THE ERASMUS INSURANCE CASE

AND

A RELATED QUESTIONNAIRE FOR

DISTRIBUTED DATABASE MANAGEMENT SYSTEMS

Saskia C. van der Made- Potuijt,
Department of Computer-Science
Erasmus University Rotterdam.

Preface

This is the third report concerning transaction managemant in the database environment. In the first report the role of the transaction manager in protecting the integrity of a database has been studied [MaNe]. In the second report a model has been given for a transaction manager as a parallel decision process [MGMo]. In this report a case, the Erasmus Insurance Case
(EI-Case) will be presented that can be used to test transaction management in a centralized and distributed environment. It can also be used to describe transaction management in a distributed environment as parallel decision processes.
The purpose of the questionnaire is to investigate the functionality of a particular distributed database management system. The answers of the questionnaire can be certified by tests on an implementation of a case. To this aim tests are added to the questions, referring to the implementation of the EI-case in the Distributed DBMS under investigation.

1984 CR catagories: C.1.2, H.2.4, K.6.2.
1980 Mathematics subject classification (1985 revision): 68M15, 68P15.
Keywords & Phrases: Integrity, Transaction Processing.

Introduction.

Because integrity is the central aspect of transaction management, a case to test transaction management must provide adequate integrity constraints. The EI-Case contains static constraints, both update and non update, and dynamic constraints. The Case is designed with the purpose of having aspects related to transaction management in it, but almost as little as possible attributes, relations and other constraints. It can easily be expanded by less or more restrictions, more attributes and other entities. In the informal description of the EI-Case a few possibilities for an extension will be given.

The description of the constraints in the Case is grouped into modules. This makes the Case very convenient for the testing of implementations. Only the modules that contain those aspects that are of interest in a certain situation can be implemented and tested. After a description of the datamodel the three constraint-modules will be presented. After this, a schema for the Case in a distributed environment will be given where the consequences of the constraints will have our special attention.

I. THE DESCRIPTION OF THE DATAMODEL.

First we will give a short informal introduction. The EI-Case is a part of the administration of an insurance company. This company is specialized in contracts with other parties, like firms, companies, non-profit organizations etc.. For convenience the contracted parties will be called 'companies'. The contracts offer an insurance for all the employees of the contracted party. These contracts however are not all different: there

are a few standard contracts and all the companies have one of these standard contracts. A new standard contract can be designed if a potential client does not consider any of the existing standard contracts satisfactory.
Although for an individual employee certain clauses of the insurance are determined by the contract between the company where the employee works and the insurance company, some individual wishes for this employee are usually possible. In the Case this is realized for only one individual wish, the height of the own risk percentage. In the contract between the insurance company and the employer it is determined what range of own risk percentage is allowed and whether the employee can increase and/or decrease the individual own risk percentage.
The insurance company offers a few types of insurances, health, car etc., but the Case is restricted to health insurance only. Furthermore it is possible for individuals to have an individual insurance policy with this insurance company, but these are not included in the Case description either.
One or more contactpersons of a client company are involved in the negotiations before a contract is signed. Certain data of these persons are registered, furthermore it is registered who the main contactperson is for that particular company.

More formally, there are four relations:

1. Contracttype, with the description of the standard contracts.

2. Company, with the description of the companies that are a client of the insurance company.

3. Contactperson, with the description of the people involved in the negotiations and signing of the contract.

4. Employee, all the employees that work for one of the companies, they are insured by the contract of their employer.

The relations have the following attributes with their respective domains:

Contracttype:

CTID	ContractTypeIDentification, the primary key that identifies a type of contract. Domain: a letter.
ORRA	OwnRiskRAnge, the percentages that are allowed as own risk are here registered. Domain: a range of 5% up to 70% with steps of 5%.
ORD	OwnRiskDirection, here is registered whether the own risk- percentage may be Increased, Decreased, Both or None. Domain: the list { I, D, B, N }.

Company:

Cname	The primary key, the name of the company. Domain: string with length 30.
Ctype	The type of company, for instance a supermarket chain. Domain: string with length 20.
Cstatus	An indicator for the status of the company as a client: is it a potential client, a new client, a 'stable' client or a former client. Domain: The list { Potential, New, Stable, Former }
Address	The address of the department involved in the contract. Domain: string with length 30.
Postcode	According to the dutch structure Domain: 4 digits followed by 2 letters.
Place	The city or town where the department involved in the contract is situated. Domain: string with length 30.
Region	The insurance company has divided the country into several regions. This attribute indicates in which region the place (the previous attribute) is. Domain : A letter.
Tel	The telephone number of the secretary of the department involved in the contract. Domain: 3 or 5 digits followed by a number of 4,5,6 or 7 digits.
CTID	The type of contract this company has signed with the insurance company. It is a foreign key, referring to the primary key in the relation Contracttype. Domain: a letter.
Pname	The name of the main contactperson for this company. It is a foreign key, referring to the primary key in the relation Contactperson. Domain: string with length 30.

Contactperson:

Pname	Primary key, the name of the contactperson. Domain: string with length 30.
Dept	The name of the department where this person works. Domain: string with length 20.
Function	The function of this person. Domain: string with length 20.
Pdesr	a short description of this person Domain: text.
Tel	The telephone number of this person. Domain: 3 or 5 digits followed by a number of 4,5,6 or 7 digits.
Cname	The name of the company for which this person is contactperson. It is a foreign key, referring to the primary key in the relation Company. Domain: string with length 30.
MPname	The name of the main contactperson of the same company. It is a foreign key, referring to a tuple in the same relation. Domain: string with length 30.

Employee:

Enr	The primary key, the number of the employee. Domain: A number of 7 digits.
Remark:	Some companies have their own system of giving numbers to emplo-yee's. These systems cannot be used, because they are not uniform for the insurance company and the numbers might not be identi-fying. We assumed that there is only one type of insurance registered in the database, therefore, the number of the policy could as well be used to identify an employee. This would have the disadvantage of not being able to expand to a situation where one employee might have more than one policy with this insurance company.
Ename	The name of the employee. Domain: string with length 30.
Address	The addresss of the employee. Domain: string with length 30.
Postcode	According to the dutch structure. Domain: 4 digits followed by 2 letters.
Place	The city or town where the employee lives. Domain: string with length 30.
Bdate	Day of birth. Domain: YY-MM-DD
ORP	OwnRiskPercentage. Domain: a range of 5% up to 70% with steps of 5%.
Bankacc	The number of the bank-account. Domain: according to the dutch system a number of 9 digits, divisible by 11.
Tdate	The date a healthtest is done. Domain: YY-MM-DD.
Treport	The report with the result of the test. Domain: text.
Cname	The name of the company the employee works for. It is a foreign key referring to the primary key of the relation Company. Domain: string with length 30.

The associations, represented by foreign keys among the relations are shown in figure 1.

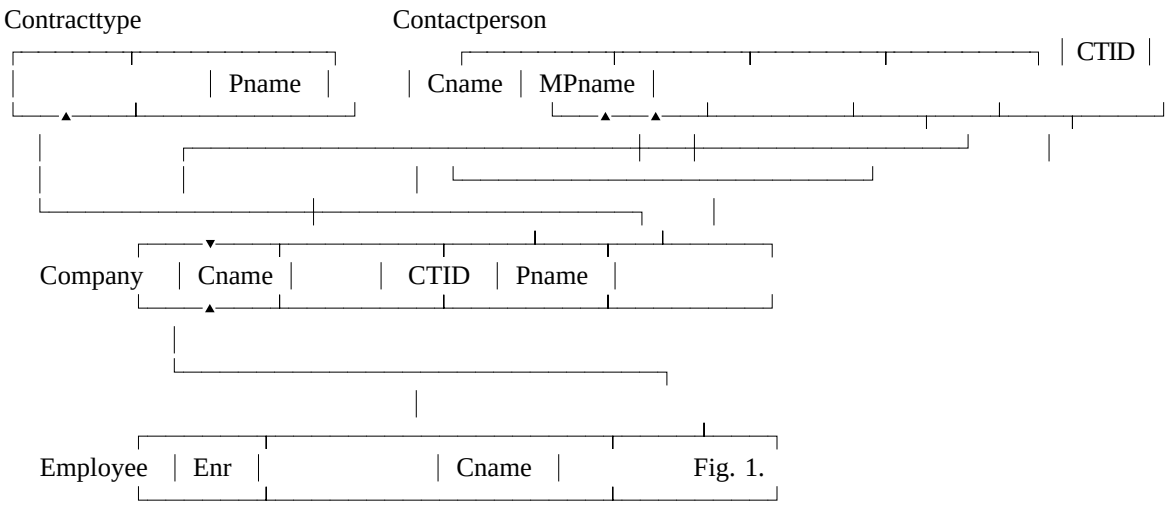


Fig. 1.

II. THE STATIC NON-UPDATE CONSTRAINTS.

Static non-update constraints can be subdivided into four groups, attribute, tuple, relation and database constraints. The constraints have a unique code, two letters that indicate what type of constraint it is, followed by a number.

A. Attribute constraints

An attribute constraint will always refer to a domain. In general it is a further restriction of a domain, but if the domain is specified well, an attribute can often have all the values of the domain.

Attribute constraints can be divided into the next four types:

1. Range or interval constraints.

at1. Ownriskrange in relation contracttype.
Domain: A range of 5% up to 70% with steps of 5%.
Remark: This attribute constraint is no further restriction of the domain.

at2. Birthday in relation employee
Domain: YY-MM-DD
Constraint: The age of an employee is higher than 15.
Remarks: Employee's that reach their age of pension can remain in the insurance scheme of their former company.
Checking this constraint involves a calculation.

2. Exclusion.

at3. In all relations the value of the primary key not allowed to be null.

at4. Furthermore the next attributes are not allowed to be null.
In relation:
Contracttype ORRA and ORD.
Company Cstatus, Address, Postcode, Place, Region, Tel and Pname.
Contactperson Pdesr and Tel.
Employee Ename, Address, Postcode, Place, Bdate and ORP.

3. A random set of values, a list.

at5. Cstatus in relation Company.
Domain: { Potential, New, Stable, Former }
Remark: This constraint is no further restriction of the domain.

at6. ORD in relation Contracttype
Domain: { I, D, B, N }
Remark: This constraint is no further restriction of the domain.

4. A set of values, the values are determined by calculations.

at7. Bankacc in table Employee.
Domain: A number of 9 digits, divisible by 11.
Remark: This constraint is no further restriction of the domain.

B. Tuple constraints

This describes the mutual dependency of attribute values within the same tuple occurrence.
There are two tuple constraints in the case, both in the relation Employee.

tu1. In relation Employee. A healthtest is always done after the person is born. Bdate < Tdate.

tu2. If a healthtest is done, there should be a report of it and if there is a report, a healthtest should be done.
In other words: If Tdate <> null then Treport <> null, and

if Treport <> null then Tdate <> null.

Remark: tu2 is a mutual dependency between two attributes:
if one of the attributes is not null, the other one is neither allowed to be null.

C. Relation or table constraints

Four types of relation constraints can be distinguished:

1. Unicity.

ta1. In all relations the value of the primary key is unique.

2. Referential integrity within one relation.

This is also called a intra-relation reference.

ta2. In relation Contactperson: The main Contactperson for a company is also a contactperson

3. General relation constraints, like functional and multi-valued dependencies.

ta3. In relation Company: All companies registered in the same place have the same area telephone code.

4. Specific relation constraints.

ta4. In relation Contactperson: The number of contactpersons for one company should not exceed 5.

D. Database constraints

These can be divided into two groups:

1. Referential integrity rules between tuples in different relations.

There are two situations to be distinguished for referential integrity: normal and cyclic.

Normal.

db1. Between the relations Company and Contracttype.
A company has one type of contract.
This relation is represented by the foreign key CTID in the relation Company, referring to the primary key of relation Contracttype.

db2. Between relation Employee and Company.
An employee works for one company.
This relation is represented by the foreign key Cname in the relation Employee, referring to the primary key of relation Company.

Cyclic.

A cyclic referential integrity relation occurs if the chain of two or more normal referential references between relations is closed. In the EI-Case this is the situation for the relations Company

and Contactperson.

db3. From relation Company to relation Contactperson.
A company has one main contactperson.
This relation is represented by the foreign key Pname in the relation Company, referring to the primary key of relation Contactperson.

db4. From relation Contactperson to relation Company.
A contactperson always works for one company.
This relation is represented by the foreign key Cname in the relation Contactperson, referring to the primary key of relation Company.

The referential integrity relations are indicated in figure 1.

2. Other database constraints, e.g. functional dependencies.

db5. The area code of the telephone number of a contactperson should be the same as the area code of the telephone number of the company for which the person is the contactperson.

db6. The own risk percentage of an employee should be in the range that belongs to the contracttype of the contract of his/her employer.

III. THE UPDATE CONSTRAINTS.

A. A short tutorial introduction.

Because an update can take place in a relation where tuples in other relations refer to, certain rules have to be specified in order to maintain referential integrity. These rules, update constraints, determine what should happen to tuples in related relations. Update constraints have to be specified in situations where referential integrity is involved and are desired in some other situations. Update constraints can be divided into delete, update and insert rules.

A delete rule must be specified for every relation that contains a foreign key to another relation. It determines what should happen to the foreign-key tuples (tuples that contain a foreign key) if the corresponding primary-key tuple is deleted. Three standard options are often used for a delete rule:

- Restrict
  A primary-key tuple can only be deleted if there are no referring foreign-key tuples.

- Cascade
  If a primary-key tuple is deleted, the corresponding foreign-key tuples will be deleted also.

- Nullifies
  If a primary-key tuple is deleted, the corresponding foreign-keys
  will be made null.

Not all the desirable possibilities however are captured in these three standard options as we shall see in the delete rule for the relation Company.

An update rule must, just like a delete rule, be specified for every relation that contains a foreign key to another relation. It determines what should happen to the foreign-key tuples if the corresponding primary-key tuple is updated. The same three standard options as for the delete rule are used for this rule. Sometimes it is necessary to implement an alternative rule. In the Case there is an alternative update rule for the relation Company.

- Restrict
  A primary-key tuple can only be updated if there are no referring foreign-key tuples.

- Cascade
  If a primary-key tuple is updated, the corresponding foreign-key tuples are updated accordingly.

- Nullifies
  If a primary-key tuple is updated, the corresponding foreign-keys will be made null.

Referential integrity implies certain insert rules: it only allows for an insert of tuples in a relation A with a foreign key to relation B if there is a matching primary key in relation B. Referential integrity often determines the order for inserting related tuples: first the tuples have to be inserted in table B where the tuples in table A refer to.

B. The update constraints in the EI-Case.

For the three relations in the EI-Case that contain tuples where tuples in other relations refer to, Contracttype, Company and Contactpersons, it has to be specified what happens at a delete or an update of the primary key of a tuple. Insert rules have to be specified for the relations that refer to other relations, so for the relations Employee, Company and Contactperson.

Relation Contracttype.

Delete rule restricted.
You cannot delete the description of a contracttype if there still is a company with this type of contract.

Update rule restricted.
It is not allowed to update the contracttype identification if there is a company with this type of contract.

Relation Contactperson.

A complicating factor is that there are two types of contactpersons: 'normal' ones and 'main' ones, with an intra-relation reference of the normal contactperson to the main contactperson. To a normal contactperson there is no referring key: a company has only a reference to the main contactperson.

Delete rule restricted (only for the main contactpersons).
For every company there should be at least one contactperson. You cannot delete the main contactperson if there still exists a company referring to this person.

Update rule cascade.
If a normal contactperson becomes main contactperson for a company, this involves a few updates:

1. Delete the former main contactperson or adjust his/her foreign key to the key of the new main contactperson.
2. Update the foreign key MPname in the tuple with the description of the new contactperson,
3. All the references of the normal contactpersons of this company have to be adjusted to the new situation.
4. In the relation Company the attribute MPname has to be updated.

During these updates the referential integrity rules may be violated:
If update 1 or 2 is done first, the foreign key MPname in Company is not referring to the correct Main Contactperson. If update 3 or 4 is done first, the company refers to another main contactperson than the contactpersons in the relation Contactpersons. In order not to violate any referential integrity rule, the four updates have to be done in one transaction and the check on the integrity rules has to be done at the moment of commit.

Insert rule.
It is easy to insert a new normal contactperson for a company: the company has to be present in the relation Company and the main contactperson for this Company (the value of attribute Pname in the new tuple) has to be present in the relation Contactperson.
There is however a problem if the main contactperson is inserted. The company should already be present in the Company relation, but in that tuple you have to refer to the main contactperson in relation Contactperson (the value of attribute Pname is not allowed to be null). This cyclic control problem can be solved by using a transaction to add the tuple that represents the contactperson and to update the value of the referring foreign key in the relation Company. The integrity rules have to be checked at the moment of commit.

Relation Company.

The delete rule.
If a company is deleted this can have different reasons that demand different update rules.

1. Delete Company Option 1:
   The company is broke and seizes to exist. The company is deleted, all its employees and all the contactpersons are deleted also (a delete cascade in two directions).
2. Delete Company Option 2:
   The company has not prolonged the contract.
   The company remains registered as a 'former' client (an update in relation Company) and all the contactpersons and employees remain registered as belonging to this company. The marketing and sales department of the Insurance Company will try to regain this company as a client. This is clearly a non-standard option.

The update rule.
An update of the primary key is a update of the name of the company, this can be necessary if the company is bought by another company. Again at least two situations can be distinguished that demand different update rules.

1. Update Company Option 1:
   The company is bought by another company that also has a contract with the insurance company. The company will be deleted, all its employees remain registered, the value of their field Cname is updated to the name of the new company. The contactpersons of this company however are deleted.
   This rule means a cascade update in the direction of the employees and a delete cascade in the direction of the contactpersons.

2. Update Company Option 2:
   The company is bought by another company that does not have a contract with the insurance company. The company is deleted, all its employees remain registered, the value of their field Cname is made null. The contactpersons of this company will not be deleted as they will be approached to suggest a contract of the buying firm with the insurance company.
   This rule means a nullifies in the direction of the employees and a cascade update in the direction of the contactpersons.

The insert rule.
At least 2 situations can be distinguished for the insert of a company:

1. Insert Company Option 1:
   A company has contacted the insurance firm with interest in a contract. The company is inserted with status Potential.

2. Insert Company Option 2:
   A company has decided already that it wants a contract with the insurance company. The client company is inserted with status New.

Before a new company is inserted, if necessary a new contracttype will also be inserted.

Relation Employee.

The delete rule.
An employee can be deleted without any problem. If the last employee of a company is deleted, a message should be given to warn the insurance company.

The update rule.
An employee cannot freely change his or her own risk. The dynamic databaseconstraint, that will be discussed in more detail in the next paragraph, restricts this.
It is also possible that an employee finds another job. If the new company also has a contract with the insurance company the employee can remain registered. We will have a closer look at these updates when we discuss the involved dynamic constraint.

The insert rule.
An employee can only be inserted if the related company is present in the relation Company.

IV. THE DYNAMIC CONSTRAINTS.

The dynamic or transition constraints describe which transitions are allowed between two states in the database. This can be related to attributes, tuples, relations and the database as a whole.

dy1. A dynamic attribute constraint:

In accordance with fig.2. a potential client (a company) can become a New client, a New client can become a Stable client etc.

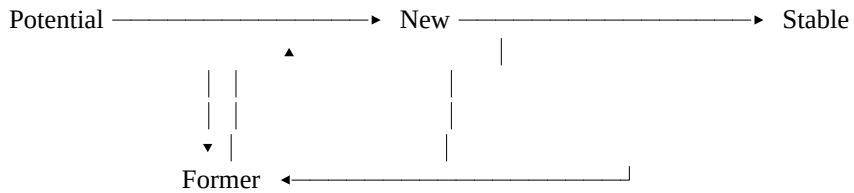


Fig.2.

Remark: A newly inserted company can only be registered as being Potential or New.

dy2. A dynamic database constraint:

In the description of the contracttypes the value of the attribute OwnRiskDirection determines whether a certain employee is allowed to increase or decrease his/her own risk percentage.
To check whether an update of the field OwnRiskPercentage of an employee is allowed, the relation with the description of the contracttypes's has to be approached, via the relation of the companies.

The next figure shows two states of the database. State 1 describes the situation of an employee who works for company X, who is not allowed to decrease his or her own risk percentage because the OwnRiskDirection of the contract that his/her company has with the insurance company is of the type T1, with OwnRiskDirection I. The transition of state 1 to state 2 is therefore not allowed.

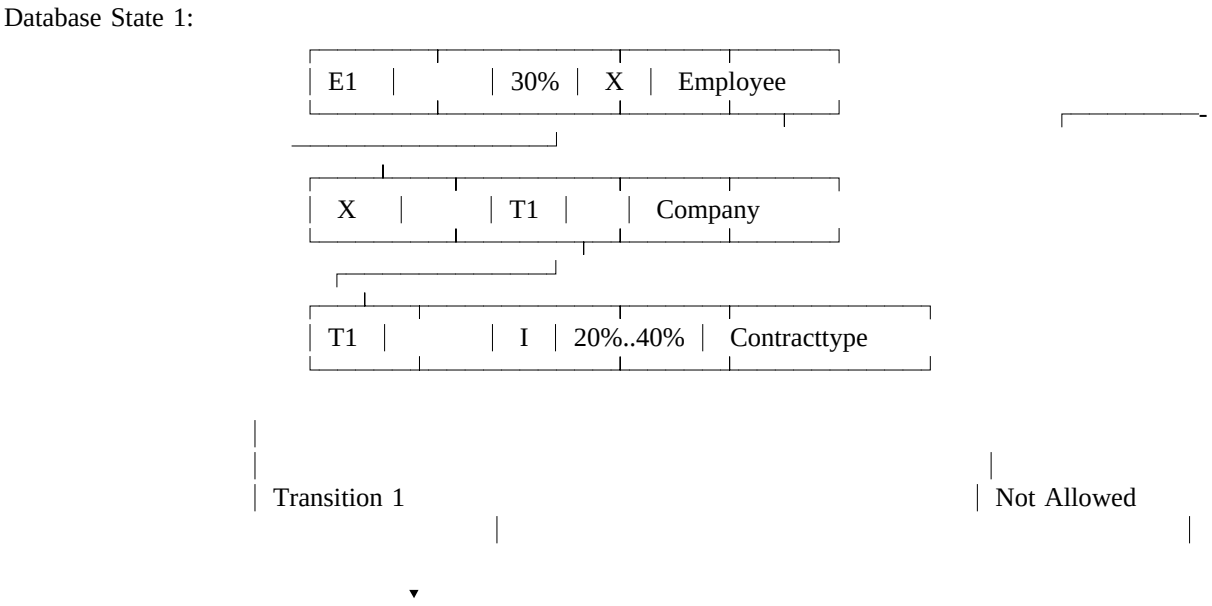
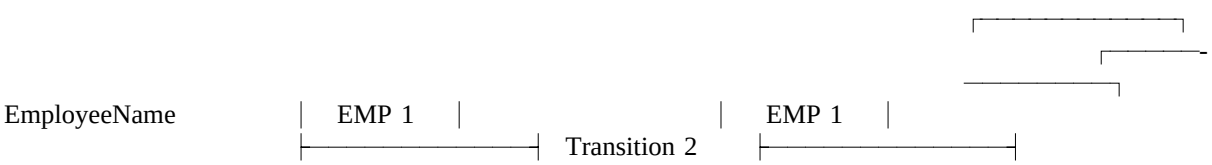


Figure 3.

We will now have a closer look at the dynamic constraint and pay attention to the influence of static database constraint db6: An employee is not allowed to have an own risk percentage that is not in the range belonging to the type of contract of his/her company.
The situation is reflected in fig.4. It does not represent the separate relations, only relevant parts of database states.



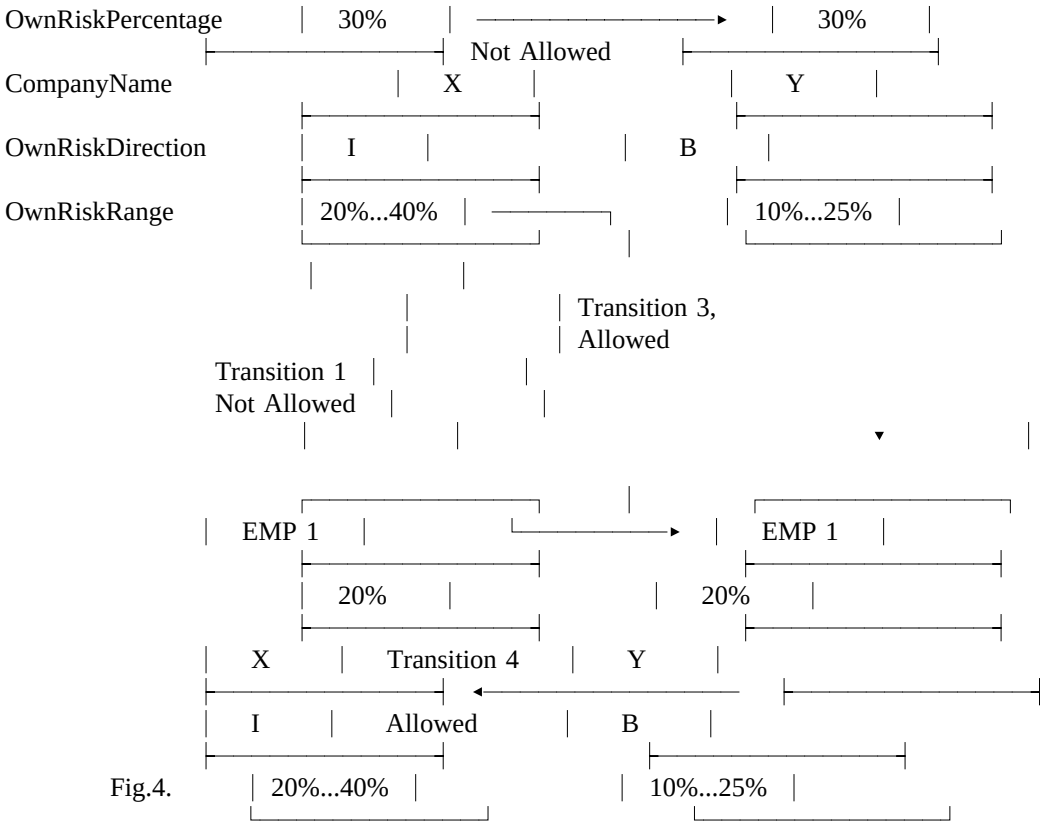


Fig.4.

Transition 1, a decrease of the own risk percentage is not allowed. A check of attribute ORD in the relation of the contracttypes is sufficient to determine this.

If the employee finds another job, with a company that also has a contract with the insurance company, the own risk percentage has to be checked in the relation of the contracttypes (transition 2). In this case the transition is not allowed due to the database constraint db6.

If however the employee does not only changes his/her own risk percentage but also finds another job (transition 3) this is allowed.

Both updates at the same time are allowed, one update at the time
(transitions 1 and 2) will be rejected. This makes it necessary to update the values ORP and Company in one transaction and to check the integrity rules at the moment of commit.

Notice that transition 4 is allowed.

V. THE EI-CASE IN THE DISTRIBUTED ENVIRONMENT.

The EI-Case can be distributed in many ways: one example will be given. The purpose of this distributed schema is that some of the complicating factors for distributed situations are involved: horizontal and vertical fragmentation is used and there is some replication. Furthermore there are local as well as global transactions.

A. The distributed datamodel.

The insurance company has divided the country in a couple of regions, each with their own office. Certain tasks, mainly marketing, sales and after sales are done from these regional offices. There is one central office with a computer center that has some other tasks as well.
The reference architecture for distributed databases of Ceri & Pelagatti [CePe] is used to design this

distributed EI-Case.

First, the global relations are split into fragments.

Contracttype is not split.

Company is split horizontally on basis of the region: companies with different regions in different fragments.

Contactperson is split horizontally. The criterium is the region of the company for which this person is the contactperson.

Employee is split horizontally and vertically as described below.

Certain attributes of Employee are not relevant in the regional offices, but for the administration in the computer center all attributes are of interest. Therefore the relation of employees is split vertically.
Due to the nature of the questions the regional offices receives, the next five attributes are necessary there:

Enr, Ename, ORF, Tdate and Cname.

This part of the relation Employee will be called the 'left' part.
The criterium for the horizontal split is the same as for the relation company: the region of the company of the employee. Every regional office will have a horizontal slice of the left part of the relation Employee.

After the fragmentation, the fragments are allocated to the different sites.

Contracttype will be allocated at all sites: the regional offices and the central site. The fragments of Company will be allocated to the corresponding regional offices. The complete relation will also be maintained at the central site. The fragments of Contactpersons will be allocated to the corresponding regional offices. The central site will not have any part of this relation. The fragments of the left side of Employee will be allocated to the corresponding regional offices. The complete relation will also be maintained at the central site.

A summary of the distributed relations:

In the regional offices the complete relation Contracttype and horizontal fragments of Company, Contactperson, the left part of Employees are maintained. The criterium for the horizontal splits is the region of the company. In the central site the complete relations Contracttype, Company and Employee are kept.

B. The consequences of this distributed schema for the constraints.

Certain tuples are registered at several sites. It is assumed that the Distributed DBMS maintains the consistency of the several copies. The relevant constraints have to be checked for one of the copies, the integrity constraints for the other copies are enforced by the consistency of the update. It is usually the most efficient to check the integrity at the site where the update is submitted, but sometimes it can be more efficient to check a constraint at another site. In what follows we will not go into detail only mention a few problems.

1. The consequences for the static non-update constraints.

Attribute constraints

No complications

Tuple constraints

The only two tuple constraints of the EI-Case are in the relation Employee, that is split vertically. The involved attributes are, for both constraints, in the two different vertical fragments. If these constraints are enforced at the regional office where the update is submitted, the central site has to be approached for this check, as one of the involved attributes is only centrally registered. This integrity check can also be done at the central site and the update propapated to the regional site where the update was submitted.

Relation constraints

A check on uniqueness should be done centrally. If not done centrally all the regional sites have to be approached where fragments are allocated. This applies for the relations Company, Contactperson and Employee, because these are horizontally fragmented.

The other relation constraints, referential integrity within one relation, general relation constraints and specific relation constraints all occur in relations that are split horizontally. They do not cause extra complications in the distributed environment as the constraints apply only to one fragment: the tuple where is referred to is present in the same fragment tas the tuple that contains the foreign-key.
A similar complication can occur if the relation was fragmented vertically and the vertical slices were maintained at different sites. The enforcement of the constraints could cause complications, e.g. if the referencing foreign key was maintained in another site than the primary key.

Database constraints

The referential integrity rules between tuples in different relations,
db1 and db2, can be cheked in one site. This is a consequence of the fact that for both constraints the participating relations are split on the same criterium.

For the cyclic referential integrity between the relations Company and Contactperson, db3 and db4, there are also no extra complications in the distributed environment because the criterium on which the relations are split are equal.

Referential integrity however can cause a check in another site: if the relation that contains the primary key where is referred to is situated in another site. This situation will often be avoided in the design phase because if operational, this would demand relative expensive communications on the network.

The database constraint db5, concerning area codes of telephone numbers, does not cause any extra complications. Db6, the own risk percentage of an employee should be in the range that belongs to the contracttype of the contract of his employer, can checked in the relevant regional office or in the central site.

2. The consequences for the update-constraints.

Relation Contracttype.

The delete rule restricted implied you cannot delete the description of a contracttype if there still is a company with this type of contract.
Just one check in the central site is necessary or all the regional sites have to be approeached in order to determine this.

The update rule restricted implied it is not allowed to update the contracttype identification if there is a company with this type of contract. Just like the previous delete rule, this involves one check in the central site or checks in all regional sites.

Relation Contactperson.

The delete rule (only for the main contactpersons).
For every company there should be at least one contactperson. This gives no complications in the distributed environment: the check involves only one regional site.

The update rule.
If a normal contactperson becomes main contactperson for a company, this involves four updates as described for the non-distributed environment.
Because all the relations involved are present in the relevant regional site there are no complications in the distributed case.

The insert of a contactperson for a company does not cause any complications, because the two relations involved are in one site.

Relation Company.

Two different situations were distinguished for the delete of a company, in both situations there are no extra complications in the distributed environment.

An update of the name of the company can be necessary if the company is bought by another company. Two situations were distinguished, the first one gives complications in the distributed environment, the second one not.

In the first situation the buying company also has a contract with the insurance company. The company that has been bought is deleted, the contactpersons of this company are deleted, but all its employees remain registered. The value of their field Cname is updated to the name of the buying company.

This rule gives an extra complication in the distributed environment if the new company is registered in another regional office. The old company and its contactpersons can be deleted but the employees have to be transferred to another site.

Two situations were distinguished for the insert of a company. Although
inserts at the regional office and in the central site will take place there are no real complications.

Relation Employee.

It is possible that an employee finds another job (update of Cname) at a company also has a contract with the assurance company the employee can remain registered. This can cause a transfer of the registration of the employee of one regional office to another regional office. It also involves an update in the central site.

Insert rule.
An employee can only be inserted if the related company is present in the relation Company. No extra complication in the distributed environment.

3. The consequences for the dynamic constraints.

The dynamic attribute constraint, a potential client can become a new client, a new client can become a stable client etc. does not cause any complications.

Due to the dynamic databaseconstraint an employee cannot freely update his or her own risk. Because the description of the contract is present at the site where the employee is registered this gives no further complications.

If however the employee also finds another job two updates take place and the integrity checks had to be done at the moment of commit of the transaction. In the distributed environment more than one site will be involved in the transaction: the site of the old company, the site of the new company and the central site.
The description of the employee will be transferred to the site of his/her new employer, and the description in the central site has to be updated also.

VI. PARTIAL QUESTIONNAIRE FOR DISTRIBUTED DBMS'S.

A. Introduction.

The purpose of the questionnaire is to investigate the functional capacities of a distributed DBMS. The functionalities can be divided into two groups: to what extent does a DBMS offer the features expected from a distributed DBMS and more general functional possibilities.
For the first group of functionality eight of the already well known 'Twelve rules of Chris Date' can be used. The aspects of distributed DBMS's that are not completely covered in the rules of Date can be

addressed in analogy with the questionnaire for non-distributed systems of IDT-Holland.

The first eight rules of Date determine to what extent a user can approach the distributed system as if it was not distributed at all. To test the support of a distributed DBMS for these rules a couple of questions can be asked for each rule. Attention is given in these questions to the next four aspects:

Retrieval, Update, the handling of Constraints and Datamanagement.

If in a rule not all these four aspects are relevant, no questions will be asked on the irrelevant aspects. The most simple form for distributed DBMSs is that, where the systems in the participating sites are equal, even if the computers and/or operating systems are not the same. This situation is called homogeneous and the presented questonnaire is for this homogeneous situation. Not all the functional aspects are covered, only those in direct relation to the first eight rules of Date.

B. The first eight rules of Date in the homogeneous situation.

We assume that the information concerning region A is maintained in site A,
concerning region B in site B etc..

Rule 1: Local autonomy.

The purpose of this rule is to check to what extent a node can operate autonomously.

1.1 R Can a transaction that is submitted in site A and only reads data in site A be executed successfully even if communicating with any other site is impossible?

test Read the description of one contactperson in a regional office, while communicating with any other site is impossible.

1.2 U Can a transaction that is submitted in site A and only updates a dataitem in site A be executed successfully even if communicating with any other site is impossible?

test Update the description of one contactperson in a regional office,
while communicating with other sites is impossible.

1.3 C Can a constraint in which only one site is involved be checked without any communication to another site?

test Increase the number of contactpersons in one site to 6, while communicating with other sites is impossible.

1.4 D Can the local transaction manager react autonomously? For example can it undo a rejected transaction?

test Update the main contactperson of a company, but let the new contactperson not be a existing contactperson of that company, but a contactperson of another company. Again communicating with other sites is impossible.

Rule 2: Independence of a central node.

In this rule it will be checked whether a transaction that involves two or more nodes can be executed successfully if other sites, e.g. the central site cannot be approached.

2.1 R Can a transaction that is submitted in site A and only reads data in site B be executed successfully while communicating with the central node is impossible?

test Read in site B the description of all the companies that belong to region A. (No communi-

cation with the central site.)

2.2 U Can a transaction that is submitted in site A and updates a dataitem in site B be executed successfully while communicating with the central site is impossible?

test Submit in site A an update for the attribute Pdesr of contactperson 'De Groot' in site B.

2.3 C Can a constraint in which more than one site is involved be checked without any communication to the central site?

test Submit in site A an update for the name of the contactperson 'Jansen' in Region B. (A uniqueness constraint)

2.4 D Can the transaction manager react without communicating to a central site? For example can it undo an unsuccessful transaction?

test Update in one transaction the primary keys of two contactpersons in two different sites to the same value. This transaction should be rejected because of a logical reason.

Rule 3: No planned shut-downs.

3.1 Is it necessary to stop the operations in other sites if a new version is installed in a certain site?

3.2 Is it necessary to stop the operations in other sites if a new node is added to the network?

3.3 Is it necessary to stop the operations in other sites if in a certain node alterations are made to the logical datamodel?

test This rule seems quit difficult to test. A relative simple test for 3.3 is adding a column to a relation.

Rule 4: Location-transparency.

In this rule it will be determined to what extent a user has to know in which node certain data is registered. The purpose is not yet to test whether the system is able to maintain replicated data at several sites transparent for the user, therefore we will only look at data that is not replicated. In the EI-Case the relation Contactpersons is not replicated.

4.1 R Can a retrieval query submitted from a certain site access data at other sites without having to indicate where that data is located.

test What are the names of the contactpersons for companies of type 'Financial'.

Remark: If a retrieval is successful, an update for the similar situation will not cause many problems (no replication complications yet). In the next question and test, a field is updated that determines the site where a tuple is stored. The issue is whether the updated tuple will be transferred to its new site.

4.2 U Does an update of the value of the attribute that determines the site of a tuple automatically implies the transfer of this tuple to the new site?

test Update the company of a contactperson.

Rule 5: Fragmentation-transparency.

Th purpose of this rule is to determine to what extent a user has to know in which fragment certain data is maintained. We do not want to test whether the system is able to maintain replicated data at several sites transparent for the user: therefore initially we would only look at data that is not replicated. This will result

in the same question and tests as for location-transparency. Therefore: no extra questions and tests for this rule

Rule 6: Replication-transparency.

The purpose of this rule is to determine to what extent a user has to know whether certain data is maintained in more than one node, in other words, whether the system is able to maintain replicated data at several sites transparent for the user.

A simple update:

6.1 U If a user updates a dataitem that is stored at several sites, does it has to be indicated at which site the data is kept?

test Change the descriptions of the companies of the type 'Hospital' into 'Medical'.

Updates that involve transfers of tuples to other fragments:

6.2 U Does an update of the value of the attribute that determines the site of a tuple automatically lead to the transfer of this tuple to the new site?

test Update the region of a company.
Remark: it has to be checked that the new region is an existing one.

6.4 U Does an update of the value of the attribute that determines the site of a tuple automatically implies a transfer of this tuple to the new site even if this involves an integrity check?

test An employee is located in the region where its employer is situated, so via a referential integrity constraint. If the employee finds another job at a company that also has a contract with the insurance firm, but that is located in another region, will the employee be transferred automatically to the new site?

Rule 7: Support for distributed query-processing.

The purpose of the questions in this rule is to determine wether the system offers specific features for the optimization of queries in a distributed environment.

7.1 Does the global optimizer supports processing on parallel processors.

7.2 Does the global optimizer use information of the system load in the various sites in order to process a part of the query there where the system load is relatively low.

7.3 Does the global optimizer use information of the local data-access possibilities?

Rule 8: Support for distributed transaction-management.

8.1 Does the system support the two-phase commit protocol?
If not what then?

8.2. Does the system Prevent or Detect global deadlock?
If so, does it solve these situations automatically?

C. A structure for a complete questionnaire.

The last four rules of Chris Date indicate on which configurations the distributed database can be made

operational, in other words, they determine what type of heterogeneous systems are supported.

After it has been established on what platforms heterogeneous processing is supported, the first eight rules can be 'copied': one set for each of the several heterogeneous environments. In a complete questionnaire for distributed database management systems the following subjects will have to be addressed:

1 The first eight rules of Date in the homogeneous situation.
2 The last four rules of Date.
3 The first eight rules of Date in a heterogeneous situation.
4 The anologon of the non-distributed questionnaire of IDT-Hollland.

The third part may be replicated to be able to describe the functionalities for several heterogeneous situations. The first part has been presented.
The next three parts can be subject of future research.

References

[CePe] Stefano Ceri and Giuseppe Pelagatti, Distributed Database Design. Singapore, Mc. Graw-Hill, 1985.

[DaWh] Chris J. Date, What is a distributed Database System? Relational Database Writings 1985-1989. Reading, Mass.: Addison Wesley 1990.

[MaNe] Saskia C. van der Made-Potuijt, The necessity for check-on-commit in the protection of the integrity of a database, 1989, Department of Computer Science, Erasmus University Rotterdam, Report EUR-CS-89-05.

[MGMo] Saskia C. van der Made-Potuijt en L.P.J. Groenewegen, Modelling a transaction manager using parallel decision processes, 1990, Department of Computer Science, Erasmus University Rotterdam, Report EUR-CS-90-07.

[NGIP] De Database Club van de Sectie Informatiesystemen van het Nederlands Genootschap voor Informatica. Het praktische gebruik van relationele systemen gepresenteerd. Amsterdam 1988.

[JGTr] Jim Gray, The transaction Concept: Virtues and Limitations. Readings in Database Systems, Morgan Kaufmann Publishers Inc. 1988.

Contents.