COURSEWORK REPORT

DreamHome Case Study

Semester A
MSc Software Engineering
2003-2004

Rev. [1.0.00]

Department of Computing
London Metropolitan University

For:

Dr. Farhi Marir (SB03G – Distributed Database Systems)

Prepared by:

Mr. Fahad Habib (ID: 02029881)
fah086@londonmet.ac.uk
Mr. Umer Ejaz Butt (ID: 03028264)
ueb001@londonmet.ac.uk

Confidential and Proprietary Document
DOC: Dev/SDD/2612/4002 (VER: 1.0.00)

ACKNOWLEDGEMENTS

We are thankful to Allah Almighty for all of His Blessings. Without Him, we are nothing. Our acknowledgement goes to Dr. Farhi Marir for his help during this coursework. We would also like to thank all of our friends who helped us during this coursework and our parents for their continuous prayers for us.

Sincerely,

Fahad Habib
Umer Ejaz Butt

Copyright © MSc Software Engineering 2003-2004

TABLE OF CONTENTS

Copyright © MSc Software Engineering 2003-2004

Copyright © MSc Software Engineering 2003-2004

LIST OF FIGURES AND TABLES

Copyright © MSc Software Engineering 2003-2004

1. INTRODUCTION

Design is a meaningful engineering representation of a product that is to be built. Software design sits at the technical kernel of software engineering and is applied regardless of the software process model that is used. Beginning once software requirements have been analyzed and specified, software design is the first of three technical activities (design, code generation, and tests) that are required to build and verify the software. Each activity transforms information in a manner that ultimately results in validated computer software.

Each of the elements of the analysis model provides information that is necessary to create the four design models required for a complete specification of design. Software requirements, manifested by the data, functional, and behavioural models, feed the design task. Using one of a number of design methods, the design task produces a data design, an architectural design, an interface design, and a component design.

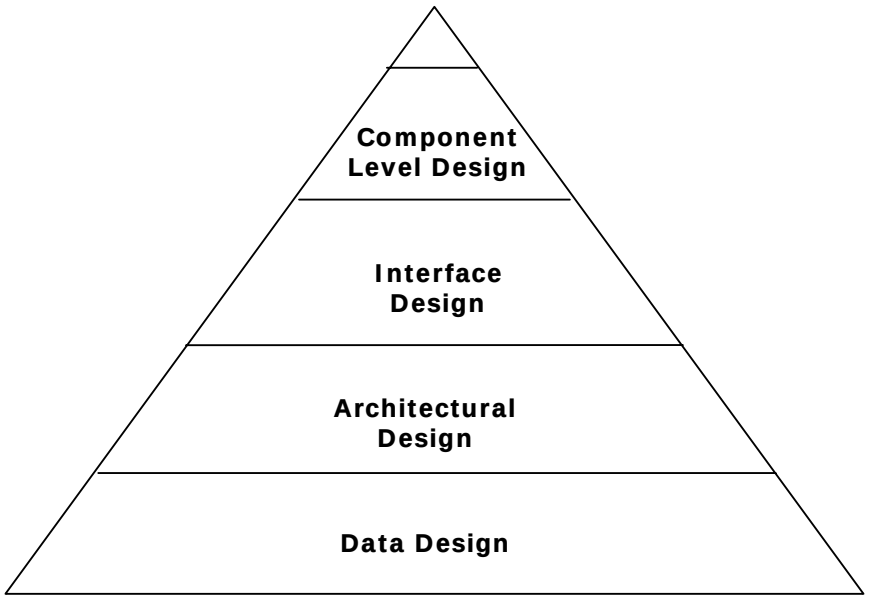


Fig 01: The Design Model

1.1 Problem Statement

1. DreamHome has branch offices in cities throughout the United Kingdom. Each branch office is allocated members of staff including a Manager to manage the operations of the office.

2. Members of staff with the role of Supervisor are responsible for the day-to-day activities of an allocated group of staff called Assistants. Not all members of staff are assigned to a Supervisor.

Copyright © MSc Software Engineering 2003-2004

3. Each branch office offers arrange of properties for rent. The management of a property is assigned to a member of staff whenever it is rented out or requires to be rented out.

4. The details of property owners are also stored. There are two main types of property owner: private owners and business owners.

5. DreamHome refers to members of the public interested in renting property as clients. To become a client, a person must first register at a branch office of DreamHome.

6. When a property is rented out, a lease is drawn up between the client and the property.

7. When required, the details of properties for rent are advertised in local and national newspapers.

1.2 Requirements/Constraints

- Each branch has a Manager to manage the operations.
- A bonus payment, based upon the Manager's performance in the property for rent market, is to be maintained.
- A maximum of ten Assistants are allocated to a single Supervisor.
- Not all members of staff are assigned to a Supervisor.
- Each branch office offers a range of properties for rent.
- The management of a property is assigned to a member of staff whenever it is rented out or requires to be rented out.
- A member of staff may manage a maximum of hundred properties for rent at any one time.
- To become a client, a person must first register at a branch office of DreamHome.
- A client may request to view property.
- A client may view the same property only once on a given date.
- A client may hold a lease associated with a given property for a minimum of three months to a maximum of one year.

Copyright © MSc Software Engineering 2003-2004

2. PROPOSED SYSTEM DESIGN

We have proposed Client/Server architecture for the DreamHome Scenario. We will give a brief description of the Client/Server architecture and the system configuration for DreamHome. The client machine and the server machine would have a specific hardware and software configuration. To illustrate this point we will assume that the client machine is a Work Station and has the following configuration.

COMPUTRITION Hospitality Suite® Traditional Client/Server Architecture - Client Workstation

Recommended Hardware Specifications

The purpose of this document is to establish a recommended baseline for configuring client workstations to operate with Hospitality Suite® in Traditional Client/Server architecture.

Traditional Client/Server Architecture – Client Workstation	
Hardware Item Type	**Description**
Processor:	Intel Pentium III 850 MHz PC or faster compatible system
RAM:	512 MB memory (RAM)
Disk Storage:	1 GB hard disk
CD-ROM:	CD-ROM Drive (4X Minimum)
Video Card:	2 MB SVGA Color Display Adapter (Support 16-bit color at 800x600 screen resolution)
Monitor:	15” SVGA Monitor
I/O Devices:	Standard 101/104 U.S. Keyboard, Mouse
Oracle Client Version:	Oracle Client V 8i (8.1.7) (8.1.7.3 or greater for Windows XP Professional)
Operation System:	Windows NT 4.0 Workstation w/SP6a, Windows 2000 Professional w/SP2, Windows XP Professional (8.1.7.3 or greater Oracle Client Version)
Required Application Support Software:	Microsoft Internet Explorer version 5.5 with SP2 or GREATER *Required to view Quick reports with Hospitality Suite Adobe Acrobat Reader 4.0 or GREATER *Required to access on-line Hospitality Suite Manual

Copyright © MSc Software Engineering 2003-2004

NOTE: Although Computrition Client/Server products will operate on slower machines, this recommendation is to ensure optimum system operation. If a slower machine is used, the performance of the Computrition software may be undesirable. These recommended specifications should serve as a minimum recommendation to ensure reasonable performance of the Computrition software. Although every attempt has been made here to offer readily available equipment configurations, frequent changes/advances in technology and price/performance may render this particular configuration economically obsolete. Please contact Computrition before purchasing your specific hardware platform to ensure you have the current minimum recommended specifications.

Document Version: 1.07

Last Revision Date: July 15, 2003

Recommended Hardware Specifications - Fat Client Workstation - version 1.07.doc

Hospitality Suite Version: 13.x

URL: http://www.hospitalitysuite.com/HSHardwareSpecs/MinimumHardwareSpecs/Fat-ClientWorkstation/RecommendedHardwareSpecifications-FatClientWorkstation-version1.07.pdf

2.1 Architectural Design

The client/server software architecture is a versatile, message-based and modular infrastructure that is intended to improve usability, flexibility, interoperability, and scalability as compared to centralized, mainframe, time sharing computing **[Schussel 96, Edelstein 94].**

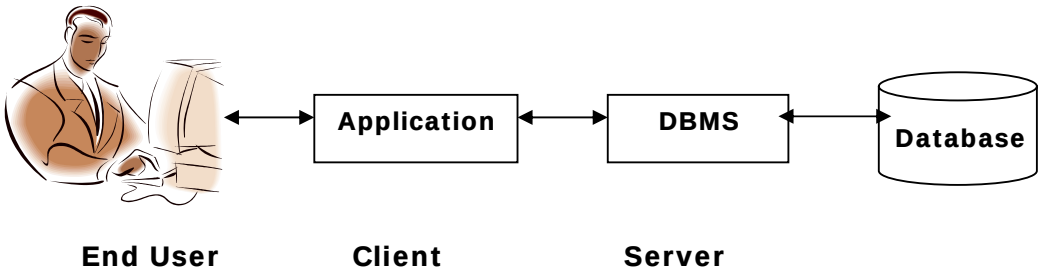


Fig 02: The Client/Server Architecture Emerged

2.1.1 Features

- Introduction of database server
- Improves multi-user updating through a GUI front end to a shared database
- Reduced network traffic by providing a query response rather than total file transfer
- Remote Procedure Calls (RPCs) or standard query language (SQL) statements are typically used to communicate between the client and server

Copyright © MSc Software Engineering 2003-2004

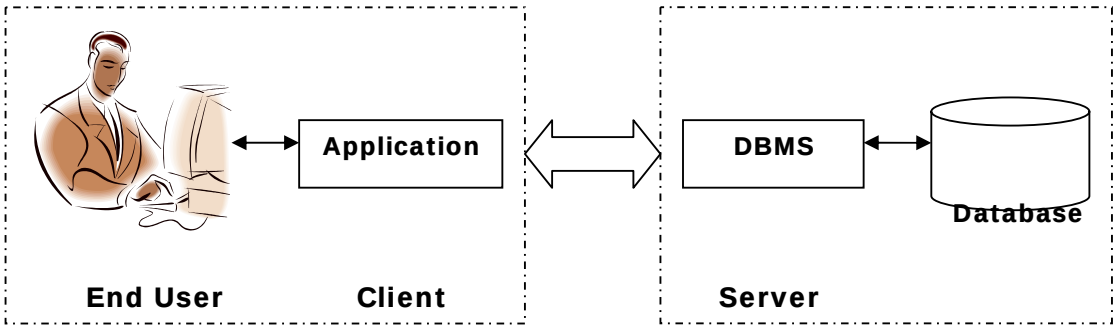


Fig 03: The Distributed Client/Server Model

2.1.2 Algorithmic Model

Locate the Resource (Client side and the Web Server)

Client Request Resource

Web Server generates Response

End

Establish Connection

Create Connection

Create Command Object

Set Command Properties

Connect to Database

End

Retrieve Data on Server (Occurs between Web Server and Database Server)

Define Query into Command

Execute Command

Destroy Command Object

End

Request Data (Web Server and Client Side)

Loop required number of records through record set

While not End of Requirements

Write Response to Client

Move to next Record, End of Loop

Destroy Record Set to free Web Server resources

End

Copyright © MSc Software Engineering 2003-2004

2.1.3 Why Three-Tier

- To give workers better access to data by improving application performance, scalability, and reliability
- To respond more quickly to customer requirements through faster development and deployment of large applications
- To connect existing business processes to the Web
- To simplify software distribution and maintenance
- To establish a consistent software infrastructure
- To lay a foundation for distributed objects
- Three tier with an ORB architecture
  - Client/server systems using technologies that support distributed objects (CORBA, COM/DCOM).
  - Industry is working on standards to improve interoperability between CORBA and COM/DCOM
- Benefits:
- Improve interoperability.
- Enhancing maintainability and adaptability of the system.
- The Object Management Group (OMG) has developed a mapping between CORBA and COM/DCOM that is supported by several products **[OMG 96]**.

2.1.4 Client/Server–A Special Case of Distributed Computing

Figure shows the interrelationships between distributed computing and client/server models. Conceptually, client/server model is a special case of distributed-computing model.

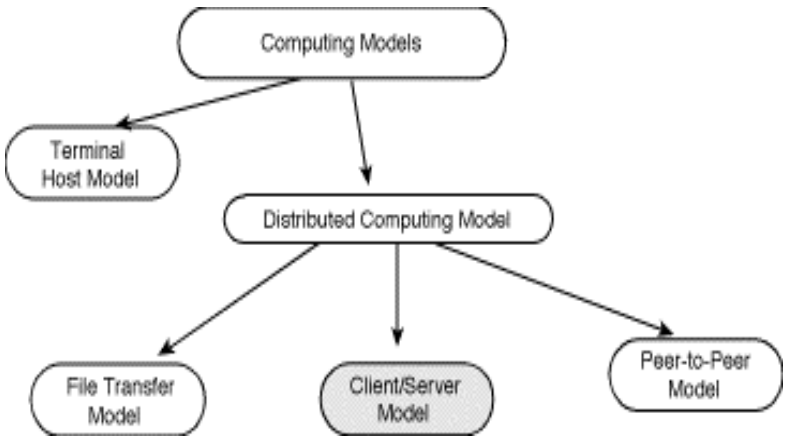


Fig 04: Interrelationships between Computing Models

A Distributed Computing System (DCS) is a collection of autonomous computers interconnected through a communication network to achieve business functions. Technically, the computers do not share main memory so that the information cannot be

Copyright © MSc Software Engineering 2003-2004

transferred through global variables. The information (knowledge) between the computers is exchanged only through messages over a network.

The restriction of no shared memory and information exchange through messages is of key importance because it distinguishes between DCS and shared memory multiprocessor computing systems. This definition requires that the DCS computers are connected through a network that is responsible for the information exchange between computers.

Distributed computing can be achieved through one or more of the following:

- File transfer model
- Client/server model
- Peer-to-peer model

The C/S model is state of the market and state of the practice for distributed computing at the time of this writing. C/S model, as stated previously, allows application processes at different sites to interactively exchange messages and is thus a significant improvement over the file transfer model. Initial versions of C/S model utilized the remote procedure call paradigm that extends the scope of a local procedure call. At present, the C/S model is increasingly utilizing the distributed objects paradigm that extends the scope of local object paradigm (i.e., the application processes at different sites are viewed as distributed objects).

Copyright © MSc Software Engineering 2003-2004

3. DESIGN

3.1 Entity Relationship Diagram (ERD)

Following figure shows the Entity Relationship Diagram with Primary keys and relations between the entities for our case study.

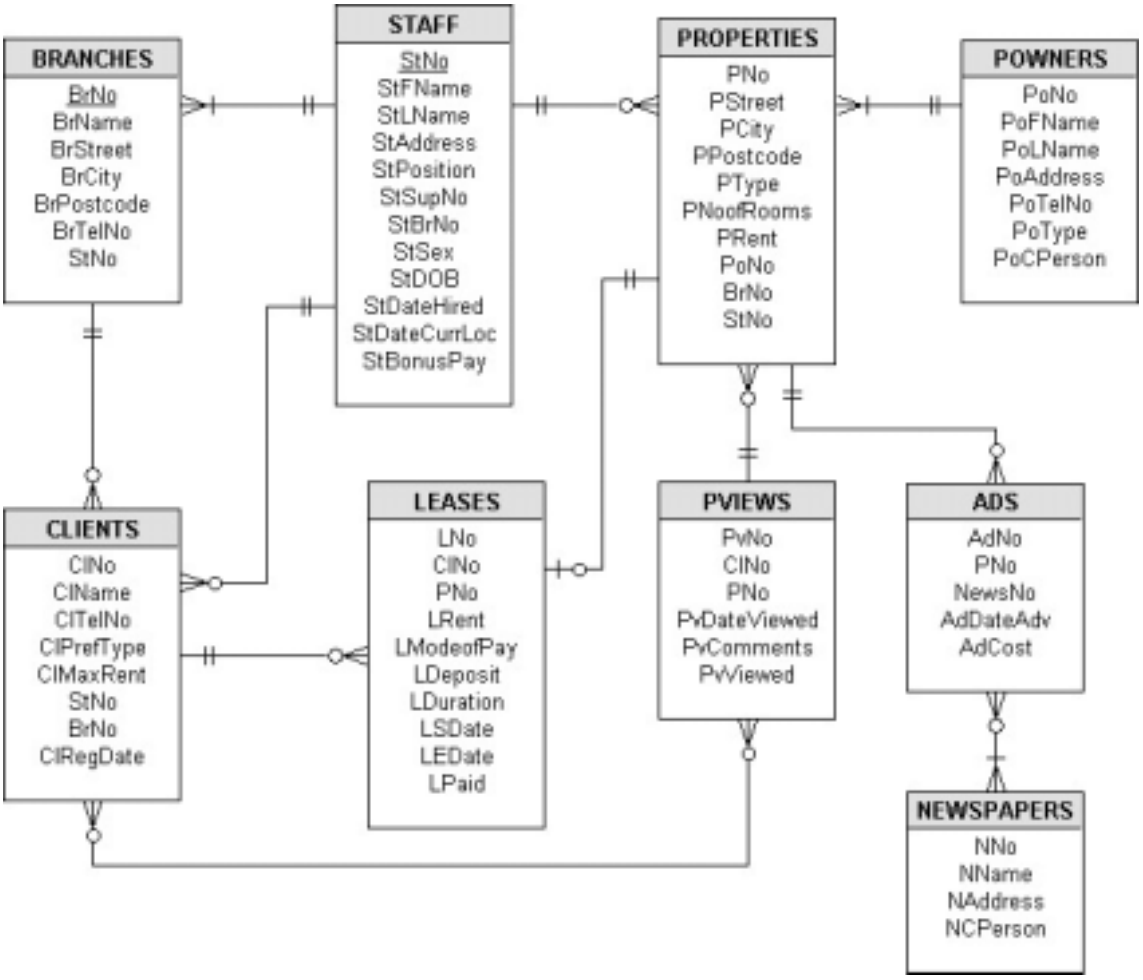
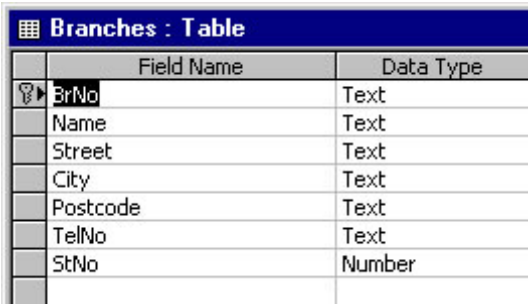


Fig 05: ERD for DreamHome Case Study

Copyright © MSc Software Engineering 2003-2004

3.2 Data Design

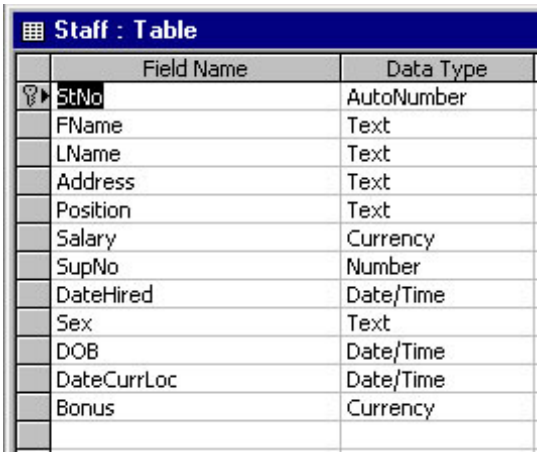
Following figures show the data design view of our database.



Branches : Table

Field Name	Data Type
BrNo	Text
Name	Text
Street	Text
City	Text
Postcode	Text
TelNo	Text
StNo	Number

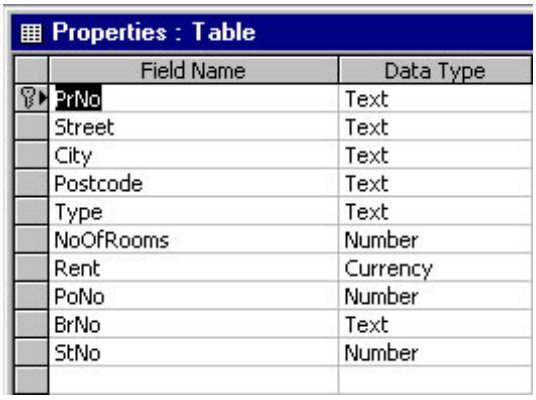
Fig 06: Branches Table



Staff : Table

Field Name	Data Type
StNo	AutoNumber
FName	Text
LName	Text
Address	Text
Position	Text
Salary	Currency
SupNo	Number
DateHired	Date/Time
Sex	Text
DOB	Date/Time
DateCurrLoc	Date/Time
Bonus	Currency

Fig 07: Staff Table

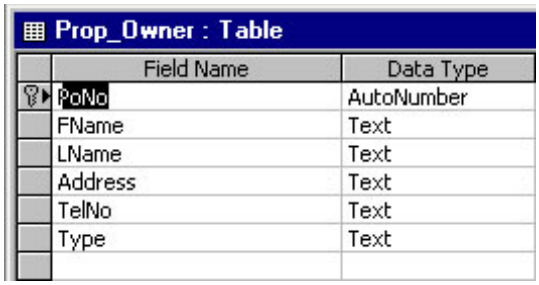


Properties : Table

Field Name	Data Type
PrNo	Text
Street	Text
City	Text
Postcode	Text
Type	Text
NoOfRooms	Number
Rent	Currency
PoNo	Number
BrNo	Text
StNo	Number

Fig 08: Properties Table

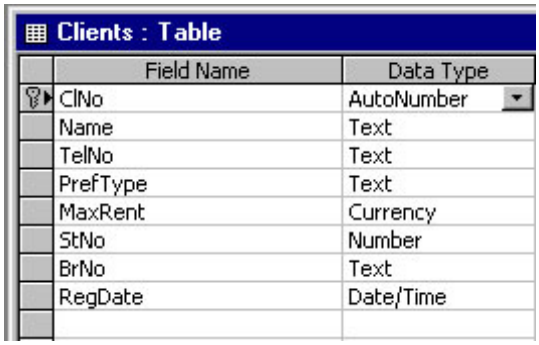
Copyright © MSc Software Engineering 2003-2004



Prop_Owner : Table

Field Name	Data Type
PoNo	AutoNumber
FName	Text
LName	Text
Address	Text
TelNo	Text
Type	Text

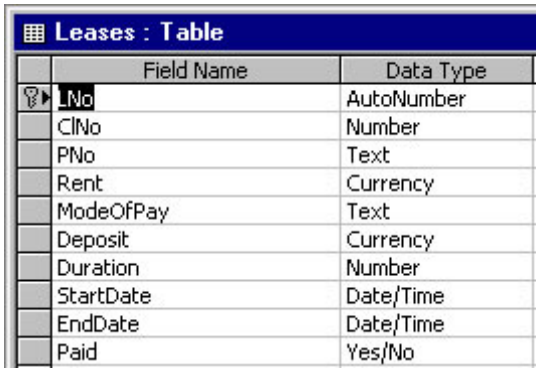
Fig 09: Prop_Owner Table



Clients : Table

Field Name	Data Type
ClNo	AutoNumber
Name	Text
TelNo	Text
PrefType	Text
MaxRent	Currency
StNo	Number
BrNo	Text
RegDate	Date/Time

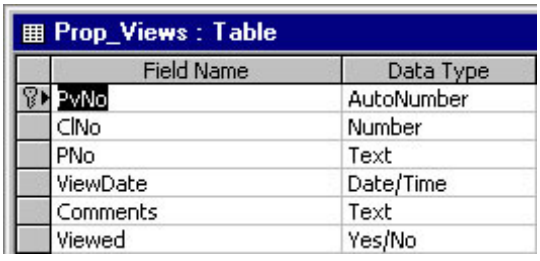
Fig 10: Clients Table



Leases : Table

Field Name	Data Type
LNo	AutoNumber
ClNo	Number
PNo	Text
Rent	Currency
ModeOfPay	Text
Deposit	Currency
Duration	Number
StartDate	Date/Time
EndDate	Date/Time
Paid	Yes/No

Fig 11: Leases Table



Prop_Views : Table

Field Name	Data Type
PvNo	AutoNumber
ClNo	Number
PNo	Text
ViewDate	Date/Time
Comments	Text
Viewed	Yes/No

Fig 12: Prop_View Table

Copyright © MSc Software Engineering 2003-2004

Newspapers : Table

	Field Name	Data Type
🔑	NpNo	AutoNumber
	Name	Text
	Address	Text
	ContactPerson	Text

Fig 13: Newspapers Table

Adverts : Table

	Field Name	Data Type
🔑	AdNo	AutoNumber
	PNo	Text
	NpNo	Number
	DateAdvert	Date/Time
	Cost	Currency

Fig 14: Adverts Table

3.3 User Interface Design

ASP (Active Server Pages) will be used as the front-end and Microsoft Access will be used for the database and act as the back-end. The detailed interface design will be submitted at the time of implementation.

Fig 15: User Interface for DreamHome

Copyright © MSc Software Engineering 2003-2004

4. FRAGMENTATION

After completing analysis and design of the relational database for the case study, it's now possible to make fragmentation according to certain criteria. We have provided with the user requirements and applications required to run on this database. Based on those requirements, we propose following assumptions according to the case study:

4.1 Assumptions

We have a sample data of 4 branches in different cities, 2 in London and 1 each in Glasgow and Aberdeen. Each branch will have a manager and 3 Staff members working and will have 10 Properties. 4 Clients, 3 Property Owners, 4 Newspapers, 5 Ads, 2 Leases and 4 Property Views records will be entered to simulate our data distribution.

Following are the data from the database:

Branches						
BrNo	**Name**	**Street**	**City**	**Postcode**	**TelNo**	**StNo**
B001	The Dreamhome City	23 High Street, Aldgate	London	EC1 6AC	02075134213	Mark
B002	The DreamHome South	56 Rye Lane, Peckham	London	SE1 8HB	02076237856	Tim
B003	The DreamHome Glasgow	65-69 King Street	Glasgow	GL2 4DE	01347843266	Johnny
B004	The DreamHome Aberdeen	12 Superior Road	Aberdeen	AB3 1UR	01825678890	Nancy

Properties									
PrNo	**Street**	**City**	**Postcode**	**Type**	**NoOfRooms**	**Rent**	**PoNo**	**BrNo**	**StNo**
PG1		London		House	4	£500	Tom	The Dreamhome City	Anna
PG10		Aberdeen		Flat	2	£275	Clive	The DreamHome Aberdeen	Justin
PG2		London		Flat	2	£200	Tom	The Dreamhome City	Willam
PG3		London		Flat	3	£250	Tom	The Dreamhome City	Willam
PG4		London		Office	4	£600	John	The Dreamhome City	Anna
PG5		London		Flat	3	£200	John	The DreamHome South	Warwick
PG6		London		House	3	£400	John	The DreamHome South	Michelle
PG7		London		Office	5	£700	John	The DreamHome South	Michelle
PG8		Glasgow		Flat	2	£250	Clive	The DreamHome Glasgow	Steve
PG9		Aberdeen		Flat	3	£350	Clive	The DreamHome Aberdeen	Justin

Prop_Owner					
PoNo	**FName**	**LName**	**Address**	**TelNo**	**Type**
1	Tom	Cruise	123 Queen street London	02071239776	Private
2	John	Berners	435 Razor Road, London	02076542398	Business
3	Clive	Woodward	11 King William Street, Aberdeen	01873342004	Business

Copyright © MSc Software Engineering 2003-2004

Staff												
StNo	FName	LName	Address	Position	Salary	SupNo	BrNo	DateHired	Sex	DOB	DateCurrLoc	Bonus
2	Mark	William	123 South Road, Bermondsey	Manager	£400	n/a	B001	15/10/2002	Male	19/05/1969	15/10/2002	£0
3	John	Wright	419 Old Kent Road, Peckham	Supervisor	£300	Mark	B001	01/01/2003	Male	23/08/1978	01/01/2003	£0
4	Anna	Margaret	21 Holloway Road, Islington	Assistant	£200	John	B001	01/01/2003	Female	15/09/1979	01/01/2003	£0
5	Willam	Blair	65 Tower Bridge Road, Tower Hamlets	Assistant	£200	John	B001	01/02/2003	Male	23/04/1974	01/02/2002	£0
6	Tim	Ray	72 High Street, Whitechapel	Manager	£400	n/a	B002	01/03/2003	Male	14/02/1964	01/03/2003	£0
7	Jason	McGrath	12 North Street	Supervisor	£300	Tim	B002	15/03/2003	Male	21/08/1976	15/10/2002	£0
8	Warwick	Henry	9 Denmark Hill	Assistant	£200	Jason	B002	01/04/2003	Male	04/10/1974	01/04/2003	£0
9	Michelle	Kamber	21 Peckham Road	Assistant	£200	Jason	B002	01/04/2003	Female	10/05/1977	01/04/2003	£0
10	Johnny	Blade	65 Tower Bridge Road	Manager	£400	n/a	B003	01/01/2003	Male	11/07/1966	01/01/2003	£0
11	Graeme	Tom	78 Abbey Street	Supervisor	£300	Johnny	B003	15/03/2003	Male	21/08/1976	15/10/2002	£0
12	Steve	Wonder	106 Tooley Street	Assistant	£200	Graeme	B003	01/04/2003	Male	04/10/1974	01/04/2003	£0
13	Sarah	Johnson	174 Woodford Avenue	Assistant	£200	Graeme	B003	01/04/2003	Female	10/05/1977	01/04/2003	£0
14	Nancy	Elizabeth	137 Avondale Rise	Manager	£400	Tim	B004	01/06/2003	Female	18/02/1968	01/06/2003	£0
15	Matthew	Waugh	344 Cheap Street	Supervisor	£300	Nancy	B004	15/06/2003	Male	21/11/1975	15/06/2003	£0
16	Justin	Walker	43 Way-to-Go Road	Assistant	£200	Matthew	B004	15/06/2003	Male	08/04/1978	15/06/2003	£0

Clients							
ClNo	Name	TelNo	PrefType	MaxRent	StNo	BrNo	RegDate
1	Mike Ritchie		Flat	£300.00	John	The Dreamhome City	05/06/2003
2	William Cole		House	£500.00	Jason	The DreamHome South	01/07/2003
3	Harry Woods		Flat	£400.00	Graeme	The DreamHome Glasgow	10/07/2003
4	David Howard		Office	£800.00	Matthew	The DreamHome Aberdeen	10/06/2003

Leases									
LNo	ClNo	PNo	Rent	ModeOfPay	Deposit	Duration	StartDate	EndDate	Paid
1	Mike Ritchie	PG2	£200.00	Direct Debit	£100.00	24	01/07/2003	30/06/2005	Yes
2	David Howard	PG9	£350.00	Cash	£150.00	36	01/08/2003	31/07/2006	Yes
3	Harry Woods	PG8	£400.00	Cash	£200.00	36	01/09/2003	31/08/2006	No

Copyright © MSc Software Engineering 2003-2004

Newspapers			
NpNo	**Name**	**Address**	**ContactPerson**
1	The Sun		Any Person
2	Daily Mirror		Someone
3	Daily Telegraph		Really Someone
4	The Independent		Call me

Adverts				
AdNo	**PNo**	**NpNo**	**DateAdvert**	**Cost**
1	PG1	The Sun	01/06/2003	£100.00
2	PG5	Daily Telegraph	03/06/2003	£85.00
3	PG6	The Independent	08/06/2003	£70.00
4	PG8	Daily Mirror	15/06/2003	£80.00
5	PG9	Daily Telegraph	01/07/2003	£150.00

Prop_Views					
PvNo	**ClNo**	**PNo**	**ViewDate**	**Comments**	**Viewed**
1	Mike Ritchie	PG4	15/06/2003	Very good location, moderate rent and beautifully furnished property	Yes
2	Harry Woods	PG8	20/07/2003		Yes
3	David Howard	PG10	12/06/2003		No
4	William Cole	PG5	25/07/2003		Yes

Copyright © MSc Software Engineering 2003-2004

4.2 Horizontal Fragmentation (HF)

In horizontal fragmentation **[OV99]** we have two different ways of fragmenting tables. These are:

1. Primary Horizontal Fragmentation
2. Derived Horizontal Fragmentation

4.2.1 Primary Horizontal Fragmentation

For primary horizontal fragmentation **[OV99]**, we have some measures that can be used in fragmenting relations in distributed databases. These measures are known as Predicates. There are two types of predicates we can use in primary horizontal fragmentation. These are **[OV99]**:

- Simple Predicates
- Minterm Predicates

4.2.1.1 Application Requirements

Following applications are required to be accessed by the organization:

1. List the details of braches in a given city
2. Identify the total number of branches in each city

Following SQL commands can be written:

```
1. SELECT brno, name, address, postcode FROM branches WHERE city=“value”
2. SELECT COUNT(*) FROM branches GROUP BY city
```

Based on these requirements, following fragmentation scheme is suggested:

4.2.1.2 Simple Predicates

According to the requirements, the data required is frequently accessed within a specific city and not by other cities. So it is suggested to keep one branch data within that branch for quick and reliable data access.

With respect to application 1, we have set of simple predicates for *BRANCHES* as

BR: {CITY = “London”, CITY = “Glasgow”, CITY= “Aberdeen”}

Copyright © MSc Software Engineering 2003-2004

4.2.1.3 Minterm Predicates

For *BRANCHES* we have minterm predicates as

mb1: CITY = "London"

mb2: CITY = "Glasgow"

mb3: CITY= "Aberdeen"

Set of minterm predicates for *BRANCHES* can be defined as

M1 = {CITY = "London", CITY = "Glasgow", CITY= "Aberdeen",
CITY = "London" $\wedge \neg$CITY = "Glasgow", CITY = "London" $\wedge \neg$CITY= "Aberdeen",
= "London" $\wedge \neg$CITY = "Glasgow" $\wedge \neg$CITY= "Aberdeen"}

Some of the implications are

i1: br1 $\Rightarrow \neg$ br2 $\wedge \neg$ br3

i2: br2 $\Rightarrow \neg$ br1 $\wedge \neg$ br3

i3: br3 $\Rightarrow \neg$ br1 $\wedge \neg$ br2

So according to our assumptions and data that exist in our database, we left with following fragments, which are:

M1 = {CITY = "London", CITY = "Glasgow", CITY= "Aberdeen"}

Let's denote *BRANCHES* with BRAN then we can write the fragments in algebraic form;

BRAN1 = $\sigma_{\text{CITY = "London"}}$ (Branches)

BRAN2 = $\sigma_{\text{CITY = "Glasgow"}}$ (Branches)

BRAN3 = $\sigma_{\text{CITY = "Aberdeen"}}$ (Branches)

Hence the result of primary horizontal fragmentation of BRANCHES and PROPERTIES is:

$F_{BRANCHES}$ = {BRAN1, BRAN2, BRAN3}

BRAN1						
BrNo	**Name**	**Street**	**City**	**Postcode**	**TelNo**	**StNo**
B001	The Dreamhome City	23 High Street, Aldgate	London	EC1 6AC	02075134213	Mark
B002	The DreamHome South	56 Rye Lane, Peckham	London	SE1 8HB	02076237856	Tim

BRAN2						
BrNo	**Name**	**Street**	**City**	**Postcode**	**TelNo**	**StNo**
B003	The DreamHome Glasgow	65-69 King Street	Glasgow	GL2 4DE	01347843266	Johnny

Copyright © MSc Software Engineering 2003-2004

BRAN3						
BrNo	**Name**	**Street**	**City**	**Postcode**	**TelNo**	**StNo**
B004	The DreamHome Aberdeen	12 Superior Road	Aberdeen	AB3 1UR	01825678890	Nancy

Now we will check for the correctness of our fragments according to our assumption.

4.2.2 Derived Horizontal Fragmentation

A derived horizontal fragmentation is defined on a member relation of a link according to a selection operation specified on its owner **[OV99]**.

As our BRANCHES table fragmented horizontally into 3 fragments, so we will apply derived fragmentation on our PROPERTIES, CLIENTS and other tables coming in the chain, so that relevant data remain at one place.

4.2.2.1 For PROPERTIES table

With respect to BRAN1, BRAN2 and BRAN3, we will apply equi-join derived fragmentation on our properties table as:

$\Rightarrow$ PROP1 = σ ($_{\sigma \text{ CITY = "London" (Branches)}}$) PROPERTIES

$\Rightarrow$ PROP1 = σ ($_{BRAN1}$) PROPERTIES

$\Rightarrow$ PROP1 = PROPERTIES $\propto$ BRAN1

PROP1									
PrNo	**St**	**City**	**Postcode**	**Type**	**Rooms**	**Rent**	**PoNo**	**BrNo**	**StNo**
PG1		London		House	4	£500.00	Tom	The Dreamhome City	Anna
PG2		London		Flat	2	£200.00	Tom	The Dreamhome City	William
PG3		London		Flat	3	£250.00	Tom	The Dreamhome City	William
PG4		London		Office	4	£600.00	John	The Dreamhome City	Anna
PG5		London		Flat	3	£200.00	John	The DreamHome South	Warwick
PG6		London		House	3	£400.00	John	The DreamHome South	Michelle
PG7		London		Office	5	£700.00	John	The DreamHome South	Michelle

Similarly we have,

$\Rightarrow$ PROP2 = PROPERTIES $\propto$ BRAN2

$\Rightarrow$ PROP3 = PROPERTIES $\propto$ BRAN3

Copyright © MSc Software Engineering 2003-2004

PROP2									
PrNo	St	City	Postcode	Type	Rooms	Rent	PoNo	BrNo	StNo
PG8		Glasgow		Flat	2	£250.00	Clive	The DreamHome Glasgow	Steve

PROP3									
PrNo	St	City	Postcode	Type	Rooms	Rent	PoNo	BrNo	StNo
PG10		Aberdeen		Flat	2	£275.00	Clive	The DreamHome Aberdeen	Justin
PG9		Aberdeen		Flat	3	£350.00	Clive	The DreamHome Aberdeen	Justin

Similarly, we can apply derived horizontal fragmentation on:

- CLIENTS and STAFF directly on BRANCHES
- LEASES, PROP_VIEWS, ADVERTS, PROP_OWNERS through the link of PROPERTIES
- NEWSPAPERS through the link of ADVERTS

Therefore, after applying these derived fragmentation schemes, we will get;

4.2.2.2 Using BRANCHES fragments

CLIENTS fragments as

$\Rightarrow$ CLIENT1 = CLIENTS $\propto$ BRAN1

$\Rightarrow$ CLIENT2 = CLIENTS $\propto$ BRAN2

$\Rightarrow$ CLIENT3 = CLIENTS $\propto$ BRAN3

CLIENT1							
ClNo	Name	Tel	Type	MaxRent	StNo	BrNo	RegDate
1	Mike Ritchie		Flat	£300.00	John	The Dreamhome City	05/06/2003
2	William Cole		House	£500.00	Jason	The DreamHome South	01/07/2003

CLIENT2							
ClNo	Name	Tel	Type	MaxRent	StNo	BrNo	RegDate
3	Harry Woods		Flat	£400.00	Graeme	The DreamHome Glasgow	10/07/2003

CLIENT3							
ClNo	Name	Tel	Type	MaxRent	StNo	BrNo	RegDate
4	David Howard		Office	£800.00	Matthew	The DreamHome Aberdeen	10/07/2003

STAFF fragments as

$\Rightarrow$ STAFF1 = STAFF $\propto$ BRAN1

$\Rightarrow$ STAFF2 = STAFF $\propto$ BRAN2

$\Rightarrow$ STAFF3 = STAFF $\propto$ BRAN3

But for STAFF, its better to have vertical fragmentation first then apply horizontal to get better performance.

Copyright © MSc Software Engineering 2003-2004

4.2.2.3 Using PROPERTIES fragments

LEASES fragments as

⇒ LEASE1 = LEASES ∝ PROP1

⇒ LEASE2 = LEASES ∝ PROP2

⇒ LEASE3 = LEASES ∝ PROP3

LEASE1									
LNo	**ClNo**	**PNo**	**Rent**	**ModeOfPay**	**Deposit**	**Duration**	**StartDate**	**EndDate**	**Paid**
1	1	PG2	£200.00	Direct Debit	£100.00	24	01/07/2003	30/06/2005	Yes

LEASE2									
LNo	**ClNo**	**PNo**	**Rent**	**ModeOfPay**	**Deposit**	**Duration**	**StartDate**	**EndDate**	**Paid**
3	3	PG8	£400.00	Cash	£200.00	36	01/09/2003	31/08/2006	No

LEASE3									
LNo	**ClNo**	**PNo**	**Rent**	**ModeOfPay**	**Deposit**	**Duration**	**StartDate**	**EndDate**	**Paid**
2	4	PG9	£350.00	Cash	£150.00	36	01/08/2003	31/07/2006	Yes

PROP_VIEWS fragments as

⇒ PV1 = PROP_VIEWS ∝ PROP1

⇒ PV2 = PROP_VIEWS ∝ PROP2

⇒ PV3 = PROP_VIEWS ∝ PROP3

PV1					
PvNo	**ClNo**	**PNo**	**ViewDate**	**Comments**	**Viewed**
1	Mike Ritchie	PG4	15/06/2003	Very good location, moderate rent and beautifully furnished property	Yes
4	William Cole	PG5	25/07/2003		Yes

PV2					
PvNo	**ClNo**	**PNo**	**ViewDate**	**Comments**	**Viewed**
2	Harry Woods	PG8	20/07/2003		Yes

PV3					
PvNo	**ClNo**	**PNo**	**ViewDate**	**Comments**	**Viewed**
3	David Howard	PG10	12/06/2003		No

ADVERTS fragments as

⇒ AD1 = ADVERTS ∝ PROP1

⇒ AD2 = ADVERTS ∝ PROP2

⇒ AD3 = ADVERTS ∝ PROP3

Copyright © MSc Software Engineering 2003-2004

AD1				
AdNo	**PNo**	**NpNo**	**DateAdvert**	**Cost**
1	PG1	The Sun	01/06/2003	£100.00
2	PG5	Daily Telegraph	03/06/2003	£85.00
3	PG6	The Independent	08/06/2003	£70.00

AD2				
AdNo	**PNo**	**NpNo**	**DateAdvert**	**Cost**
4	PG8	Daily Mirror	15/06/2003	£80.00

AD3				
AdNo	**PNo**	**NpNo**	**DateAdvert**	**Cost**
5	PG9	Daily Telegraph	01/07/2003	£150.00

4.2.3 Checking for Correctness

1. Completeness

For a relation R who has a link to some owner relation S having fragments F_s = $\{S_1, S_2, S_3, ..., S_n\}$, and A be the attribute join both relations, then for each tuple t in R, there should be a tuple t' in S **[OV99]**.

$t[a] = t'[A]$

In our case, for our PROPERTIES table which has an owner link to BRANCHES, there is no tuple in PROPERTIES that relates to non existent tuple of BRANCHES. For example, no tuple exists for Brno= "B007" in PROPERTIES as we have no Brno = "B007" in BRANCHES relation.

Hence we can say that our fragments are complete as we have

$t[brno] = t'[brno]$

2. Reconstruction

Reconstruction of the global relation from its fragments is performed by the union operator in horizontal fragmentation. For example for relation R with fragments F_R = $\{R_1, R_2, R_3, ..., R_n\}$, we have **[OV99],**

$R = U\ R_i$

As in our case, we have 3 fragments defined on *BRANCHES*, as $F_{BRANCHES}$ = {BRAN1, BRAN2, BRAN3}, so we have

BRANCHES = BRAN1 U BRAN2 U BRAN3

Thus, if we unite our 3 fragments, we get the actual relation, so our fragments are correct.

Copyright © MSc Software Engineering 2003-2004

3. **Disjointness**

   It guarantees that as minterm predicates are determining the fragmentation, so our fragments should be mutually exclusive **[OV99]**.

   As we have,

   m1: City = "London"

   m2: City = "Glasgow"

   m3: City = "Aberdeen"

   It is not possible to have all three values in the same attribute CITY in a single row, so our fragments are mutually exclusive. Hence fulfils the concept of disjointness.

4.2.4 Summary

Thus after the result of horizontal fragmentation, we can successfully fragment our tables into number of fragments. Both Primary and Derived horizontal fragmentation were used to achieve the desired results.

Copyright © MSc Software Engineering 2003-2004

4.3 Vertical Fragmentation (VF)

Another approach for fragmenting tables is Vertical fragmentation **[OV99]**. In vertical fragmentation we distribute table by dividing columns across two or more sites. It is more difficult than horizontal fragmentation because more alternatives exist in this case **[OV99]**.

Two approaches are used to achieve vertical fragmentation;

1. Grouping, in which attributes are divided into fragments
2. Splitting, in which relations are distributed in fragments

Further we require different measures to calculate the fragmentation criteria. These measures are **[OV99]**:

- Affinity Measures
- Attribute Usage Values and
- Clustering and Partitioning

Therefore for our case study we will calculate these measures before fragmenting our tables vertically.

4.3.1 Application Requirements

Following applications are required to be accessed by the organization:

1. List the name, position and salary of staff at a given branch ordered by staff name
2. Identify the total number of staff and sum of their salaries
3. identify the total number of staff in each position at branches in Glasgow
4. List the name of each Manager at each branch, ordered by branch address
5. list the name of staff supervised by a named Supervisor

Following SQL commands can be written accordingly;

1. `SELECT fname, lname,  position, salary FROM staff WHERE brno='br0001' ORDER BY name;`
2. `SELECT COUNT(*), SUM(salary) from staff;`

Copyright © MSc Software Engineering 2003-2004

3. SELECT COUNT(*), staff.position FROM staff WHERE staff.brno = branches.bno AND branches.city = 'Glasgow' GROUP BY staff.position;

4. SELECT staff.fname, staff.lname, FROM staff, branches WHERE staff.brno = branches.bno AND staff.position='manager' ORDER BY branches.brno;

5. SELECT s1.fname, s1.lname FROM staff s1, staff s2 WHERE s1.stid = s2.sup_id AND s2.position = 'supervisor';

Based on these requirements, following fragmentation scheme is suggested:

4.3.2 Usage of Attributes, use (q_i, A_j)

For calculating the affinity measures, we first need attribute usage values according to the requirements.

Using above SQL queries, following usage matrix can be formed

A1 = StID, A2 = fname, A3 = lname, A4 = address, A5 = position, A6 = salary, A7 = sup_id, A8 = brno, A9 = date_hired, A10 = date_curr_loc, A11 = sex, A12 = dob, A13 = bonus

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13
Q1	0	1	1	0	1	1	0	1	0	0	0	0	0
Q2	0	0	0	0	0	1	0	0	0	0	0	0	0
Q3	0	0	0	0	1	0	0	1	0	0	0	0	0
Q4	0	1	1	0	1	0	0	1	0	0	0	0	0
Q5	1	1	1	0	1	0	1	0	0	0	0	0	0

Fig 16: Attribute Usage table

This matrix will help us in calculating affinity measures.

4.3.3 Affinity Measures aff (A_i, A_j)

In affinity measures, we calculate affinity of two given, A_i and A_j attributes. To calculate it we also require access frequencies for each query from different sites.

We assume following Access Frequencies for our requirements

Copyright © MSc Software Engineering 2003-2004

	S1	S2	S3
Q1	15	7	10
Q2	5	10	3
Q3	0	15	5
Q4	10	5	6
Q5	8	3	12

Fig 17: Site access frequencies table

The formula for calculating affinity measure is

$$aff(A_i, A_j) = \sum^{0}_{k} \quad \sum^{0}_{L} (S_L(qk) * acc(qk))$$
$$use(q_k, A_i)=1$$
$$use(q_k, A_j)=1$$

where k is the number of queries and L is the number of sites

In our case k = 5 and L = 3, thus using above formula following calculations are performed.

For A1 and A1

$$\Rightarrow aff(A1,A1) = \underset{use(q1,A1)=0}{\underset{use(q1,A1)=0}{0}} + \underset{use(q2,A1)=0}{\underset{use(q2,A1)=0}{0}} + \underset{use(q3,A1)=0}{\underset{use(q3,A1)=0}{0}} + \underset{use(q4,A1)=0}{\underset{use(q4,A1)=0}{0}} + \underset{use(q5,A1)=1}{\underset{use(q5,A1)=1}{(8*1 + 3*1 + 12*1)}}$$

$$\Rightarrow aff(A1,A1) = 0 + 0 + 0 + 0 + 23$$

$$\Rightarrow aff(A1,A1) = 23$$

For A1 and A2

$$\Rightarrow aff(A1,A2) = \underset{use(q1,A1)=0}{\underset{use(q1,A2)=1}{0}} + \underset{use(q2,A1)=0}{\underset{use(q2,A2)=0}{0}} + \underset{use(q3,A1)=0}{\underset{use(q3,A2)=0}{0}} + \underset{use(q4,A1)=0}{\underset{use(q4,A2)=1}{0}} + \underset{use(q5,A1)=1}{\underset{use(q5,A2)=1}{(8*1 + 3*1 + 12*1)}}$$

$$\Rightarrow aff(A1,A2) = 0 + 0 + 0 + 0 + 23$$
$$\Rightarrow aff(A1,A2) = 23$$

Copyright © MSc Software Engineering 2003-2004

For A1 and A3

```
⇒ aff(A1,A3) =          0     +          0     +          0     +
               use(q1,A1)=0      use(q2,A1)=0      use(q3,A1)=0
               use(q1,A3)=1      use(q2,A3)=0      use(q3,A3)=0

                     0     +          0
               use(q4,A1)=0      use(q5,A1)=1
               use(q4,A3)=1      use(q5,A3)=1

⇒ aff(A1,A3) = 0 + 0 + 0 + 0 + 23
⇒ aff(A1,A3) = 23
```

For A1 and A4

```
⇒ aff(A1,A4) =          0     +          0     +          0     +
               use(q1,A1)=0      use(q2,A1)=0      use(q3,A1)=0
               use(q1,A4)=0      use(q2,A4)=0      use(q3,A4)=0

                     0     +          0
               use(q4,A1)=0      use(q5,A1)=1
               use(q4,A4)=0      use(q5,A4)=0


⇒ aff(A1,A4) = 0
```

For A1 and A5

```
⇒ aff(A1,A5) =          0     +          0     +          0     +
               use(q1,A1)=0      use(q2,A1)=0      use(q3,A1)=0
               use(q1,A5)=1      use(q2,A5)=0      use(q3,A5)=1

                     0     +    (8*1 + 3*1 + 12*1)
               use(q4,A1)=0      use(q5,A1)=1
               use(q4,A5)=1      use(q5,A5)=1

⇒ aff(A1,A5) = 0 + 0 + 0 + 0 + 23
⇒ aff(A1,A5) = 23
```

For A1 and A6

```
⇒ aff(A1,A6) =          0     +          0     +          0     +
               use(q1,A1)=0      use(q2,A1)=0      use(q3,A1)=0
               use(q1,A6)=1      use(q2,A6)=1      use(q3,A6)=0

                     0     +          0
               use(q4,A1)=0      use(q5,A1)=0
               use(q4,A6)=0      use(q5,A6)=0

⇒ aff(A1,A6) = 0
```

For A1 and A7

```
⇒ aff(A1,A7) =          0     +          0     +          0     +
               use(q1,A1)=0      use(q2,A1)=0      use(q3,A1)=0
               use(q1,A7)=0      use(q2,A7)=0      use(q3,A7)=0

                     0     +    (8*1 + 3*1 + 12*1)
```


Copyright © MSc Software Engineering 2003-2004

```
                  use(q4,A1)=0      use(q5,A1)=1
                  use(q4,A7)=0      use(q5,A7)=1

⇒ aff(A1,A7) = 0 + 0 + 0 + 0 + 23
⇒ aff(A1,A7) = 23
```

For A1 and A8

```
⇒ aff(A1,A8) =         0     +           0     +           0     +
                  use(q1,A1)=0      use(q2,A1)=0      use(q3,A1)=0
                  use(q1,A8)=1      use(q2,A8)=0      use(q3,A8)=1

                        0     +           0
                  use(q4,A1)=0      use(q5,A1)=1
                  use(q4,A8)=1      use(q5,A8)=0

⇒ aff(A1,A8) = 0
```

For A1 and A9, A1 and A10, A1 and A11, A1 and A12, A1 and A13 affinity measure will be zero because usage of A9, A10, A11, A12 and A13 are zero.

```
⇒ aff(A1,A9) = 0
⇒ aff(A1,A10) = 0
⇒ aff(A1,A11) = 0
⇒ aff(A1,A12) = 0
⇒ aff(A1,A13) = 0
```

For A2 and A2

```
⇒ aff(A2,A2) = (15*1+7*1 +10*1) +         0     +           0     +
                   use(q1,A2)=1      use(q2,A2)=0      use(q3,A2)=0
                   use(q1,A2)=1      use(q2,A2)=0      use(q3,A2)=0

                   (10*1+5*1+6*1) +  (8*1 + 3*1 + 12*1)
                   use(q4,A2)=1      use(q5,A2)=1
                   use(q4,A2)=1      use(q5,A2)=1

⇒ aff(A2,A2) = 32 + 0 + 0 + 21 + 23
⇒ aff(A2,A2) = 76
```

For A2 and A3

```
⇒ aff(A2,A3) = (15*1+7*1 +10*1) +         0     +           0     +
                   use(q1,A2)=1      use(q2,A2)=0      use(q3,A2)=0
                   use(q1,A3)=1      use(q2,A3)=0      use(q3,A3)=0

                   (10*1+5*1+6*1) +  (8*1 + 3*1 + 12*1)
                   use(q4,A2)=1      use(q5,A2)=1
                   use(q4,A3)=1      use(q5,A3)=1

⇒ aff(A2,A3) = 32 + 0 + 0 + 21 + 23
⇒ aff(A2,A3) = 76
```


Copyright © MSc Software Engineering 2003-2004

For A2 and A4

As all A4 usage is zero so affinity measure will be zero. Hence

⇒ aff(A2,A4) = 0

For A2 and A5

```
⇒ aff(A2,A5) = (15*1+7*1 +10*1) +        0     +           0     +
                 use(q1,A2)=1      use(q2,A2)=0      use(q3,A2)=0
                 use(q1,A5)=1      use(q2,A5)=0      use(q3,A5)=1

                 (10*1+5*1+6*1) +  (8*1 + 3*1 + 12*1)
                 use(q4,A2)=1      use(q5,A2)=1
                 use(q4,A5)=1      use(q5,A5)=1

⇒ aff(A2,A5) = 32 + 0 + 0 + 21 + 23
⇒ aff(A2,A5) = 76
```

For A2 and A6

```
⇒ aff(A2,A6) = (15*1+7*1 +10*1) +        0     +           0     +
                 use(q1,A2)=1      use(q2,A2)=0      use(q3,A2)=0
                 use(q1,A6)=1      use(q2,A6)=1      use(q3,A6)=0

                       0     +           0
                 use(q4,A2)=1      use(q5,A2)=1
                 use(q4,A6)=0      use(q5,A6)=0

⇒ aff(A2,A6) = 32 + 0 + 0 + 0 + 0
⇒ aff(A2,A6) = 32
```

For A2 and A7

```
⇒ aff(A2,A7) =         0     +           0     +           0     +
                 use(q1,A2)=1      use(q2,A2)=0      use(q3,A2)=0
                 use(q1,A7)=0      use(q2,A7)=0      use(q3,A7)=0

                       0     +     (8*1 + 3*1 + 12*1)
                 use(q4,A2)=1      use(q5,A2)=1
                 use(q4,A7)=0      use(q5,A7)=1

⇒ aff(A2,A7) = 0 + 0 + 0 + 0 + 23
⇒ aff(A2,A7) = 23
```

For A2 and A8

```
⇒ aff(A2,A8) = (15*1+7*1 +10*1) +        0     +           0     +
                 use(q1,A2)=1      use(q2,A2)=0      use(q3,A2)=0
                 use(q1,A8)=1      use(q2,A8)=0      use(q3,A8)=1

                 (10*1+5*1+6*1) +        0
                 use(q4,A2)=1      use(q5,A2)=1
                 use(q4,A8)=1      use(q5,A8)=0

⇒ aff(A2,A8) = 32 + 0 + 0 + 21 + 0
```


Copyright © MSc Software Engineering 2003-2004

```
⇒ aff(A2,A8) = 53
```

Similarly, for A2 and A9, A2 and A10, A2 and A11, A2 and A12, A2 and A13 affinity measure will be zero because usage of A9, A10, A11, A12 and A13 are zero.

```
⇒ aff(A2,A9) = 0
⇒ aff(A2,A10) = 0
⇒ aff(A2,A11) = 0
⇒ aff(A2,A12) = 0
⇒ aff(A2,A13) = 0
```

For A3 and A3

```
⇒ aff(A3,A3) = (15*1+7*1 +10*1) +        0     +           0     +
                 use(q1,A3)=1      use(q2,A3)=0      use(q3,A3)=0
                 use(q1,A3)=1      use(q2,A3)=0      use(q3,A3)=0

                 (10*1+5*1+6*1) +  (8*1 + 3*1 + 12*1)
                 use(q4,A3)=1      use(q5,A3)=1
                 use(q4,A3)=1      use(q5,A3)=1

⇒ aff(A3,A3) = 32 + 0 + 0 + 21 + 23
⇒ aff(A3,A3) = 76
```

For A3 and A4

As all A4 usage is zero so affinity measure will be zero. Hence

```
⇒ aff(A3,A4) = 0
```

For A3 and A5

```
⇒ aff(A3,A5) = (15*1+7*1 +10*1) +        0     +           0     +
                 use(q1,A3)=1      use(q2,A3)=0      use(q3,A3)=0
                 use(q1,A5)=1      use(q2,A5)=0      use(q3,A5)=1

                 (10*1+5*1+6*1) +  (8*1 + 3*1 + 12*1)
                 use(q4,A3)=1      use(q5,A3)=1
                 use(q4,A5)=1      use(q5,A5)=1

⇒ aff(A3,A5) = 32 + 0 + 0 + 21 + 23
⇒ aff(A3,A5) = 76
```

For A3 and A6

```
⇒ aff(A3,A6) = (15*1+7*1 +10*1) +        0     +           0     +
                 use(q1,A3)=1      use(q2,A3)=0      use(q3,A3)=0
                 use(q1,A6)=1      use(q2,A6)=1      use(q3,A6)=0

                       0     +           0
                 use(q4,A3)=1      use(q5,A3)=1
                 use(q4,A6)=0      use(q5,A6)=0

⇒ aff(A3,A6) = 32 + 0 + 0 + 0 + 0
⇒ aff(A3,A6) = 32
```


Copyright © MSc Software Engineering 2003-2004

For A3 and A7

```
⇒ aff(A3,A7) =          0     +           0     +           0     +
               use(q1,A3)=1      use(q2,A3)=0      use(q3,A3)=0
               use(q1,A7)=0      use(q2,A7)=0      use(q3,A7)=0

                     0     +     (8*1 + 3*1 + 12*1)
               use(q4,A3)=1      use(q5,A3)=1
               use(q4,A7)=0      use(q5,A7)=1

⇒ aff(A3,A7) = 0 + 0 + 0 + 0 + 23
⇒ aff(A3,A7) = 23
```

For A3 and A8

```
⇒ aff(A3,A8) = (15*1+7*1 +10*1) +        0     +           0     +
                 use(q1,A3)=1      use(q2,A3)=0      use(q3,A3)=0
                 use(q1,A8)=1      use(q2,A8)=0      use(q3,A8)=1

                 (10*1+5*1+6*1) +        0
                 use(q4,A3)=1      use(q5,A3)=1
                 use(q4,A8)=1      use(q5,A8)=0

⇒ aff(A3,A8) = 32 + 0 + 0 + 21 + 0
⇒ aff(A3,A8) = 53
```

Similarly, for A3 and A9, A3 and A10, A3 and A11, A3 and A12, A3 and A13 affinity measure will be zero because usage of A9, A10, A11, A12 and A13 are zero.

```
⇒ aff(A3,A9) = 0
⇒ aff(A3,A10) = 0
⇒ aff(A3,A11) = 0
⇒ aff(A3,A12) = 0
⇒ aff(A3,A13) = 0
```

For attribute A4, all the affinity measures with other attributes will be zero, as A4 usage is zero. Thus,

```
⇒ aff(A4,A4) = 0
⇒ aff(A4,A5) = 0
⇒ aff(A4,A6) = 0
⇒ aff(A4,A7) = 0
⇒ aff(A4,A8) = 0
⇒ aff(A4,A9) = 0
⇒ aff(A4,A10) = 0
⇒ aff(A4,A11) = 0
⇒ aff(A4,A12) = 0
⇒ aff(A4,A13) = 0
```


Copyright © MSc Software Engineering 2003-2004

For A5 and A5

```
⇒ aff(A5,A5) = (15*1+7*1+10*1) +         0     +     (0*1+15*1+5*1)    +
                 use(q1,A5)=1      use(q2,A5)=0      use(q3,A5)=1
                 use(q1,A5)=1      use(q2,A5)=0      use(q3,A5)=1

                 (10*1+5*1+6*1) +  (8*1+3*1+12*1)
                 use(q4,A5)=1      use(q5,A5)=1
                 use(q4,A5)=1      use(q5,A5)=1

⇒ aff(A5,A5) = 32 + 0 + 20 + 21 + 23
⇒ aff(A5,A5) = 96
```

For A5 and A6

```
⇒ aff(A5,A6) = (15*1+7*1+10*1) +         0     +           0     +
                 use(q1,A5)=1      use(q2,A5)=0      use(q3,A5)=1
                 use(q1,A6)=1      use(q2,A6)=1      use(q3,A6)=0

                       0     +           0
                 use(q4,A5)=1      use(q5,A5)=1
                 use(q4,A6)=0      use(q5,A6)=0

⇒ aff(A5,A6) = 32 + 0 + 0 + 0 + 0
⇒ aff(A5,A6) = 32
```

For A5 and A7

```
⇒ aff(A5,A7) =         0     +           0     +           0     +
                 use(q1,A5)=1      use(q2,A5)=0      use(q3,A5)=1
                 use(q1,A7)=0      use(q2,A7)=0      use(q3,A7)=0

                       0     +     (8*1+3*1+12*1)
                 use(q4,A5)=1      use(q5,A5)=1
                 use(q4,A7)=0      use(q5,A7)=1

⇒ aff(A5,A7) = 0 + 0 + 0 + 0 + 23
⇒ aff(A5,A7) = 23
```

For A5 and A8

```
⇒ aff(A5,A8) =   (15*1+7*1+10*1) +       0     +     (0*1+15*1+5*1) +
                 use(q1,A5)=1      use(q2,A5)=0      use(q3,A5)=1
                 use(q1,A8)=1      use(q2,A8)=0      use(q3,A8)=1

                 (10*1+5*1+6*1)  +       0
                 use(q4,A5)=1      use(q5,A5)=1
                 use(q4,A8)=1      use(q5,A8)=0

⇒ aff(A5,A8) = 32 + 0 + 20 + 21 + 0
⇒ aff(A5,A8) = 73
```

Similarly, for A5 and A9, A5 and A10, A5 and A11, A5 and A12, A5 and A13 affinity measure will be zero because usage of A9, A10, A11, A12 and A13 are zero.

```
⇒ aff(A5,A9) = 0
⇒ aff(A5,A10) = 0
```


Copyright © MSc Software Engineering 2003-2004

```
⇒ aff(A5,A11) = 0
⇒ aff(A5,A12) = 0
⇒ aff(A5,A13) = 0
```

For A6 and A6

```
⇒ aff(A6,A6) =   (15*1+7*1+10*1) + (5*1+10*1+3*1) +        0     +
                 use(q1,A6)=1      use(q2,A6)=1      use(q3,A6)=0
                 use(q1,A6)=1      use(q2,A6)=1      use(q3,A6)=0

                       0     +           0
                 use(q4,A6)=0      use(q5,A6)=0
                 use(q4,A6)=0      use(q5,A6)=0

⇒ aff(A6,A6) = 32 + 10 + 0 + 0 + 0
⇒ aff(A6,A6) = 50
```

For A6 and A7

```
⇒ aff(A6,A7) =         0     +           0     +           0     +
                 use(q1,A6)=1      use(q2,A6)=1      use(q3,A6)=0
                 use(q1,A7)=0      use(q2,A7)=0      use(q3,A7)=0

                       0     +           0
                 use(q4,A6)=0      use(q5,A6)=0
                 use(q4,A7)=0      use(q5,A7)=1

⇒ aff(A6,A7) = 0
```

For A6 and A8

```
⇒ aff(A6,A8) =   (15*1+7*1+10*1) +       0     +           0     +
                 use(q1,A6)=1      use(q2,A6)=1      use(q3,A6)=0
                 use(q1,A8)=1      use(q2,A8)=0      use(q3,A8)=1

                       0     +           0
                 use(q4,A6)=0      use(q5,A6)=0
                 use(q4,A8)=1      use(q5,A8)=1

⇒ aff(A6,A8) = 32 + 0 + 0 + 0 + 0
⇒ aff(A6,A9) = 32
```

For A6 and A9, A6 and A10, A6 and A11, A6 and A12, A6 and A13 affinity measure will be zero because usage of A9, A10, A11, A12 and A13 are zero.

```
⇒ aff(A6,A9) = 0
⇒ aff(A6,A10) = 0
⇒ aff(A6,A11) = 0
⇒ aff(A6,A12) = 0
⇒ aff(A6,A13) = 0
```


Copyright © MSc Software Engineering 2003-2004

For A7 and A7

```
⇒ aff(A7,A7) =          0     +           0     +           0     +
                  use(q1,A7)=0      use(q2,A7)=0      use(q3,A7)=0
                  use(q1,A7)=0      use(q2,A7)=0      use(q3,A7)=0

                        0     +     (8*1 + 3*1 + 12*1)
                  use(q4,A7)=0      use(q5,A7)=1
                  use(q4,A7)=0      use(q5,A7)=1

⇒ aff(A7,A7) = 0 + 0 + 0 + 0 + 23
⇒ aff(A7,A7) = 23
```

For A7 and A8

```
⇒ aff(A7,A8) =          0     +           0     +           0     +
                  use(q1,A7)=0      use(q2,A7)=0      use(q3,A7)=0
                  use(q1,A8)=1      use(q2,A8)=0      use(q3,A8)=1

                        0     +           0
                  use(q4,A7)=0      use(q5,A7)=1
                  use(q4,A8)=1      use(q5,A8)=0

⇒ aff(A7,A8) = 0
```

For A7 and A9, A7 and A10, A7 and A11, A7 and A12, A7 and A13 affinity measure will be zero because usage of A9, A10, A11, A12 and A13 are zero.

```
⇒ aff(A7,A9) = 0
⇒ aff(A7,A10) = 0
⇒ aff(A7,A11) = 0
⇒ aff(A7,A12) = 0
⇒ aff(A7,A13) = 0
```

For A8 and A8

```
⇒ aff(A8,A8) = (15*1+7*1+10*1) +          0     +     (0*1+15*1+5*1)    +
                   use(q1,A8)=1      use(q2,A8)=0      use(q3,A8)=1
                   use(q1,A8)=1      use(q2,A8)=0      use(q3,A8)=1

                   (10*1+5*1+6*1) +        0
                   use(q4,A8)=1      use(q5,A8)=0
                   use(q4,A8)=1      use(q5,A8)=0

⇒ aff(A8,A8) = 32 + 0 + 20 + 21 + 0
⇒ aff(A8,A8) = 73
```

For A8 and A9, A8 and A10, A8 and A11, A8 and A12, A8 and A13 affinity measure will be zero because usage of A9, A10, A11, A12 and A13 are zero.

```
⇒ aff(A8,A9) = 0
⇒ aff(A8,A10) = 0
⇒ aff(A8,A11) = 0
⇒ aff(A8,A12) = 0
⇒ aff(A8,A13) = 0
```


Copyright © MSc Software Engineering 2003-2004

For A9, A10, A11, A12, and A13 all affinity measures will be zero

⇒ aff(A9,A9) = 0
⇒ aff(A9,A10) = 0
⇒ aff(A9,A11) = 0
⇒ aff(A9,A12) = 0
⇒ aff(A9,A13) = 0

⇒ aff(A10,A10) = 0
⇒ aff(A10,A11) = 0
⇒ aff(A10,A12) = 0
⇒ aff(A10,A13) = 0

⇒ aff(A11,A11) = 0
⇒ aff(A11,A12) = 0
⇒ aff(A11,A13) = 0

⇒ aff(A12,A12) = 0
⇒ aff(A12,A13) = 0

⇒ aff(A13,A13) = 0

4.3.4 Affinity Matrix

Thus after calculating all attributes affinity measure values, we can now formulate an affinity matrix.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13
A1	23	23	23	0	23	0	23	0	0	0	0	0	0
A2	23	76	76	0	76	32	23	53	0	0	0	0	0
A3	23	76	76	0	76	32	23	53	0	0	0	0	0
A4	0	0	0	0	0	0	0	0	0	0	0	0	0
A5	23	76	76	0	96	32	23	73	0	0	0	0	0
A6	0	32	32	0	32	50	0	32	0	0	0	0	0
A7	23	23	23	0	23	0	23	0	0	0	0	0	0
A8	0	53	53	0	73	32	0	73	0	0	0	0	0
A9	0	0	0	0	0	0	0	0	0	0	0	0	0
A10	0	0	0	0	0	0	0	0	0	0	0	0	0
A11	0	0	0	0	0	0	0	0	0	0	0	0	0
A12	0	0	0	0	0	0	0	0	0	0	0	0	0
A13	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig 18: Affinity Matrix

Copyright © MSc Software Engineering 2003-2004

4.3.5 Bond Energy Algorithm (BEA)

After calculating affinity matrix, the next step is to calculate bond energy between the attribute and reordering of the attributes with higher values. The result will be the clustered affinity matrix.

The Bond Energy Algorithm takes as input the attribute affinity matrix, permutes its rows and columns and generates a clustered affinity matrix CA **[OV99]**. General formula for BEA is:

$$cont(A_i, A_k, A_j) = 2 * bond(A_i, A_k) + 2 * bond(A_k, A_j) - 2 * bond(A_i, A_j)$$

Where

$$bond(A_x, A_y) = \sum^{n}_{z=1} aff(A_z, A_x) * aff(A_z, A_y)$$

Thus we have to follow this formula to get the desired CA matrix.

Generation of clustered affinity matrix will be done in 3 steps **[OV99],**

1. Initialization: placement of any column arbitrarily between any 2 columns
2. Iteration: placing remaining columns in the other remaining places
3. Row ordering: ordering the rows according to the placement of columns

4.3.5.1 Ordering 3-5-4

So to calculate our CA matrix, we will consider order of 3-5-4, i.e. we are putting A5 between A3 and A4 and will check the bond energy between A3 and A5 to determine whether it's suitable to put these two together.

	A1	A2	**A3**	**A5**	**A4**
A1	23	23	23	23	0
A2	23	76	76	76	0
A3	23	76	76	76	0
A4	0	0	0	0	0
A5	23	76	76	96	0
A6	0	32	32	32	0
A7	23	23	23	23	0
A8	0	53	53	73	0
A9	0	0	0	0	0
A10	0	0	0	0	0
A11	0	0	0	0	0
A12	0	0	0	0	0

Fig 19: Ordering 3-5-4

Copyright © MSc Software Engineering 2003-2004

$cont(A_3, A_5, A_4) = 2 * bond(A_3, A_5) + 2 * bond(A_5, A_4) - 2 * bond(A_3, A_4)$

Then

$\Rightarrow bond(A_3, A_5) = \sum^{12}_{z=1} aff(A_z, A_x) * aff(A_z, A_y)$

$\Rightarrow bond(A_3, A_5) = aff(A_1, A_3) * aff(A_1, A_5) + aff(A_2, A_3) * aff(A_2, A_5) +$
$aff(A_3, A_3) * aff(A_3, A_5) + aff(A_4, A_3) * aff(A_4, A_5) +$
$aff(A_5, A_3) * aff(A_5, A_5) + aff(A_6, A_3) * aff(A_6, A_5) +$
$aff(A_7, A_3) * aff(A_7, A_5) + aff(A_8, A_3) * aff(A_8, A_5) +$
$aff(A_9, A_3) * aff(A_9, A_5) + aff(A_{10}, A_3) * aff(A_{10}, A_5) +$
$aff(A_{11}, A_3) * aff(A_{11}, A_5) + aff(A_{12}, A_3) * aff(A_{12}, A_5) +$

$\Rightarrow bond(A_3, A_5) =$ 23*23 + 76*76 + 76*76 + 0 + 76*96 + 32*32 + 23*23 + 53*73 + 0 + 0 + 0 + 0

$\Rightarrow bond(A_3, A_5) = 529 + 5776 + 5776 + 0 + 7296 + 1024 + 529 + 3869 + 0$

$\Rightarrow bond(A_3, A_5) = 24799$

Similarly,

$\Rightarrow bond(A_5, A_4) =$ 23*0 + 76*0 + 76*0 + 0*0 + 96*0 + 32*0 + 23*0 + 73*0 + 0 + 0 + 0 + 0

$\Rightarrow bond(A_5, A_4) = 0$

And,

$\Rightarrow bond(A_3, A_4) = 0$

Hence we have,

$\Rightarrow cont(A_3, A_5, A_4) = 2 * 24799 + 2*0 + 2*0$

$\Rightarrow$ **$cont(A_3, A_5, A_4) = 49598$**

4.3.5.2 Ordering 1-5-2

	A1	**A5**	**A2**	A3	A4	A6	A7
A1	23	23	23	23	0	0	23
A2	23	76	76	76	0	32	23
A3	23	76	76	76	0	32	23
A4	0	0	0	0	0	0	0
A5	23	96	76	76	0	32	23
A6	0	32	32	32	0	50	0
A7	23	23	23	23	0	0	23
A8	0	73	53	53	0	32	0
A9	0	0	0	0	0	0	0
A10	0	0	0	0	0	0	0
A11	0	0	0	0	0	0	0
A12	0	0	0	0	0	0	0

Copyright © MSc Software Engineering 2003-2004

Fig 20: Ordering 1-5-2

$cont(A_1, A_5, A_2) = 2 * bond(A_1, A_5) + 2 * bond(A_5, A_2) - 2 * bond(A_1, A_2)$

Then,

$\Rightarrow bond(A_1, A_5) = \sum^{12}_{z=1} aff(A_z, A_x) * aff(A_z, A_y)$

$\Rightarrow bond(A_1, A_5) = 23*23+23*76+23*76+0*0+23*96+0*32+23*23+0*73+0+0$

$\Rightarrow bond(A_1, A_5) = 529 + 1748 + 1748 + 0 + 2208 + 0 + 529 +0+0+0$

$\Rightarrow bond(A_1, A_5) = 6762$

Similarly,

$\Rightarrow bond(A_5, A_2) = 23*23+76*76+76*76+0+96*76+32*32+23*23+73*53+0+0$

$\Rightarrow bond(A_5, A_2) = 529 + 5776 +5776 + 0 + 7296 + 1024 + 529 + 3869$

$\Rightarrow bond(A_5, A_2) = 24799$

And,

$\Rightarrow bond(A_1, A_2) = 23*23+23*76+23*76+0+23*76+0+23*23+0+0+0+0$

$\Rightarrow bond(A_1, A_2) = 529 + 1748 + 1748 + 0\ 1748 + 0 + 529 +0$

$\Rightarrow bond(A_1, A_2) = 6302$

Hence we get,

$\Rightarrow cont(A_1, A_5, A_2) = 2 * 6762 + 2 * 24799 - 2 * 6302$

$\Rightarrow cont(A_1, A_5, A_2) = 13524 + 49598 - 12604$

$\mathbf{\Rightarrow cont(A_1, A_5, A_2) = 50518}$

4.3.5.3 Ordering 1-2-3

	A1	A2	A3
A1	23	23	23
A2	23	76	76
A3	23	76	76
A4	0	0	0
A5	23	76	76
A6	0	32	32
A7	23	23	23
A8	0	53	53
A9	0	0	0
A10	0	0	0
A11	0	0	0
A12	0	0	0

Fig 21: Ordering 1-2-3

Copyright © MSc Software Engineering 2003-2004

$cont(A_1, A_2, A_3) = 2 * bond(A_1, A_2) + 2 * bond(A_2, A_3) - 2 * bond(A_1, A_3)$

We have

$\Rightarrow bond(A_1, A_2) = 6302$

As affinity measures are same for A2 and A3, so we have

$\Rightarrow bond(A_1, A_3) = bond(A_1, A_2) = 6302$

And,

$\Rightarrow bond(A_2, A_3) = 23*23+76*76+76*76+0+76*76+32*32+23*23+53*53+0+0$

$\Rightarrow bond(A_2, A_3) = 529 + 5776 + 5776 + 0 + 5776 +1024 + 529 + 2809$

$\Rightarrow bond(A_2, A_3) = 22219$

Thus we get,

$\Rightarrow cont(A_1, A_2, A_3) = 2 *6302 + 2 * 22219 - 2 * 6302$

$\Rightarrow cont(A_1, A_2, A_3) = 12604 + 44438 - 12604$

$\Rightarrow \mathbf{cont(A_1, A_2, A_3) = 44438}$

4.3.5.4 Ordering 3-7-4

	A1	A2	**A3**	**A7**	**A4**
A1	23	23	23	23	0
A2	23	76	76	23	0
A3	23	76	76	23	0
A4	0	0	0	0	0
A5	23	76	76	23	0
A6	0	32	32	0	0
A7	23	23	23	23	0
A8	0	53	53	0	0
A9	0	0	0	0	0
A10	0	0	0	0	0
A11	0	0	0	0	0
A12	0	0	0	0	0
A13	0	0	0	0	0

Fig 22: Ordering 3-7-4

$cont(A_3, A_7, A_4) = 2 * bond(A_3, A_7) + 2 * bond(A_7, A_4) - 2 * bond(A_3, A_4)$

We have

$\Rightarrow bond(A_3, A_4) = 0$ as calculated before

For

$\Rightarrow bond(A_3, A_7) = 23*23+76*23+76*23+0+76*23+32*0+23*23+53*0+0+0+0$

Copyright © MSc Software Engineering 2003-2004

$\Rightarrow$ bond(A_3, A_7) = 529 + 1748 + 1748 + 1748 + 0 + 529 + 0

$\Rightarrow$ bond(A_3, A_7) = 6302

And

$\Rightarrow$ bond(A_7, A_4) = 23*0+23*0+23*0+0+0+0+23*0

$\Rightarrow$ bond(A_7, A_4) = 0

Hence,

$\Rightarrow$ cont(A_3, A_7, A_4) = 2 * 6302 + 2 * 0 – 2 * 0

$\Rightarrow$ **cont(A_3, A_7, A_4) = 12604**

4.3.5.5 Ordering 5-2-3

	A1	**A5**	**A2**	**A3**	A4
A1	23	23	23	23	0
A2	23	76	76	76	0
A3	23	76	76	76	0
A4	0	0	0	0	0
A5	23	96	76	76	0
A6	0	32	32	32	0
A7	23	23	23	23	0
A8	0	73	53	53	0
A9	0	0	0	0	0
A10	0	0	0	0	0
A11	0	0	0	0	0
A12	0	0	0	0	0
A13	0	0	0	0	0

Fig 23: Ordering 5-2-3

cont(A_5, A_2, A_3) = 2 * bond(A_5, A_2) + 2 * bond(A_2, A_3) – 2 * bond(A_5, A_3)

We have

$\Rightarrow$ bond(A_2, A_3) = 22219 (as calculated before)

and

$\Rightarrow$ bond(A_5, A_2) = 24799

For bond(A_5, A_3), as we have similar affinity measure values for A3 and A2, so we have

$\Rightarrow$ bond(A_5, A_3) = bond(A_5, A_2) = 24799

$\Rightarrow$ bond(A_5, A_3) = 24799

Thus we get,

$\Rightarrow$ cont(A_5, A_2, A_3) = 2 * 24799 + 2 * 22219 – 2 * 24799

$\Rightarrow$ cont(A_5, A_2, A_3) = 49598 + 44438 – 49598

Copyright © MSc Software Engineering 2003-2004

$\Rightarrow$ **cont(A_5, A_2, A_3) = 44438**

4.3.5.6 Ordering 7-8-6

	A1	A2	A3	A4	A5	A7	A8	A6
A1	23	23	23	0	23	23	0	0
A2	23	76	76	0	76	23	53	32
A3	23	76	76	0	76	23	53	32
A4	0	0	0	0	0	0	0	0
A5	23	76	76	0	96	23	73	32
A6	0	32	32	0	32	0	32	50
A7	23	23	23	0	23	23	0	0
A8	0	53	53	0	73	0	73	32
A9	0	0	0	0	0	0	0	0
A10	0	0	0	0	0	0	0	0
A11	0	0	0	0	0	0	0	0
A12	0	0	0	0	0	0	0	0
A13	0	0	0	0	0	0	0	0

Fig 24: Ordering 7-8-6

cont(A_7, A_8, A_6) = 2 * bond(A_7, A_8) + 2 * bond(A_8, A_6) – 2 * bond(A_7, A_6)

We have

$\Rightarrow$ bond(A_7, A_8) = 23*0+53*0+23*53+0+23*73+0+0+0

$\Rightarrow$ bond(A_7, A_8) = 0+0+ 1219 +0+ 1679 +0

$\Rightarrow$ bond(A_7, A_8) = 2898

and

$\Rightarrow$ bond(A_8, A_6) = 0+32*53+32*53+32*73+50*32+0+32*73

$\Rightarrow$ bond(A_8, A_6) = 1696 + 1696 + 2336 + 1600 + 2336

$\Rightarrow$ bond(A_8, A_6) = 9664

and for

$\Rightarrow$ bond(A_7, A_6) = 0+23*32+23*32+0+23*32+0+0+0

$\Rightarrow$ bond(A_7, A_6) = 736 + 736 + 736

$\Rightarrow$ bond(A_7, A_6) = 2208

So we get,

$\Rightarrow$ cont(A_7, A_8, A_6) = 2 * 2898 + 2 * 9664 – 2 * 2208

$\Rightarrow$ cont(A_7, A_8, A_6) = 5796 + 19328 – 4416

$\Rightarrow$ cont(A_7, A_8, A_6) = 20708

Copyright © MSc Software Engineering 2003-2004

4.3.5.7 Conclusion

Thus after calculating different orderings, we finally arrange our columns in the order of 1-5-2-3-7-8-6-4-9-10-11-12-13.

Therefore we will get the clustered affinity matrix as:

	A1	A5	A2	A3	A7	A8	A6	A4	A9	A10	A11	A12	A13
A1	23	23	23	23	23	0	0	0	0	0	0	0	0
A2	23	76	76	76	23	53	32	0	0	0	0	0	0
A3	23	76	76	76	23	53	32	0	0	0	0	0	0
A4	0	0	0	0	0	0	0	0	0	0	0	0	0
A5	23	96	76	76	23	73	32	0	0	0	0	0	0
A6	0	32	32	32	0	32	50	0	0	0	0	0	0
A7	23	23	23	23	23	0	0	0	0	0	0	0	0
A8	0	73	53	53	0	73	32	0	0	0	0	0	0
A9	0	0	0	0	0	0	0	0	0	0	0	0	0
A10	0	0	0	0	0	0	0	0	0	0	0	0	0
A11	0	0	0	0	0	0	0	0	0	0	0	0	0
A12	0	0	0	0	0	0	0	0	0	0	0	0	0
A13	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig 25: Clustered Affinity (CA) matrix

After performing Row Ordering we will get

	A1	A5	A2	A3	A7	A8	A6	A4	A9	A10	A11	A12	A13
A1	23	23	23	23	23	0	0	0	0	0	0	0	0
A5	23	96	76	76	23	73	32	0	0	0	0	0	0
A2	23	76	76	76	23	53	32	0	0	0	0	0	0
A3	23	76	76	76	23	53	32	0	0	0	0	0	0
A7	23	23	23	23	23	0	0	0	0	0	0	0	0
A8	0	73	53	53	0	73	32	0	0	0	0	0	0
A6	0	32	32	32	0	32	50	0	0	0	0	0	0
A4	0	0	0	0	0	0	0	0	0	0	0	0	0
A9	0	0	0	0	0	0	0	0	0	0	0	0	0
A10	0	0	0	0	0	0	0	0	0	0	0	0	0
A11	0	0	0	0	0	0	0	0	0	0	0	0	0
A12	0	0	0	0	0	0	0	0	0	0	0	0	0
A13	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig 26: Clustered Affinity (CA) matrix after Row ordering

Copyright © MSc Software Engineering 2003-2004

4.3.6 Partitioning Algorithm

The objective of the splitting activity is to find sets of attributes that are accessed by distinct set of applications **[OV99]**. Based on our clustered affinity matrix, we can divide our matrix as follows:

4.3.6.1 Splitting point n

	A1	A5	A2	A3	A7	A8	A6	A4	A9	A10	A11	A12	A13
A1	23	23	23	23	23	0	0	0	0	0	0	0	0
A5	23	96	76	76	23	73	32	0	0	0	0	0	0
A2	23	76	76	76	23	53	32	0	0	0	0	0	0
A3	23	76	76	76	23	53	32	0	0	0	0	0	0
A7	23	23	23	23	23	0	0	0	0	0	0	0	0
A8	0	73	53	53	0	73	32	0	0	0	0	0	0
A6	0	32	32	32	0	32	50	0	0	0	0	0	0
A4	0	0	0	0	0	0	0	0	0	0	0	0	0
A9	0	0	0	0	0	0	0	0	0	0	0	0	0
A10	0	0	0	0	0	0	0	0	0	0	0	0	0
A11	0	0	0	0	0	0	0	0	0	0	0	0	0
A12	0	0	0	0	0	0	0	0	0	0	0	0	0
A13	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig 27: Locating splitting point n

The upper left cluster has the attributes most closely related to each other and we denote this cluster as TA. The bottom right cluster is denoted by BA. As we have a set of applications defined Q = {q1, q2, q3, q4, q5}, so based on those applications we will now calculate our partitions and cost functions **[OV99]**.

Let AQ(qi) be the Attributes accessed by each application then we will have

⇒ AQ(q1) = {A2, A3, A5, A6, A8}

⇒ AQ(q2) = {A6}

⇒ AQ(q3) = {A5, A8}

⇒ AQ(q4) = {A2, A3, A5, A8}

⇒ AQ(q5) = {A1, A2, A3, A5, A7}

TQ be the applications accessed by TA and BQ set of applications accessed by BA, then we will have **[OV99]**

⇒ TQ = {q1, q2, q3, q4, q5}

⇒ BQ = {}

Copyright © MSc Software Engineering 2003-2004

And OQ be the set of applications accessed by both TA and BA then we have **[OV99]**

$\Rightarrow$ OQ = Q - {TQ U BQ}

$\Rightarrow$ OQ = Q - {q1, q2, q3, q4, q5}

$\Rightarrow$ OQ = {}

The cost functions are:

CQ is the total number of accesses to attributes by Q then we have **[OV99]**

$\Rightarrow$ CQ = (15+7+10) + (5+10+3) + (0+15+5) + (10+5+6) + (8+3+12)

$\Rightarrow$ CQ = 32 + 18 + 20 + 21 + 23

$\Rightarrow$ **CQ = 114**

CTQ be the total number of accesses to attributes by applications that access only TA **[OV99]** then we have,

$\Rightarrow$ CTQ = (15+7+10) + (5+10+3) + (0+15+5) + (10+5+6) + (8+3+12)

$\Rightarrow$ CTQ = 32 + 18 + 20 + 21 + 23

$\Rightarrow$ **CTQ = 114**

CBQ be the total number of accesses to attributes by applications that access only BA **[OV99]** then we have,

$\Rightarrow$ **CBQ = 0**

COQ be the total number of accesses to attributes by applications that access both TA **[OV99]** and BA then we have,

$\Rightarrow$ **COQ = 0**

Then the point along diagonal will be

$\Rightarrow$ $Z = CTQ * CBQ - COQ^2$

$\Rightarrow$ Z = 114 * 0 – 0

$\Rightarrow$ **Z = 0**

Copyright © MSc Software Engineering 2003-2004

4.3.6.2 Splitting point n-1

	A1	A5	A2	A3	A7	A8	A6	A4	A9	A10	A11	A12	A13
A1	23	23	23	23	23	0	0	0	0	0	0	0	0
A5	23	96	76	76	23	73	32	0	0	0	0	0	0
A2	23	76	76	76	23	53	32	0	0	0	0	0	0
A3	23	76	76	76	23	53	32	0	0	0	0	0	0
A7	23	23	23	23	23	0	0	0	0	0	0	0	0
A8	0	73	53	53	0	73	32	0	0	0	0	0	0
A6	0	32	32	32	0	32	50	0	0	0	0	0	0
A4	0	0	0	0	0	0	0	0	0	0	0	0	0
A9	0	0	0	0	0	0	0	0	0	0	0	0	0
A10	0	0	0	0	0	0	0	0	0	0	0	0	0
A11	0	0	0	0	0	0	0	0	0	0	0	0	0
A12	0	0	0	0	0	0	0	0	0	0	0	0	0
A13	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig 28: Locating splitting point n-1

TQ be the applications accessed by TA and BQ set of applications accessed by BA **[OV99]**, then we will have

⇒ TQ = {q3, q4, q5}

⇒ BQ = {q2}

And OQ be the set of applications accessed by both TA and BA **[OV99]** then we have

⇒ OQ = Q - {TQ U BQ}

⇒ OQ = Q - {q2, q3, q4, q5}

⇒ OQ = {q1}

The cost functions are:

CQ is the total number of accesses to attributes by Q **[OV99]** then we have

⇒ CQ = (15+7+10) + (5+10+3) + (0+15+5) + (10+5+6) + (8+3+12)

⇒ CQ = 32 + 18 + 20 + 21 + 23

⇒ **CQ = 114**

CTQ be the total number of accesses to attributes by applications that access only TA **[OV99]** then we have,

⇒ CTQ = (0+15+5) + (10+5+6) + (8+3+12)

⇒ CTQ = 20 + 21 + 23

⇒ **CTQ = 64**

Copyright © MSc Software Engineering 2003-2004

CBQ be the total number of accesses to attributes by applications that access only BA **[OV99]** then we have,

$\Rightarrow$ CBQ = (5+10+3)

$\Rightarrow$ **CBQ = 18**

COQ be the total number of accesses to attributes by applications that access both TA **[OV99]** and BA then we have,

$\Rightarrow$ CBQ = (15+7+10)

$\Rightarrow$ **COQ = 32**

Then the point along diagonal will be

$\Rightarrow Z = CTQ * CBQ - COQ^2$

$\Rightarrow Z = 64 * 18 - 32^2$

$\Rightarrow$ **Z = 128**

Hence, Z is maximized for this point; so we will split our matrix from this point into 2 parts.

4.3.7 Fragmented STAFF table

After calculation, we can now divide the STAFF relation into 2 fragments as below:

$STAFF1 = \Pi_{stno, fname, lname, position, brno, sup\_id}(STAFF)$
$STAFF2 = \Pi_{stno, address, salary, datehired, dateCurrLoc, sex, dob, bonus}(STAFF)$

Copyright © MSc Software Engineering 2003-2004

STAFF1					
StNo	**FName**	**LName**	**Position**	**SupNo**	**BrNo**
1	n/a	n/a	n/a	n/a	n/a
2	Mark	William	Manager	n/a	B001
3	John	Wright	Supervisor	Mark	B001
4	Anna	Margaret	Assistant	John	B001
5	William	Blair	Assistant	John	B001
6	Tim	Ray	Manager	n/a	B002
7	Jason	McGrath	Supervisor	Tim	B002
8	Warwick	Henry	Assistant	Jason	B002
9	Michelle	Kamber	Assistant	Jason	B002
10	Johnny	Blade	Manager	n/a	B003
11	Graeme	Tom	Supervisor	Johnny	B003
12	Steve	Wonder	Assistant	Graeme	B003
13	Sarah	Johnson	Assistant	Graeme	B003
14	Nancy	Elizabeth	Manager	Tim	B004
15	Matthew	Waugh	Supervisor	Nancy	B004
16	Justin	Walker	Assistant	Matthew	B004

STAFF2							
StNo	**Address**	**Salary**	**DateHired**	**Sex**	**DOB**	**DateCurrLoc**	**Bonus**
1	n/a	£0.00		n/a			£0.00
2	123 South Road, Bermondsey	£400.00	15/10/2002	Male	19/05/1969	15/10/2002	£0.00
3	419 Old Kent Road, Peckham	£300.00	01/01/2003	Male	23/08/1978	01/01/2003	£0.00
4	21 Holloway Road, Islington	£200.00	01/01/2003	Female	15/09/1979	01/01/2003	£0.00
5	65 Tower Bridge Road, Tower Hamlets	£200.00	01/02/2003	Male	23/04/1974	01/02/2002	£0.00
6	72 High Street, Whitechapel	£400.00	01/03/2003	Male	14/02/1964	01/03/2003	£0.00
7	12 North Street	£300.00	15/03/2003	Male	21/08/1976	15/10/2002	£0.00
8	9 Denmark Hill	£200.00	01/04/2003	Male	04/10/1974	01/04/2003	£0.00
9	21 Peckham Road	£200.00	01/04/2003	Female	10/05/1977	01/04/2003	£0.00
10	65 Tower Bridge Road	£400.00	01/01/2003	Male	11/07/1966	01/01/2003	£0.00
11	78 Abbey Street	£300.00	15/03/2003	Male	21/08/1976	15/10/2002	£0.00
12	106 Tooley Street	£200.00	01/04/2003	Male	04/10/1974	01/04/2003	£0.00
13	174 Woodford Avenue	£200.00	01/04/2003	Female	10/05/1977	01/04/2003	£0.00
14	137 Avondale Rise	£400.00	01/06/2003	Female	18/02/1968	01/06/2003	£0.00
15	344 Cheap Street	£300.00	15/06/2003	Male	21/11/1975	15/06/2003	£0.00
16	43 Way-to-Go Road	£200.00	15/06/2003	Male	08/04/1978	15/06/2003	£0.00

Copyright © MSc Software Engineering 2003-2004

4.3.8 Checking for Correctness

1. Completeness

It is guaranteed by partitioning algorithm since each attribute A of the global relation R is divided into different fragments **[OV99]**.

$$A = U\ Ri$$

So for our STAFF fragments, if we join STAFF1 and STAFF2, we will get the original number of attributes as in global relation STAFF

$$A = STAFF1\ U\ STAFF2$$

2. Reconstruction

Reconstruction of the global relation from its fragments is performed by the join operation in vertical fragmentation. For example for relation R with fragments $F_R = \{R_1, R_2, R_3, ..., R_n\}$ and key attributes k **[OV99]**, we have

$$R = \infty_k R_i \ \forall\ R_i \in F_R$$

As in our case, we have 2 fragments defined on *STAFF*, as F_{STAFF} = {STAFF1, STAFF2}, so we have

$$STAFF = STAFF1 \infty_{stno} STAFF2$$

Thus, if we unite our 2 fragments, we get the actual relation, so our fragments are correct.

3. Disjointness

As we have key attribute in all fragments, so disjoint characteristic states that if we have different, mutually exclusive non-key attributes in all fragments, then we satisfy the disjointness criteria **[OV99]**.

In our case, we have STNO in both fragments; all other attributes are mutually exclusive. Hence our fragments are correct.

4.3.9 Summary

Thus it is clear from the calculations that Vertical fragmentation technique is very complex and explosive in nature. We have to go through a lengthy process of finding the suitable fragmentation scheme. The Bond Energy Algorithm and Partitioning Algorithm, especially, are very complex but results in the perfect order of the fragments and helps in achieving the vertical fragmentation successfully.

Copyright © MSc Software Engineering 2003-2004

5. DATA ALLOCATION AND REPLICATION

5.1 Data Allocation

As we have 4 different branches for our case study according to our assumptions, so we have 4 sites that will have data. So each fragment will be stored at each location. The data of all branches will be allocated to each main branch of that city.

Suppose we have a fragment BRAN2, which contains data about Glasgow branch, that fragment will be stored on the server at the Glasgow branch.

In future, if we plan to open another branch in Glasgow, then that record will be entered in that fragment, and an updated copy of that data will also be available for use to that new branch.

5.2 Data Replication

Database replication is the process of sharing data or database design changes between databases in different locations without having to copy the entire database. Replication involves copying a database so that two or more copies of a single database remain synchronized. The original database is called a *Design Master* and each copy of the database is called a *Replica*. Together, the Design Master and the replicas make up a *Replica Set* **[SS00]**.

Synchronization is the process of ensuring that every copy of the database contains the same objects and data. When you synchronize the replicas in a replica set, only the data that has changed is updated **[SS00]**. There are 3 types of replication **[SS00]**.

5.2.1 Types

3 types of replication are:

1. Snapshot
2. Transactional
3. Merge

5.2.1.1 Snapshot Replication

Snapshot replication copies data or database objects exactly as they exist at any moment. Snapshot publications are typically defined to happen on a scheduled basis. Snapshot replication is used where the source data is relatively static and the amount of data to replicate is small **[SS00]**.

Copyright © MSc Software Engineering 2003-2004

Snapshot replication is often used when needing to browse data such as price lists, online catalogues, or data for decision support, where the most current data is not essential and the data are used as read-only **[SS00].**

Benefits

Following are some of the scenarios where snapshot replication fits in ideally **[SS00]**:

- Data/Db objects are static or do not change frequently
- Replicate Look Up tables that do not change frequently
- The amount of data to be replicated is small
- Users often work in disconnected mode, and are not always interested in the latest data.

5.2.1.2 Transactional Replication

Transactional replication is also known as dynamic replication. In transactional replication, modifications to the publication at the publisher are propagated to the subscriber incrementally **[SS00]**.

Benefits

Following are some of the benefits of transactional replication **[SS00].**

- Replicating Database with rollup information, Database with regional, central sales or inventory database that is updated and replicated to different sites.
- Subscribers always need the latest data for processing.

5.2.1.3 Merge Replication

Merge replication provides advantages of both Snapshot replication and Transactional replication. The data is synchronized on a scheduled basis or on demand **[SS00]**.

Benefits

Following are some of the benefits of merge replication **[SS00].**

- Updates to the data are made independently at more than one server.
- Data is merged on a scheduled basis or on demand.
- Allows users to work online/offline and synchronize data on a scheduled basis or on demand.
- Site autonomy is very critical.

Copyright © MSc Software Engineering 2003-2004

5.2.2 Planning for Replication

Careful planning before replication deployment can maximize data consistency, minimize demands on network resources, and prevent troubleshooting later **[SS00]**.

Consider these areas when planning for replication **[SS00]**:

- Whether replicated data needs to be updated, and by whom
- Your data distribution needs regarding consistency, autonomy, and latency
- The replication environment, including business users, technical infrastructure, network and security, and data characteristics
- Types of replication and replication options
- Replication topologies and how they align with the types of replication

5.2.3 Replication and DreamHome

Because of the benefits of **Snapshot Replication** and its usage, as mentioned above, this strategy suits our scenario the most. Thus, we will update our replica sets after the end of every business day. The replica of fragments for every relation will be updated on day-to-day basis.

Similarly, the location of the replicated fragments will be stored at alternative locations. For example, the replica of Glasgow branch will be stored in London, the replica for Aberdeen will be stored in Glasgow and for London data will be replicated in Aberdeen.

Copyright © MSc Software Engineering 2003-2004

6. IMPLEMENTATION

6.1 User Interface

For developing the user interface we use Active Server Page (ASP) as front-end technology for our application and Microsoft Access database for our back-end data storage. Thus we implement our case study as a Web-based interface, which can be applied on any intranet or internet scenario.

We develop different sections that are divided into Properties, Branches, Staff, Clients, Owners, Leases, Ads and Property Views. The first of our interface looks like this:

Fig 29: Homepage for DreamHome web-based interface

Now we will discuss few sections of our interface with different functionalities.

6.2 Data Viewing

Different pages are provided to view the data of all of our fragments from the database individually and collectively. By simple clicking on the links on the navigation bar, users can easily navigate between different pages of the website. Next figure shows the Introduction page of the website

Copyright © MSc Software Engineering 2003-2004

Fig 30: Welcome page for DreamHome

Now by clicking on any of the links on left-hand side, user can easily go to any part of the website. Suppose user clicks on the Properties link, then page immediately redirects to it and fetch all the record form the database from different fragments, combines it and shows to the user. Next figure shows the Properties data as shown on the website:

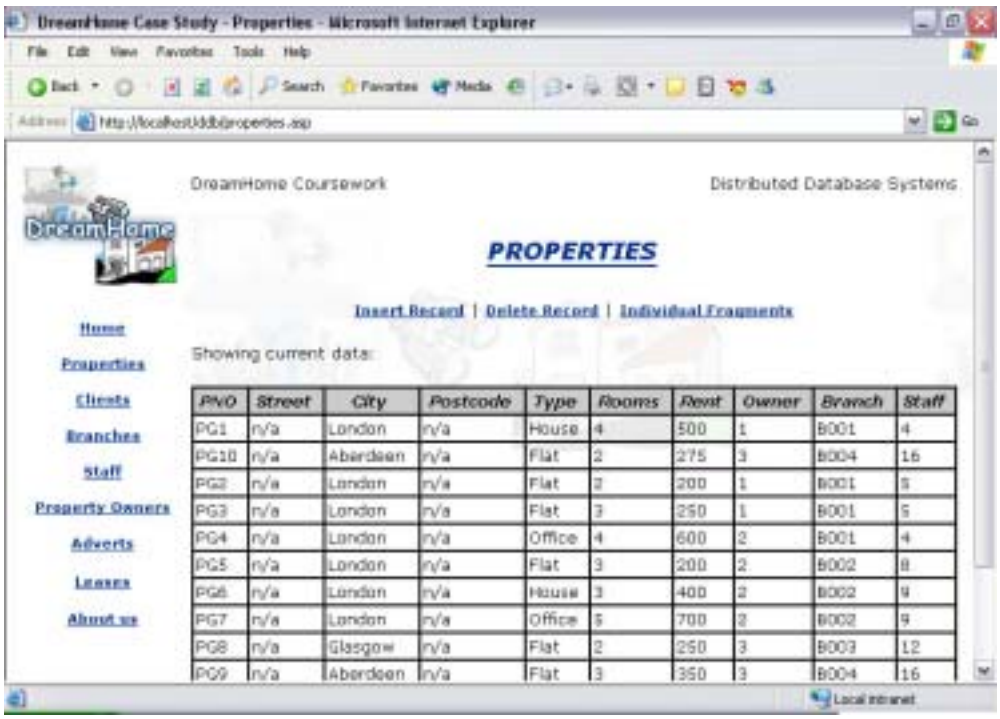


Fig 31: Properties view page for DreamHome

Copyright © MSc Software Engineering 2003-2004

Further user can view data individually coming from different fragments by click on the Individual Fragments link of the relevant page. Next figure demonstrates this idea about the Properties fragments.

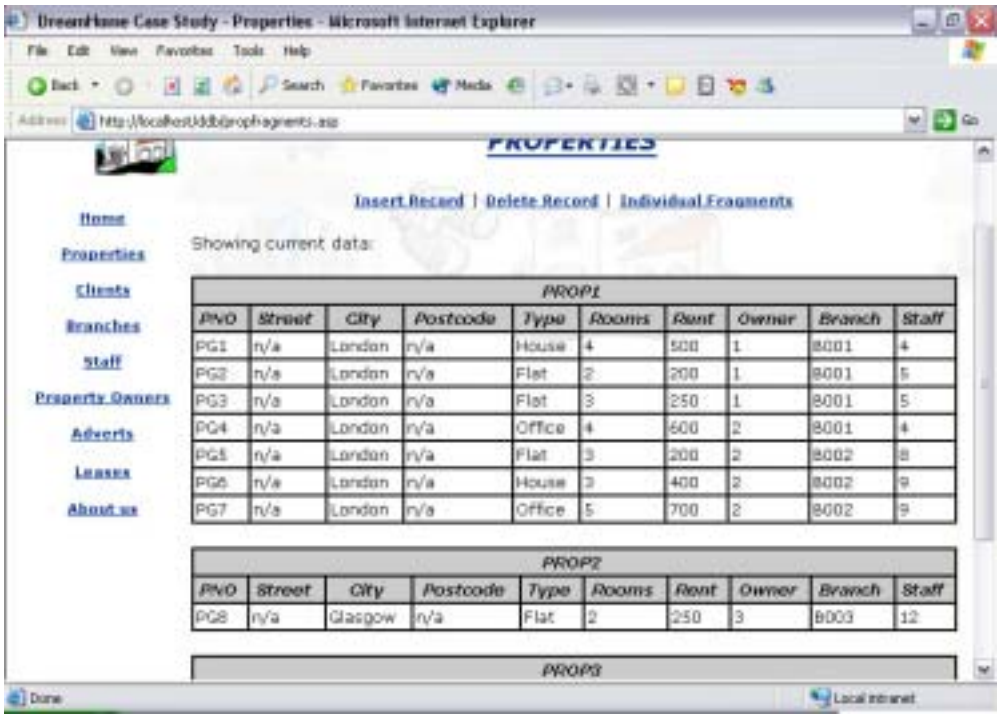


Fig 32: Properties fragments data page

6.3 Data Insertion

Further on any of the data viewing sections, links are provided for data insertion for that section. Suppose we are in Properties section, then by clicking on the Insert Records link, user will goes to the data insertion page for the Properties.

On simple forms provided, users can easily insert new records for each section. On pressing the Insert button of the forms, the data goes to the corresponding fragments of the database.

Suppose user wants to enter some property data for the London branch, then that data will be inserted in the PROP1 fragment, and not in the PROP2 or PROP3, because PROP1 contains the data that corresponds to the London records.

Next figure shows the form that can be used to insert new data for the Properties.

Copyright © MSc Software Engineering 2003-2004

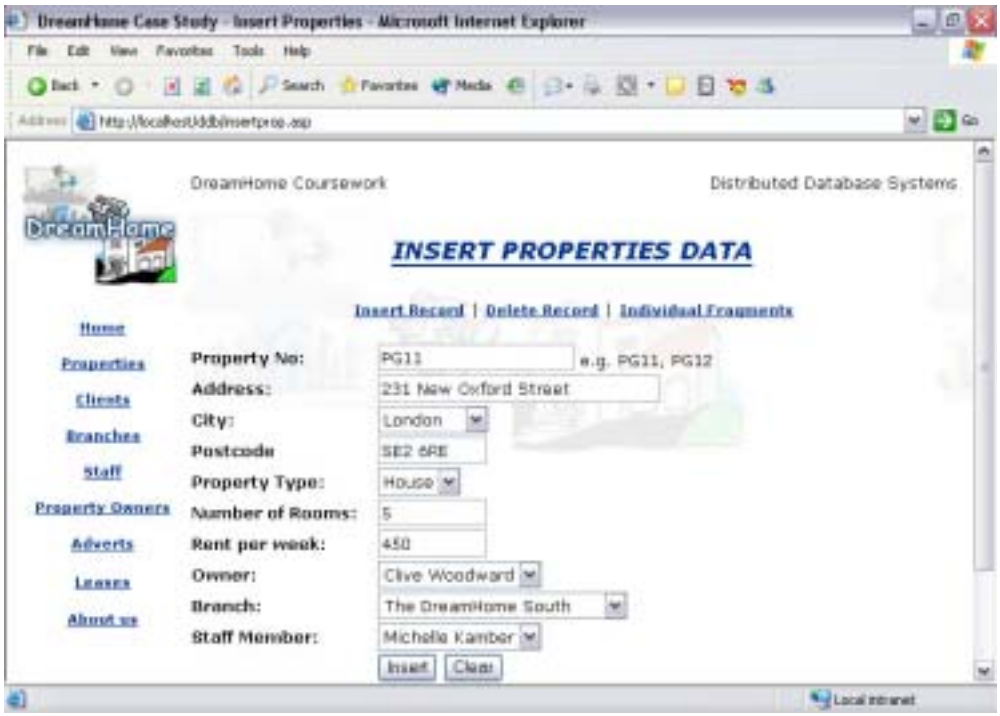


Fig 33: Insert records page for Properties

After insertion of data, user will get the confirmation that data inserted successfully. Next figure illustrate this idea.

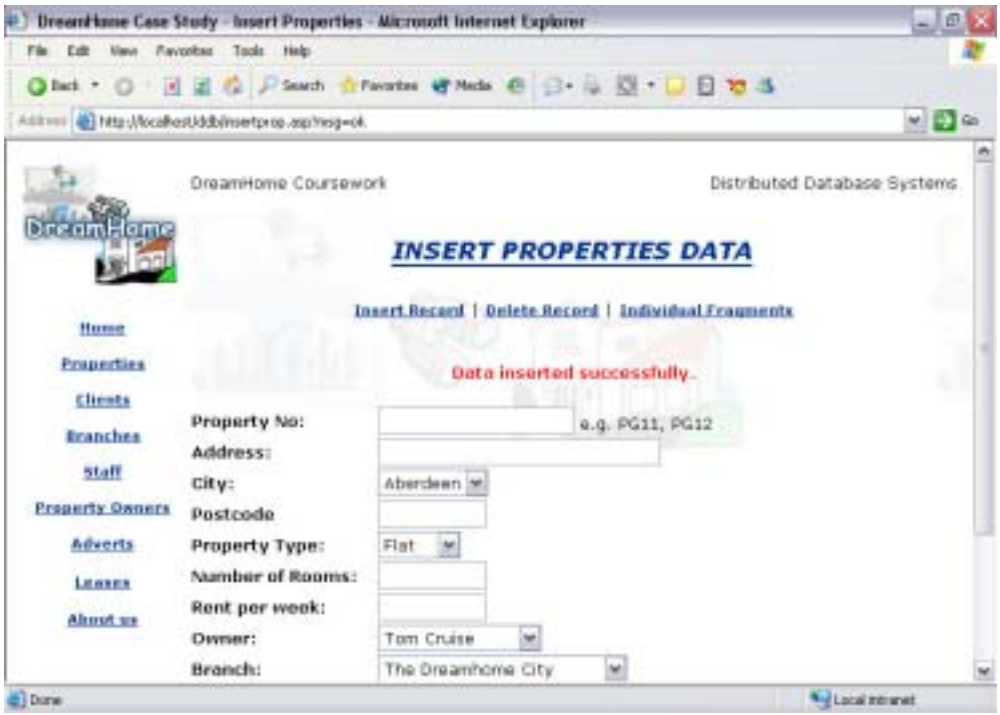


Fig 34: Insert record confirmation for Properties

Copyright © MSc Software Engineering 2003-2004

6.4 Data Deletion

Similar to insertion of records, we also provide the link for deleting the records for the database. By clicking on the Delete Records link, users will options of deleting records. Following figure shows delete records page for Properties.

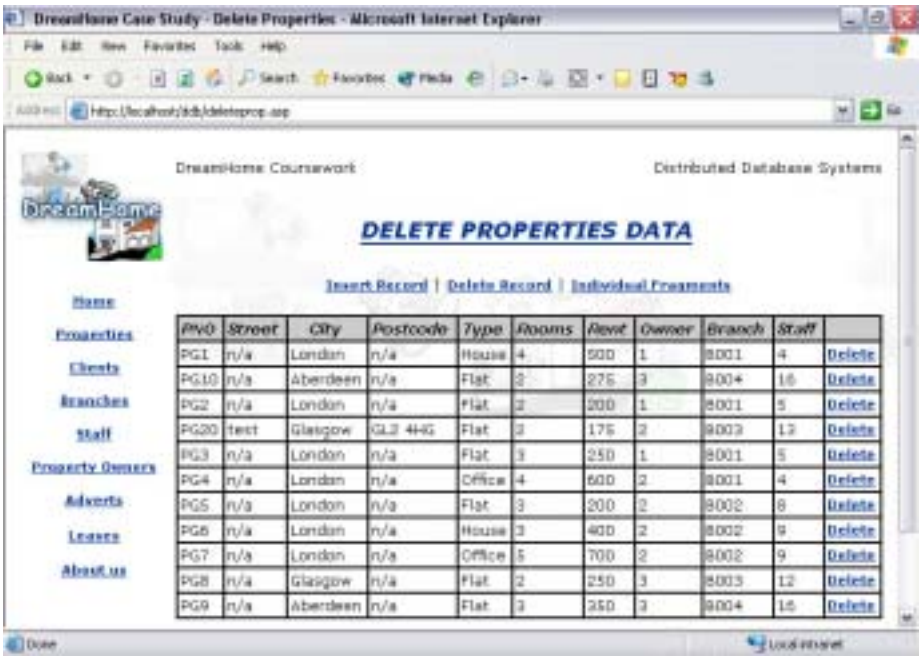


Fig 35: Delete records page for Properties

By selecting the records to delete, user can click on the corresponding Delete link to delete that record from the database. Record will be deleted from the specific fragment where that record exists. Next figure shows successful deletion of record.

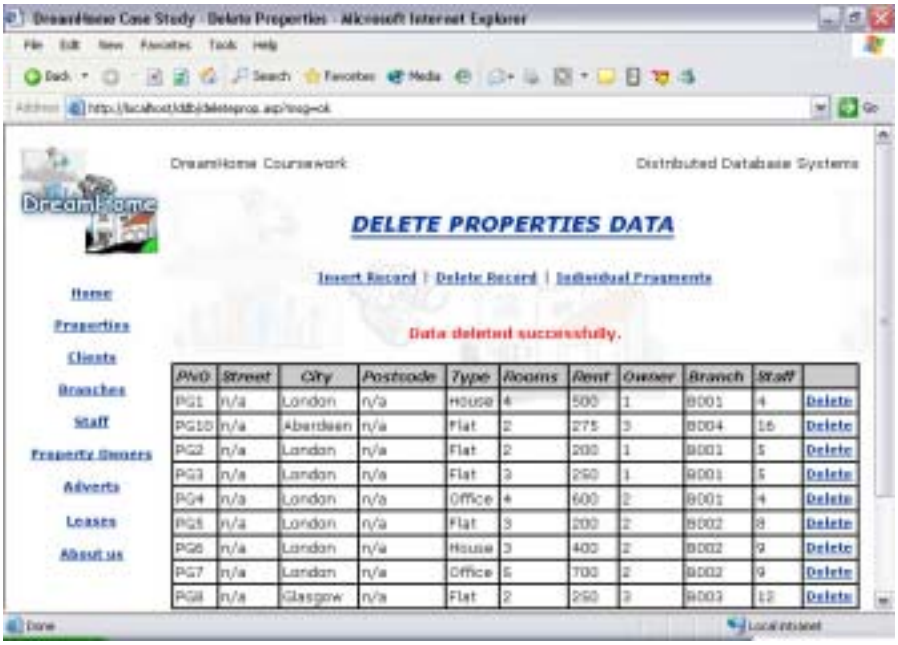


Fig 36: Delete record confirmation for Properties

Copyright © MSc Software Engineering 2003-2004

6.5 Summary

Similar to Properties, every section has same functionality of viewing, insertion and deletion of the record. The user just view, insert and delete in the normal fashion but on the back end, data manipulated and gathered from different fragments. User cannot see that complexity of the data. Hence user will remain transparent to the fragmentation of the tables and feels that he/she is working on one single relation of the database.

Copyright © MSc Software Engineering 2003-2004

7. REFERENCES AND BIBLIOGRAPHY

- **[OV99]** Principles of Distributed Database Systems – Second Edition, M. Tamer Ozsu, Patrick Valduriez, Prentice Hall Publishers, 1999

- **[IDDT]** IEEE System Design Document Template

- **[CS03]** Users' Requirements Specification for DreamHome Case Study (provided by the lecturer, Dr. Farhi Marir)

- **[SS00]** Website (Visited on: Tuesday, January 06, 2004 6:15 PM, URL: http://www.devarticles.com/c/a/SQL_Server/Replication_SQL_Server_2000__Part_1/4

- http://www.hospitalitysuite.com/HSHardwareSpecs/MinimumHardwareSpecs/Fat-ClientWorkstation/RecommendedHardwareSpecifications-FatClientWorkstation-version1.07.pdf (Wednesday, December 03, 2003 10:15 PM)

- http://www.hospitalitysuite.com/HSHardwareSpecs/TraditionalClientServer.htm (Wednesday, December 03, 2003 11:15 PM)

- http://www.istis.unomaha.edu/isqa/vanvliet/arch/minipres/Vellore.pdf (Wednesday, December 03, 2003 10:40 PM)

- http://www.networkcomputing.com/netdesign/1005part1a.html (Wednesday, December 03, 2003 11:45 PM)

- IEEE System Design Document Template

- Software Engineering, Roger S. Pressman, Addison Wesley 1992 (3rd Edition)

Copyright © MSc Software Engineering 2003-2004