are sorted in this order: by customer number, by policy number, by policy cover start date or claim loss date (these two are equivalent) and by claim number.

It is possible to simplify the cascade and save space by replacing the combination of two or more join keys with a single key possessing the same sort order. The tables then become:

Customer:	Customer number	Age	Marital status
	Address	Postcode	
Policy:	Customer number	Policy key	Inception date
	Quotation date	Marketing source	
Policy cover:	Policy key	Policy cover key	Start date
	End date	Premium	
	Cover type (e.g. TPFT)	Car group (and other car attributes)	
	Business use indicator	Voluntary excess	No claims bonus
	Number of drivers	Main driver age	Main driver sex
Claim:	Policy key	Claim number	Claim key
	Loss date	Notification date	Claim status
Claim transaction:	Claim key	Amount paid	Revised estimate

The space saving becomes greatest at the base of the cascade, if there are many levels.

This model is efficient, because:

- All the tables share the same sort order, so the joins are linear and can be resolved rapidly in a sequential database without the use of indexes.
- Apart from the join keys, there is no duplication of data in records, e.g. customer data is held once only for each customer. This model is close to the theoretical minimum space required to hold this information.

Appendix 4 - Tools

1. Introduction

A data warehouse should provide a controlled, cleansed source of data. For a data warehousing project to be implemented successfully this data needs to be accessed and turned into useful information. There are many different tools available from software companies to choose from, alternatively a custom made application can either be developed in-house or written by outside consultants. This appendix briefly describes the different types of tools which are available in general terms.

2. Types of tools

Almost any product that uses data could be used with a data warehouse. There are however several different classes of product that are frequently mentioned in the context of data warehousing and these are described below. There is some overlap in the definitions of these classes and one product may fit into several categories. Business Intelligence is often used as a general description of these tools.

2.1 Report and query systems

These may also be called Decision Support Systems (DSS). These are used to build ad-hoc queries and generate reports.

2.2 Executive Information Systems (EIS)

These products have been around for several years. They are designed to be very easy to use - typically being "point and click" - but are not as flexible as most of the other tools. Typical features would be custom reporting and graphing, limited what-if analysis and exception reporting.

2.3 OLAP and multidimensional engines

Multi-dimensional data structures can provide a powerful way to analyse data stored in a warehouse. Multi-dimensional tools that access these data structures are often referred to as On-Line Analytical Processing (OLAP) products. Such products are normally aimed at non-technical end-users and would usually allow users to "slice & dice", "drill down" into data and

produce reports. Some OLAP products are designed to serve as front-ends to specific database products and structures. OLAP technology is comparatively recent and while one of its big selling points has been speed of access to large databases through the use of the multi-dimensional structure, other approaches can give similar performance.

OLAP is now often used to refer to the concept of warehousing and products are described as (a) MOLAP (multi-dimensional OLAP) which uses summary tables to give quick response (see MDDB in appendix 2), (b) ROLAP (relational OLAP) which refers to OLAP using relational models such as the star model, and (c) HOLAP (hybrid OLAP) which combines elements of MOLAP and ROLAP.

2.4 Data Mining Tools

Data mining is the latest IT warehousing product and as such has been much hyped, although some of the products that are now sold as data mining packages would have previously been sold under another label. These products aim to unlock the information within a large body of data by highlighting possible relationships between different variables. Such products go one stage further than traditional decision support systems and EIS tools. With traditional products the user forms a hypothesis and uses the query tools to verify or reject the hypothesis. With data mining the system researches the data and determines patterns, classifications and associations without being specifically guided. There is always a danger that such a tool will come up with spurious relationships, since it has no intuitive knowledge of likely or expected relationships. Many data mining applications quoted are in the retail sector but one area often quoted in insurance where data mining has been used is in fraud detection. Data mining applications are now being marketed for use in premium rating.

When using data mining software it is particularly important that the data structures are correctly set up and fields properly defined. Also, because of the higher level of automation, it is even more important than usual that the data is clean and accurate or at least that known exceptions are excluded from the analysis.

Various techniques can be used in data mining and different packages will provide different ranges of techniques. It is therefore important to decide which techniques are likely to be successful with the available data, and then

to ensure that the selected product covers such techniques. Some of the methods used are very advanced such as fuzzy logic, neural networks and genetic algorithms, while others such as regression analysis are more established. Data mining tools would generally be used by specialist operators such as statisticians or actuaries, and in the US there are now specialist data mining engineers, although some packages claim to allow non-specialists to use them. There seems to be some uncertainty over the precise definition of data mining. Some literature states very clearly that it is not statistical analysis even though other sources refer to statistical techniques being used.

3. User requirements

To illustrate the range of possibilities which a data tool must cover, three user types are examined:

3.1 Chief Executive Officer (CEO)

The CEO is not going to have time to learn to use a complex system, so an easy to use, point and click, intuitive system is essential. His computer skills are likely to be limited, so writing program code is not possible and in fact using the keyboard is probably best avoided altogether. In general he will be looking at high level data probably no more detailed than line of business. There will be certain queries and reports that he wants to see regularly so custom views and reports that can be easily set up are useful. Flexibility will not be a major requirement and complexity is almost certainly not required. He will want to be able to spot problems or trends as soon as possible so reporting by exception, traffic-lighting and simple trend analysis may be appropriate. Speed will be more critical than for any other user: he will expect immediate responses to his requests or he will not bother to use the system.

3.2 Actuary or Statistician

Whilst ease of use is always helpful an actuary may be more willing to spend time learning a more complex system if it will be useful in the future. Actuaries will have some regular requirements but will also have far more one-off requests, so flexibility is much more important for an actuarial user than a CEO. For unusual one-off requests a slower response time may be acceptable so long as the more routine queries are fast. Actuaries will look at many levels of data from total company down to individual policies and claims. They will be interested in more variables than a CEO who will tend to

concentrate on the key performance indicators. Actuaries may also require specialised software for reserving, rating etc. It is therefore likely to be important that data can easily be moved into other software environments. Actuaries are computer literate and will therefore usually be happy to write some code themselves.

3.3 Clerk

A clerk will probably be concerned with individual transactions rather than company level information. Apart from this difference, the clerk's requirements may be very similar to the CEOs. The clerk will be doing standard queries or reports rather than ad-hoc queries and will probably have limited computer expertise so an easy to learn, easy to use system will be needed. This may also be important if there is high turnover of such staff. Speed will again be critical since such employees will be expected to deal with many transactions a day.

4. Evaluation of Data Warehousing Tools

Rather than worry too much about whether software is an EIS, DSS, MIS, OLAP, business intelligence, report and query, data mining or statistical analysis product, a better starting point is to consider what users want or need to be able to do with the data. Various approaches and packages can then be evaluated to see how they meet the users' needs and at what cost. It would be possible to build a warehouse and then buy or write applications for it. However it is appropriate to at least consider the type of tools to be used before the warehouse is built as certain tools perform better with certain data structures.

A fundamental decision is how much work will be done in-house, how much by consultants and how much is to be bought off the shelf. The advantages of using consultants are that they may have done warehousing applications before and fewer in-house resources will be used. The disadvantages are that they may not understand insurance, they will certainly not be as familiar with the company's systems and data as internal staff, and they can be expensive. When considering consultants check with which companies (both software and hardware) they are associated. Some consultants are tied so that they always use the same hardware or software regardless of application. This may be obvious if the consultants are specifically part of a larger group, but many ties exist between software companies and otherwise independent consultants.

When selecting tools to use, the areas of expertise already within the company or that can easily be bought into the company both within the IT area and in other areas, may be an important factor in the decision. For example, if a company already uses SAS and has considerable in-house SAS knowledge then SAS may be preferred to other options. Conversely there is no point in using data mining tools if there are no suitably qualified statisticians to interpret the results.

A further decision is whether to use one system throughout, use different products from the same company, or use a variety of different tools from different companies. There may be corporate IT strategies to consider, e.g. rules about which companies will be considered as software suppliers or consultants. A company may prefer to use a longer established supplier with a track record rather than a new small specialist data warehousing company even if the latter has a more state-of-the-art product. It may also be preferable to use an established product than one which may experience teething troubles. When looking at a potential supplier's expertise it is worth considering not only whether they have data warehousing experience but also whether they have experience of doing projects for similar insurance companies. A supplier with lots of experience in direct motor may not be in the best position to implement a solution for a London Market company.

Response time is often a crucial factor in companies' data warehousing decisions. If response times are not adequate the solution is almost certainly unacceptable. Response time will usually depend on the volumes of data being interrogated, how the data is stored, how the data is accessed and what hardware is being used. Some products may give acceptable performance on summarised data but not on detailed data. If only a small amount of data is usually queried then it may be more efficient to hold this on individual PCs. If large volumes of data are being queried (as is typically the case in insurance companies) then it will be necessary to hold the data on a server. A key issue is where the processing will take place. True client/server tools which do not rely on processing at the desktop will usually be more efficient.

An important consideration is flexibility. This may mean flexibility within a product or the flexibility to amend the tool, e.g. to extend functionality or cope with a change in the underlying data. In terms of flexibility to amend a product, an off the shelf package would score poorly whereas a well-documented in-house tool would score well. There are many packages which

allow development of custom made applications and these may be as flexible as an in-house tool but this needs to be evaluated carefully. Flexibility in use is not always useful if it makes a product harder to use or configure.

Ease of use is another critical factor. For a data warehouse to be successful a wide range of people must be able to use the data. The aim should be to allow all users to do what they need to easily and quickly. Ease of use is to some extent a matter of personal taste and will also depend on the computer background of users and their other expertise. Considerations can include how long it takes to learn to use the system and how easy it is to use with familiarity. Items to look at are (a) manuals or on-line help, (b) is there a help desk and how good is it?

Control of information may be an issue. It might be considered desirable to limit the views that end-users can see and limit the creation of new fields. This will help stop meaningless or incorrect reports being created and help ensure standardisation within the organisation.

The time that will be needed to implement different solutions must be considered. It may be better to set up an adequate system quickly rather than spend years developing the perfect system. A system which takes 5 years to complete will be obsolete immediately.

Cost is almost certain to be an issue. Evaluating the cost of various tools is not necessarily a simple task. The cost of any software should be straightforward but extra costs are (a) the cost of any consultancy, (b) the cost of in-house IT involvement (which even when outside consultants are involved can be considerable), (c) the cost of other personnel who need to be involved (for example, users' specifying data structures), (d) the costs of training. Different tools may have different hardware requirements, for example one tool may require a high-specification PC for every user.

It could be argued that the most advanced all singing all dancing solution which can be afforded should be bought; certainly many software companies would take this view. However complexity may not be beneficial, functionality which is not used is a waste of money and there will probably be a trade off between complexity and ease or speed of use. With data mining and other statistical techniques careful thought must be given to whether the data quality or volume is appropriate and more generally whether the techniques are even valid in an insurance context.

The best way to evaluate different tools is to see them in operation and if at all possible use them on real data to see if they cope with the company's data structure and volume. All tools can look good: fast and flexible with sample databases used to demonstrate their capabilities. The difficulty is to assess whether it will be flexible enough to cope with real data and fast enough when handling the volumes of data in the warehouse.

Every company is different, and there is no ideal solution for all companies or even for all motor insurers. It is important to get the tools that are right for the precise needs of the company and fit in with the company's expertise, philosophy and IT strategy.