2008

# Intelligent query for real estate search

Mandeep Jandir
*San Jose State University*

# INTELLIGENT QUERY FOR REAL ESTATE SEARCH

A Thesis

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Mandeep Jandir

August 2008

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

[illegible]

Dr. Chris Tseng

[illegible]

Dr. Chris Pollett

[illegible]

Dr. Sami Khuri

APPROVED FOR THE UNIVERSITY

[illegible] 07/15/08

ABSTRACT

INTELLIGENT QUERY FOR REAL ESTATE SEARCH

by Mandeep Jandir

The purpose of this project is to improve search query accuracy in a real estate website by developing an intelligent query system which provides the best matching result for standard search criteria. This intelligent query website utilizes fuzzy logic and partial membership to filter query results based on user input data. Fuzzy logic helps obtain results that are otherwise not attainable from a non-fuzzy search. A non-fuzzy search entails search results that match exactly with the given criteria. This project also allows a user to do a free keyword search. This type of search uses synonyms of the keywords to query for houses. The resulting information is more credible and precise than the traditional website because it provides a reasonable result, of the specified search, to the user.

# ACKNOWLEDGEMENTS

I would like to express my appreciation to Dr. Chris Tseng who provided the motivation, resources, and invaluable insights without which this project could never have been accomplished.

I would also like to express my gratitude to my committee members, Dr. Chris Pollett and Dr. Sami Khuri, for their useful comments on my work.

Finally, I would like to thank my parents, sister, and brother for their continuous encouragement and support.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

## LIST OF TABLES

## LIST OF FIGURES

# INTRODUCTION

Real estate websites are gradually becoming more popular on the World Wide Web. These websites are like search engines for finding properties, which are useful for users who want to buy homes or other property. However, there are no objective ways for retrieving search results based on personalized data. If we take a look at Zillow, REIL, MLSListings, ZipListing, and Prudential California Realty websites, we can find that all of these sites utilize inputs from the standard fields to retrieve search results for any type of property desired. But, these mentioned websites return strict search results based on user request. For instance, if a user wants to buy a *4 bedroom, 2-car garage house,* the search result will only retrieve *4 bedroom, 2-car garage houses;* whereas a fuzzy logic real estate system will retrieve *5 bedroom, 2-car garage houses* or *4 bedroom, 3-car garage houses,* etc. depending on how and which membership function is applied to the standard fields to obtain partial membership for the linguistic terms.

The above mentioned websites – Zillow, REIL, MLSListings, ZipListing, and Prudential California Realty – do not have the option that allows the user to do a free keyword search for real estate properties. A keyword search feature allows the user to search for houses based on personal preference rather than utilizing the standard search fields. For example, a user may want to search for homes with *big rooms,* for which the search result will return house properties with the original keyword *big rooms* along with synonymous keywords such as *large rooms, spacious rooms, huge rooms, enormous rooms,* etc.

Therefore, a more objective way of utilizing a user's search inputs has been implemented. By using fuzzy logic on queries for real estate search, we are able to filter and retrieve meaningful results which are close to what the user requests. On the other hand, in the traditional query, there is no real matching result for the search specified by the user via the standard search fields or the synonymous search bar. Thus, fuzzy logic returns a matching result if there exists one and therefore, it does an equal or better query to find the results.

We begin by presenting in Section 2 an overview of the background and related work – how fuzzy intelligence can be used to perform a better search result for the real estate website domain. In Section 3, the design of this project will be described and the implementation will be discussed in Section 4. The results of this project will be compared in Section 5. This will be followed by a future work segment of the project in Section 6.

## BACKGROUND AND RELATED WORK

In the past few years, there has been an increase in applications of fuzzy logic and the Internet, especially in intelligent search engines. Fuzzy logic allows for "making useful, human-understandable, deductions from semi-structured information readily available on the web" (NikRavesh, 2002). For this proposed project, we have utilized a conventional DBMS (MySQL) and integrated a fuzzy logic based querying system (jFuzzyLogic).

The real estate websites store insurmountable amounts of information for different types of properties in databases. We have noticed that there seems to be an inherent discrepancy between the "hard" machine and the "soft" human being (Kacprzyk & Zadrozny, 1994). Fuzzy logic has played a crucial role in making it possible to significantly improve the interfaces by offering formal means to deal with vagueness resulting from the utilization of the natural language (Kacprzyk, Owsinski, & Zadrozny, 2001).

It is important to know how Zadeh's paradigm of computing works: linguistic values (e.g. low), linguistic relations (e.g. much less than half), linguistic modifiers (e.g. very), and linguistic quantifiers (e.g. most) can be utilized in developing a more human-consistent and human-friendly querying interface to DBMSs (Kacprzyk & Zadrozny, 2001). Basically, this type of querying system will allow for queries of the type "find (all) property such that most (or any other suitable linguistic quantifier) of the important attributes (such as price, number of bedrooms/bathrooms, etc...) are as specified (equal to 3, greater than $400,000, much less than 2500 square feet, etc...)" (Kacprzyk & Zadrozny, 1996).

The main problem is how to extend a query language so as to allow for the use of the fuzzy terms, including: low, much greater than, most, etc. The linguistic terms are identified by membership functions on a fixed interval, thus allowing context-independent classifications. The membership functions that can be utilized for any fuzzy logic system are Trapezoidal, Gaussian, Possibility, or PossibilityTest.

A fuzzy query system basically allows the user to write SQL with fuzzy matching and makes it feasible to use fuzzy elements in queries to help facilitate the use of a DBMS (Kacprzyk & Zadrozny, 1996). The membership functions of a fuzzy linguistic quantifier utilizes a piecewise linear graph, which signifies that there needs to be two numbers provided. Basically, the vocabulary for querying languages is extended with the use of linguistic terms which allows for query conditions, such as: "temp is high" or "income is much greater than expenditures."

A fuzzy querying engine known as FQuery provides a way to process the fuzzy queries mentioned above. FQuery is built on top of the Microsoft Access database management system. However, obtaining this fuzzy engine was not feasible. After receiving no response from numerous email attempts to the author of this engine, for attaining FQuery, a search for another fuzzy logic system followed. Most of the fuzzy logic systems described in various papers was theoretical. Hence, the fuzzy logic systems mentioned in the literature were not a functional system readily available to use for this project. After searching for a lot of different fuzzy systems, the most promising system discovered was the jFuzzyLogic package software. Section 4 describes this fuzzy logic system in detail.

## *Experience in Researching Fuzzy SQL*

Initially, a lot of literature pertaining to fuzzy logic in SQL was obtained for this topic. Despite finding lots of information on the web, there was some difficulty in finding the corresponding systems and software mentioned in the literature to use for this project. As mentioned above, several email attempts, to some of the authors who wrote the papers on introducing a fuzzy system for any application, went in vain. No response to those emails was received, regarding how to obtain a copy of that particular software mentioned in the paper. An observation was made; that most of the papers which mentioned an implementation of a fuzzy software system were written by those who lived outside of the USA.

Upon researching, we noticed that we could only study in finding information on fuzzy SQL; we could not search for anything based on fuzzy in real estate since it has never been done before. There is no such literature written on including fuzzy logic in real estate search websites. All the references, mentioned in Section 8 of this paper, which we found and read for our project, give detailed explanations on how to include fuzzy logic in DBMS.

Another important piece of information that came across during the reviewing of all the literature was that the dates for all the writings collected, varied drastically. The range for the papers that were gathered is twenty-seven years, 1978 – 2005. And all these pieces of work contained very important relevant information for obtaining fuzzy logic in DBMS.

The majority of the literature obtained and reviewed was written by Janusz Kacprzyk. Kacprzyk is a professor at the Systems Research Institute at

Polish Academy of Sciences in Warsaw, Poland. He is the person who implemented FQuery – an add-on for "Microsoft Access which makes it possible to use queries involving fuzzy values and relations as well as non-standard aggregation operators" (Kacprzyk & Zadrozny, 1996). The next section discusses the design part of the project.

## DESIGN

In this section we describe how we have designed this project. We discuss the usage of the fuzzy logic system, jFuzzyLogic, in our intelligent query system. We also explain what fuzzy inference system and fuzzy control logic is and how they are used in jFuzzyLogic. We also illustrate and describe in detail the architectural model for our developed system.

### *jFuzzyLogic – Fuzzy Logic System*

jFuzzyLogic is a fuzzy logic software package written in Java that is available for everyone to use via the GNU General Public License (GPL). This fuzzy logic package has many features, which include: parametric optimization algorithms, FCL compliance, membership functions, defuzzifiers, rule aggregation, rule connection operators, and rule implication methods.

The different types of parametric optimization algorithms that jFuzzyLogic allows are derivate, gradient descent, and jump. The membership functions that can be utilized are continuous and discrete as well as custom defined membership functions. The different types of continuous membership functions defined in jFuzzyLogic are GenBell, Sigmoidal, Trapezoidal, Gaussian, PieceWiseLinear, and Triangular.

As shown in Figure 1, the continuous membership functions provided by jFuzzyLogic are drawn accordingly. From the legend, m1 refers to a PieceWiseLinear function (blue line), m2 refers to a Triangular function (purple line), m3 refers to a Trapezoidal function (green line), m4 refers to a GenBell function (orange line), m5 refers to a Gaussian function (yellow line), and m6 refers to a Sigmoidal function (red line).

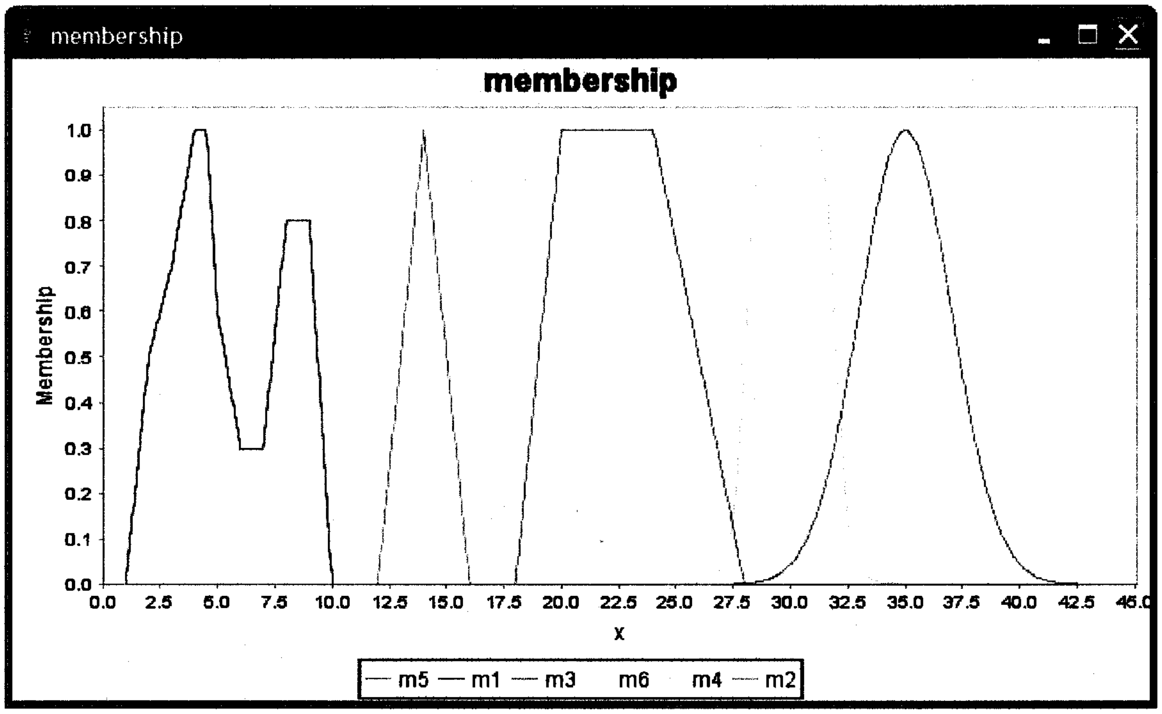


*Figure 1.* Continuous Membership Functions

Each linguistic term in a Triangular membership function is described by three points (min, mid, and max), whereas each linguistic term in a Trapezoidal membership function is described by four points (min, mid low, mid high, and max). The linguistic terms for a Gaussian membership function are defined by two points (mean and standard deviation) and each of the linguistic terms for the GenBell membership function are defined by three points (a, b, and mean). The linguistic terms for Sigmoidal membership functions are described by two points (gain and t0) while the linguistic terms associated with the PieceWiseLinear membership function each have as many (x, y) pair coordinates as needed to define the membership of that variable.

The various types of discrete membership functions described in the fuzzy logic system are Singleton and GenericSingleton. There are also several types of defuzzifiers provided by the jFuzzyLogic system: continuous, discrete, and custom defined defuzzifiers. The continuous defuzzifiers listed for this fuzzy logic system include CenterOfGravity, RightMostMax, CenterOfArea, LeftMostMax, and MeanMax. There is only one type of discrete defuzzifier defined in jFuzzyLogic, which includes CenterOfGravitySingletons. Rule aggregation incorporates how rules are accumulated and there are several ways of achieving aggregation: BoundedSum, Max, ProbOr, Sum, or NormedSum.

The two types of rule connections operators allowed in the jFuzzyLogic system include AND and OR.

The jFuzzyLogic system implements a Fuzzy Inference System (FIS) and Fuzzy Control Logic compliance (FCL). Based on the user-defined FCL file, a user can input variables. The system fuzzifies the values of those input variables and returns a defuzzified output along with membership values and graphs of all the input variables. This jFuzzyLogic system is employed for retrieving membership values of the linguistic terms based on user selected variables from the real estate website. The linguistic terms whose membership values are greater than zero are then utilized to query the database to obtain meaningful search results. The next section discusses the terms FIS and FCL.

### *Fuzzy Inference System (FIS) and Fuzzy Control Logic (FCL)*

Fuzzy Inference is the process utilized to produce a mapping of the given input to an output by using fuzzy logic. The mapping itself proposes a starting point from which decisions can be made, or patterns perceived. The process of fuzzy inference involves membership functions, logic operations, and if-then rules. A fuzzy inference system (FIS) uses fuzzy control logic.

Fuzzy Control Language (FCL) is a language for implementing fuzzy logic, especially fuzzy control. Fuzzy controllers include an input phase, a processing phase, and an output phase. The first phase maps the inputs to the appropriate membership functions and truth values. The second phase first applies each appropriate rule and produces a result for each, and then totals the results of the rules. Lastly, the output phase translates the totaled result back into a precise control output value.

In the jFuzzyLogic system, a fuzzy inference system is created by defining one or more FuzzyRuleSets. Each FuzzyRuleSet is compiled by some FuzzyRules and each FuzzyRule is written using an antecedent (IF part) and a consequent (THEN part). Consequents are a set of FuzzyRuleTerms and an antecedent is denoted by a FuzzyRuleExpression. A FuzzyRuleExpression consists of two terms linked together by a RuleImplicationMethod (rule connectors are AND, OR, and NOT). Each FuzzyRuleTerm is described by a Variable and a LinguisticTermName. Each Variable has a name and some LinguisticTerms. A class diagram representation describing the jFuzzyLogic system is shown in Figure 2.

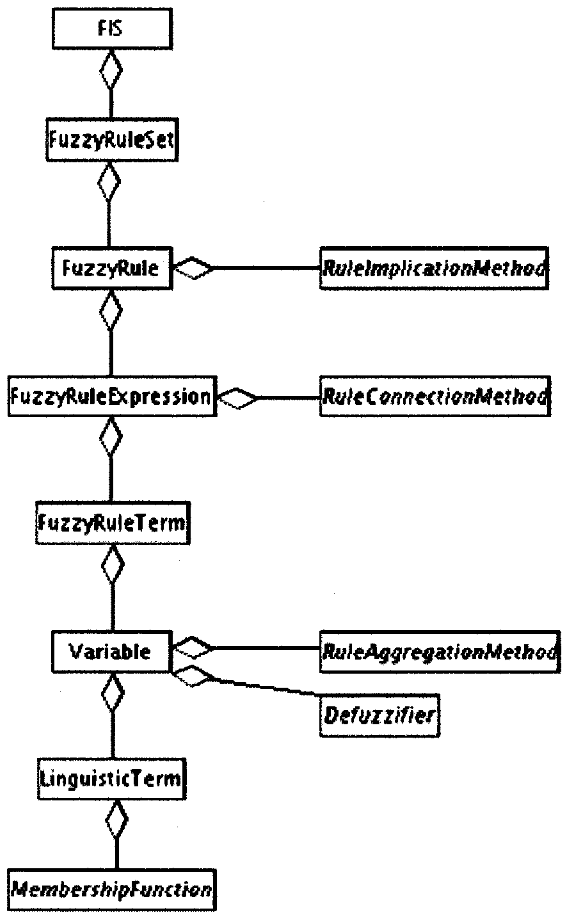


*Figure* 2. Class Diagram for jFuzzyLogic

This jFuzzyLogic system defines a Fuzzy Function Block inside the FCL file and it contains the following type of information shown in Figure 3.

```
FUNCTION_BLOCK Fuzzify
VAR_INPUT
        temperature : REAL;
        pressure    : REAL;
END_VAR
VAR_OUTPUT
        valve       : REAL;
END_VAR
FUZZIFY temperature
        TERM cold := (5, 1) (30, 0);
        TERM hot  := (5, 0) (30, 1);
END_FUZZIFY
FUZZIFY pressure
        TERM low  := (60, 1) (100, 0);
        TERM high := (60, 0) (100, 1);
END_FUZZIFY
DEFUZZIFY valve
        TERM drainage := -100;
        TERM closed   := 0;
        TERM open     := 100;
        ACCU : MAX;
        METHOD : COGS;
        DEFAULT := 0;
END_DEFUZZIFY
RULEBLOCK block1
        AND : MIN;
        RULE 1 : IF temperature IS cold AND pressure IS low THEN valve IS open;
        RULE 2 : IF temperature IS cold AND pressure IS high THEN valve IS closed WITH 0.7;
        RULE 3 : IF temperature IS hot AND pressure IS low THEN valve IS closed;
        RULE 4 : IF temperature IS hot AND pressure IS high THEN valve IS drainage;
END_RULEBLOCK
END_FUNCTION_BLOCK
```

*Figure* 3. Sample FCL file

As seen in the sample FCL file from Figure 3, within each function block, the input and output variables are defined, the variables are converted into degrees of membership by using FUZZIFY blocks and the output is defuzzified using the DEFUZZIFY block based on the rules listed in the RULEBLOCK. For our project, we are only interested in defining input variables and FUZZIFY blocks. The information from the FUZZYIFY blocks will be used by the jFuzzyLogic system to return partial membership values. The exact definition of the FUZZYIFY blocks that we have used for our intelligent search query will be discussed in detail in Section 4.3. The next section describes the architectural model of the intelligent query system developed in this project.

## *Architecture Model of Homes For You*

The architecture model shown in Figure 4 illustrates the functional/logical view of the intelligent query system that we have defined. Based on the model, the user can either utilize the standard search or synonymous search to query for house listings. If a user opts to utilize the standard search, the query is constructed using the membership function values obtained by passing the user specified values to the jFuzzyLogic engine. The membership values used to query for house listings from the database yield the filtered results. These results are then ranked based on the membership values returned by the jFuzzyLogic engine. The ranked output is then displayed for the user to view.

If the user chooses to use the synonymous search, the keyword phrase entered would then be sent to the Encarta synonymous website to obtain the synonyms of the keyword phrase. These synonyms along with the original keyword phrase would then be used to construct the query. When executed in the database, this query would then produce the filtered results. These results would then be ranked with the original keyword phrase at the top of the output list and the synonymous phrases following the original keyword phrase. This ranked output would then be displayed for the user to view.

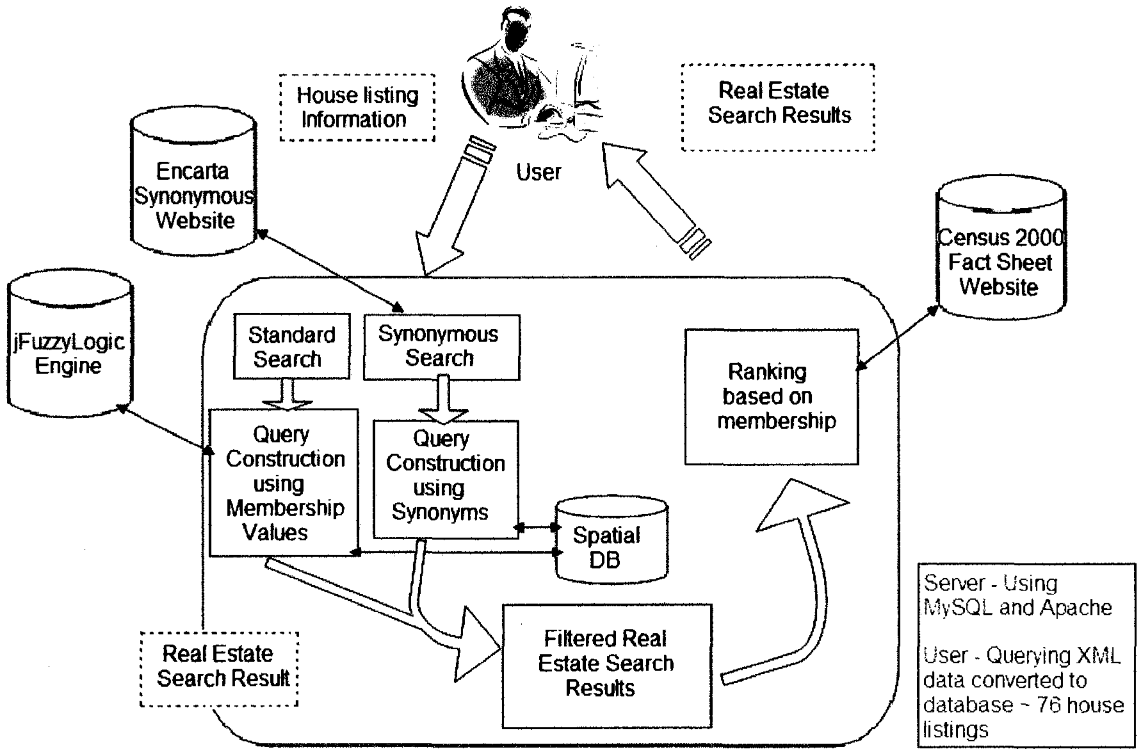

House listing Information
User
Real Estate Search Results
Encarta Synonymous Website
Census 2000 Fact Sheet Website
jFuzzyLogic Engine
Standard Search
Synonymous Search
Query Construction using Membership Values
Query Construction using Synonyms
Ranking based on membership
Spatial DB
Filtered Real Estate Search Results
Real Estate Search Result
Server - Using MySQL and Apache
User - Querying XML data converted to database ~ 76 house listings

*Figure 4.* Architecture Model of *Homes For You* Website

Each house from the ranked house listing result, displays the Census 2000 fact information along with the house property information for the user to view. This census information is retrieved from the fact finder website created by the

U.S. Census Bureau. The statistics obtained from the census website are discussed in Section 4.1. This project has been developed using the Apache server, PHP scripts, and MySQL database. This will be explained further in the next section, which describes the implementation part of this project.

# IMPLEMENTATION

## *Real Estate Website*

Our real estate website *Homes For You* allows for traditional and non-traditional standard search queries for finding homes and other properties. Our website is an enhancement, to the other traditional websites, that allows for fuzzy search based on the standard input selection. For example, if a user wants to buy a *4 bedroom, 3 bath, and 3 car garage* home, the search results will also show a *4 bedroom, 3 bath, and 2 car garage* home amongst the properties available.

For the non-traditional search, users can enter in keywords in a search bar that may not be available as a standard search field option. For example, in a search bar, a user can enter: *hardwood floors* or *big backyard*. And, these keywords will then be analyzed accordingly by utilizing synonyms of the keywords to obtain the search results. The synonymous query will expand the search for particular property and thus, help obtain meaningful results.

The main outline for achieving search results based on the non-traditional search is to find the synonymous keywords from the following columns fields from the database: extra info, cooling, general exterior, more info, flooring, garage, heating, public remarks 1, public remarks 2, agent remarks, kitchen info, location, laundry info, pool, roof, sewer info, bath info, water, land features, special features, and report info. Thus, these column fields are utilized to find the user specified keyword phrase using synonyms.

Another feature of this project incorporates the usage of a zip code, for a particular house listing, to obtain the Census 2000 demographic profile highlights of the area. The fact finder website only allows access to the data from the year 2000, not the data from the year 2006. The site shows data in number and percent values as well as a link to view a map for each of the different types of characteristics. Based on the zip code and pattern matching, the number values are accessed from this website and displayed under a particular house listing description. This census information provides statistics relating to the general, social, economic, and housing characteristics of the chosen area.

The general characteristics segment lists in numbers the total population of male and female, median age (in years), different age groups, different races (White, Black or African American, American Indian and Alaska Native, Asian, Native Hawaiian and Other Pacific Islander, Other), average household size, and the total housing units (owner-occupied housing units, renter-occupied housing units, and vacant housing units) in the area. The social characteristics portion lists in numbers the population that is twenty-five years of age and over – as well as those twenty-five year olds that have a high school degree and a bachelor's degree – civilian veterans, disability status, foreign born, and those who speak a language other than English.

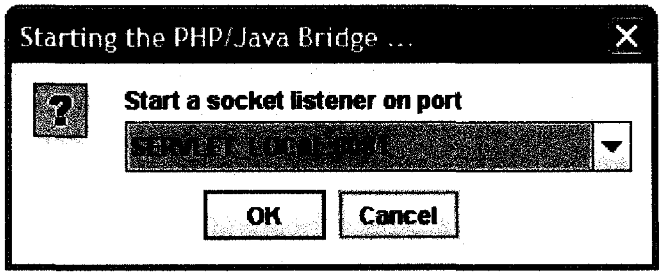
The economic characteristics section lists people in the labor force that are sixteen years of age and over, mean travel time to work in minutes, median household income in 1999 (dollars), median family income in 1999 (dollars), per capita income in 1999 (dollars), families below poverty level, and individuals below poverty level. The housing characteristics segment lists all the single-family owner-occupied homes and the median value (dollars), median of selected monthly owner costs – with a mortgage and without a mortgage.

This census information is incorporated into this project to provide a more descriptive result for the house listing. This additional fact allows the user to obtain general knowledge of the area around a particular property that he/she might be interested in. Certain areas are attracting home owners based on the families already living in that neighborhood. Thus, this extra information for a particular house listing may further provide the incentive to buy this property. In the next section, we describe the PHP configuration for JAVA classes.

*PHP Configuration for Java Classes*

The real estate website, which was created using fuzzy logic, was written in PHP and integrated with the jFuzzyLogic system mentioned in Section 3. The PHP scripts use the Apache Web Server and MySQL database, from the WAMP 2 bundled package. In order to get the PHP files to connect to the JAVA classes, a special bridge known as the PHP/Java Bridge package was downloaded and incorporated into the project. As mentioned in the README file from the PHP/Java Bridge software, "The PHP/Java Bridge is a network protocol which can be used to connect a native script engine (PHP) with a Java VM". The simple and easy to follow directions from the README file of the software package were utilized to setup the PHP/Java Bridge correctly.

To use this bridge, it was described in the README file that we needed to double click on the JavaBridge executable jar file and select a port, as shown in Figure 5.


Starting the PHP/Java Bridge ...

Start a socket listener on port

SERVLET_LOCAL:[illegible]

*Figure 5.* PHP Java Bridge Window

Next, we had to create a jar file containing all the .class files that were required for the project. In this instance, we needed all the .class files of the jFuzzyLogic software. Then, as instructed in the README file, to include the Java class(es) in the PHP file, we had to add the following two lines of code in that PHP file calling the Java class(es):

```
require_once("./JavaBridge/java/Java.inc");
java_require("./jFuzzyLogic/HouseNoPackage.jar;
                 ./jFuzzyLogic/jFuzzyLogic_1_2_1.jar");
```

It is necessary that the relative pathnames used above are according to the location of the Java Bridge folder and while also making sure that the .class files are from the point of view of the PHP script using the java_require function call. One important note that we had to follow is that every time we restarted the server machine, we needed to double click on the JavaBridge.exe file again and select a port in order to keep the functionality of the PHP/Java Bridge working correctly. In the next section, we discuss the real estate data that we have used for our intelligent search query system.

*Real Estate Data*

The data utilized for this real estate search web site is a real data set which was received in an xml file format. The xml document was converted into a database. Along with the conversion, some of the xml field names were renamed to more descriptive column names in the database table because several of the field names consisted of xstr123, xDate4, Feature34, etc. These field names did not correlate with their contents. Thus, the xml field names were renamed according to the contents of these variables. As shown in Table 1, changing the field names from the xml file to more descriptive names was easier to do by basing the new name on the contents. For a complete table listing of all the renamed field names, please refer to Appendix A.

Table 1.

*XML to Database field names conversion*

| XML Field Name | Sample Contents of One Entry | Database Table Column Name |
|---|---|---|
| XSTR72 | Wall to Wall Carpeting, Linoleum or Vinyl, Tile | FLOORING |
| XSTR65 | Concrete Perimeter | OUTSIDE INFO |
| XSTR89 | Built-in Oven/Range Combo, Microwave Oven, 1 Dishwasher, Disposer, 1 Refrigerator | KITCHEN INFO |
| FEATURE20 | 155 Heather Ln | MAILING ADDRESS |
| FEATURE21 | Palo Alto CA | MAILING CITYSTATE |
| FEATURE36 | 4 | BEDROOMS |

The database table contains seventy-six entries of real estate properties, where each entry contains around ninety-three columns of associated data. Some of the important information each house listing contains is: style, original list price, area, type, county, contract date, cross streets, MLS agent id, general exterior, view, fireplace fuel, fireplace location, flooring, finance terms, living dining room information, outside information, garage, heating, insulation, kitchen, public remarks, agent remarks, showing instructions, kitchen information, location, occupant phone, laundry information, pool, roof, sewer information, bath information, parking features, water, land features, special features, report information, days on market, owner name, property address, city, zip code, approx square feet, last transaction date, deed number, zoning, transfer value, year built, tax amount, tax rate area, bedrooms, bathrooms. The next section describes the fuzzy inputs and their associated linguistic terms utilized for this project.

## *Fuzzy Inputs*

For this real estate project, there are five input values required from the standard search fields that have been fuzzified using the jFuzzyLogic software: price of a house, number of beds, number of baths, size of a house in square feet, and lot size of the plot in square feet. In the next section we describe the membership graphs for those user-defined input variables.

### *Fuzzy Rule Set for Price Variable*

The price variable, as shown in Figure 6, has nine linguistic terms. Each term has been composed of a name and a membership function. Each linguistic term is associated with a price, in terms of how much a particular property costs, and has been named according to the price range that the linguistic term covers. The linguistic terms were chosen based on the data; the cutoffs have been based on the minimum and maximum house listing prices from the database. For this variable the membership function that each term uses is the piece-wise linear function.

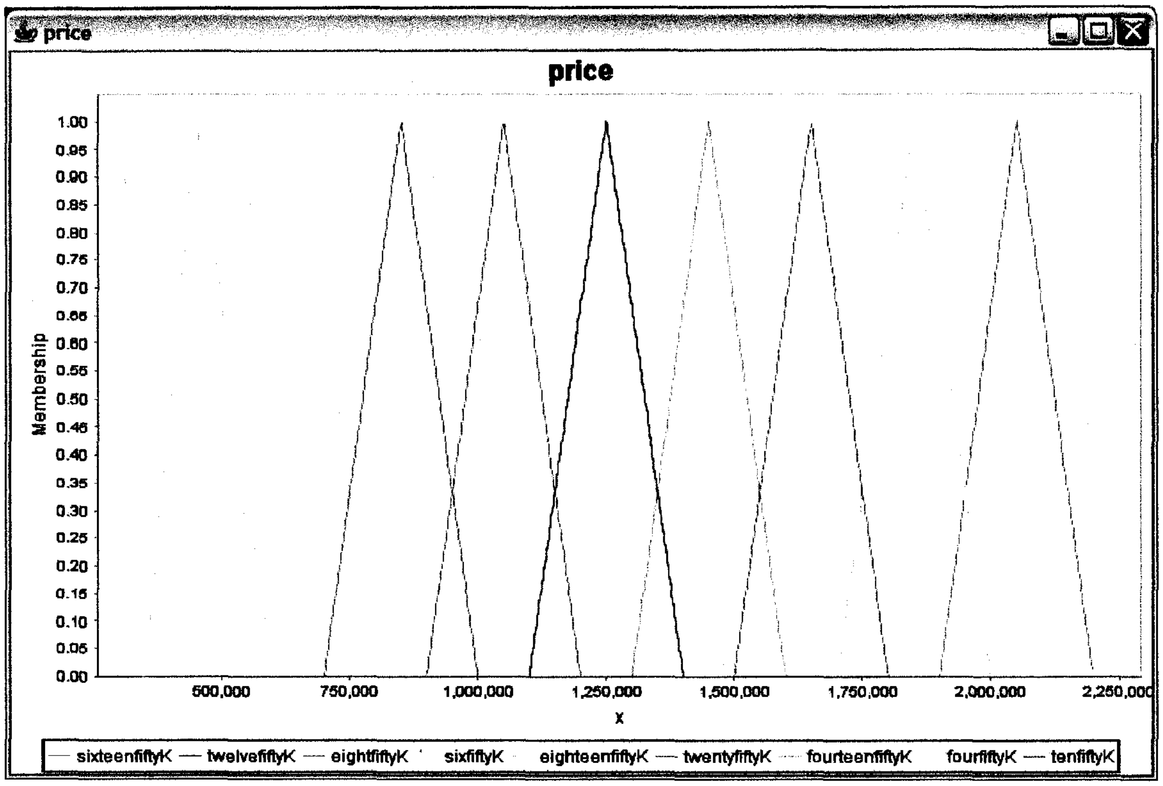


*Figure 6*. Fuzzy Rule Set for Price Variable

The linguistic term names employed for the price variable are: fourfiftyK, sixfiftyK, eightfiftyK, tenfiftyK, twelvefiftyK, fourteenfiftyK, sixteenfiftyK, eigthteenfiftyK, and twentyfiftyK. The K represents thousand, hence fourfiftyK equals to four thousand and fifty (4050). The justification for utilizing such names as linguistic term names is that it signifies that the middle defined point has that value. It also indicates the lower and upper bounds for that term. For example, the first linguistic term is named fourfiftyK, which denotes that the middle point for this term has been defined at (450000, 1).

Based on the design of this variable's linguistic terms, the lower bound has been classified at being 100000 less than the linguistic term value. Hence the lower bound point for linguistic term fourfiftyK is described at (350000, 0). The upper bound for this variable's linguistic terms has been defined to be 150000 more than the linguistic term value. Consequently the upper bound is described at (600000, 0). The next section describes the fuzzy rule set for the bedroom variable.

*Fuzzy Rule Set for Beds Variable*

The beds variable, as shown in Figure 7, has seven linguistic terms. Each of these linguistic variables associates the number of beds that a particular property has and has been named according to the range of bedrooms the house listings has. The linguistic terms were chosen based on the data; the cutoffs have been based on the minimum and maximum number of beds listed in the database. And similarly, as with the above variable, the beds variable also has utilized the piece-wise linear function as its membership function.

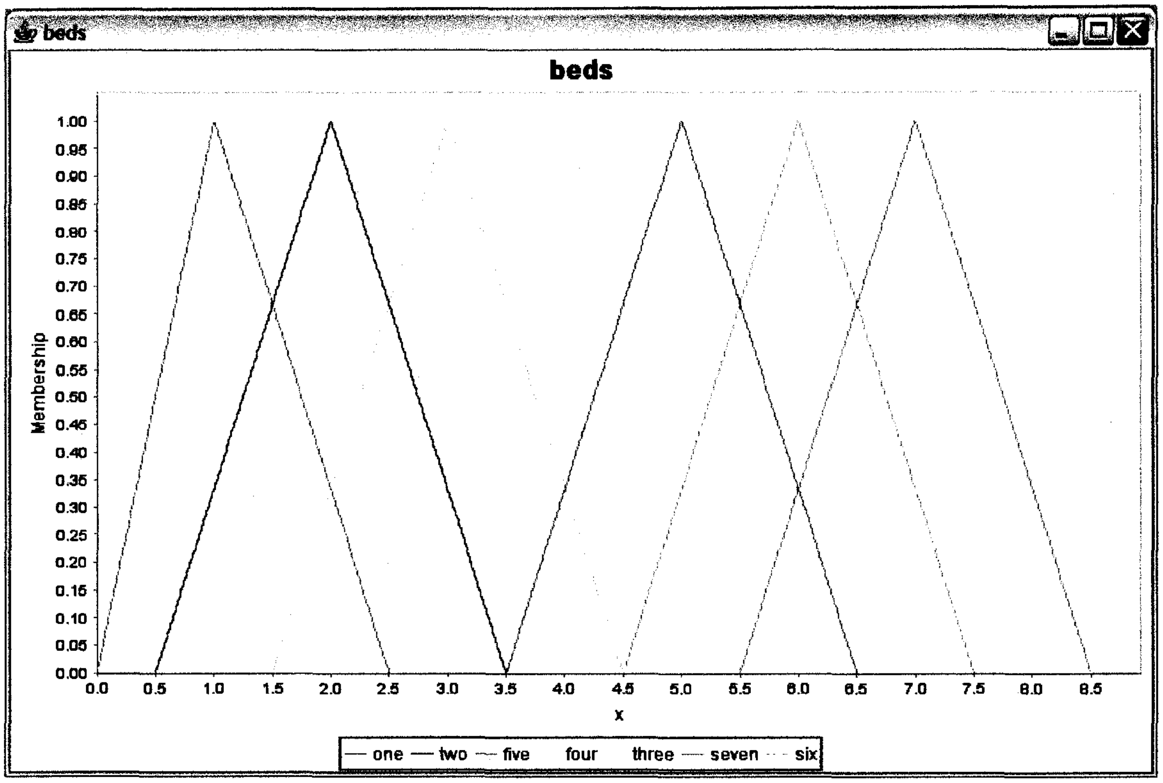


*Figure 7.* Fuzzy Rule Set for Beds Variable

The linguistic term names utilized for the beds variable are: one, two, three, four, five, six, and seven. The justification for using such names as linguistic term names is that the term names signifies that the middle defined point has that value. It also indicates the lower and upper bound for that term. For example, the middle linguistic term has been named four, which denotes that the middle point for this term is at (4, 1). Based on the design of this variable's linguistic terms, the lower bound has been defined as 1.5 less than the linguistic term value. Hence the lower bound point for linguistic term four has been described at (2.5, 0).

The upper bound for this variable's linguistic terms was defined to be 1.5 more than the linguistic term value. Consequently the upper bound has been described at (5.5, 0). The reason for having decimal related lower and upper bounds is so as to include and give partial membership to values exactly one less or one more than the user selected. Going back to the same example, if the user has selected *four-bedrooms,* houses with three- and five-bedrooms would receive partial membership, hence signaling the fuzzy intelligence system that the database also needs to be queried for three and five bedroom houses. The next section discusses the fuzzy rule set for the bathroom variable.

### *Fuzzy Rule Set for Bathrooms Variable*

The bathrooms variable, as shown in Figure 8, also has seven linguistic terms. Each of these linguistic terms associates the number of bathrooms that a particular property has and has been named according to the number of bathrooms that the linguistic term covers. The linguistic terms were chosen based on the data; the cutoffs have been based on the minimum and maximum number of bathrooms listed in the database. This variable also uses the piece-wise linear function as its membership function.

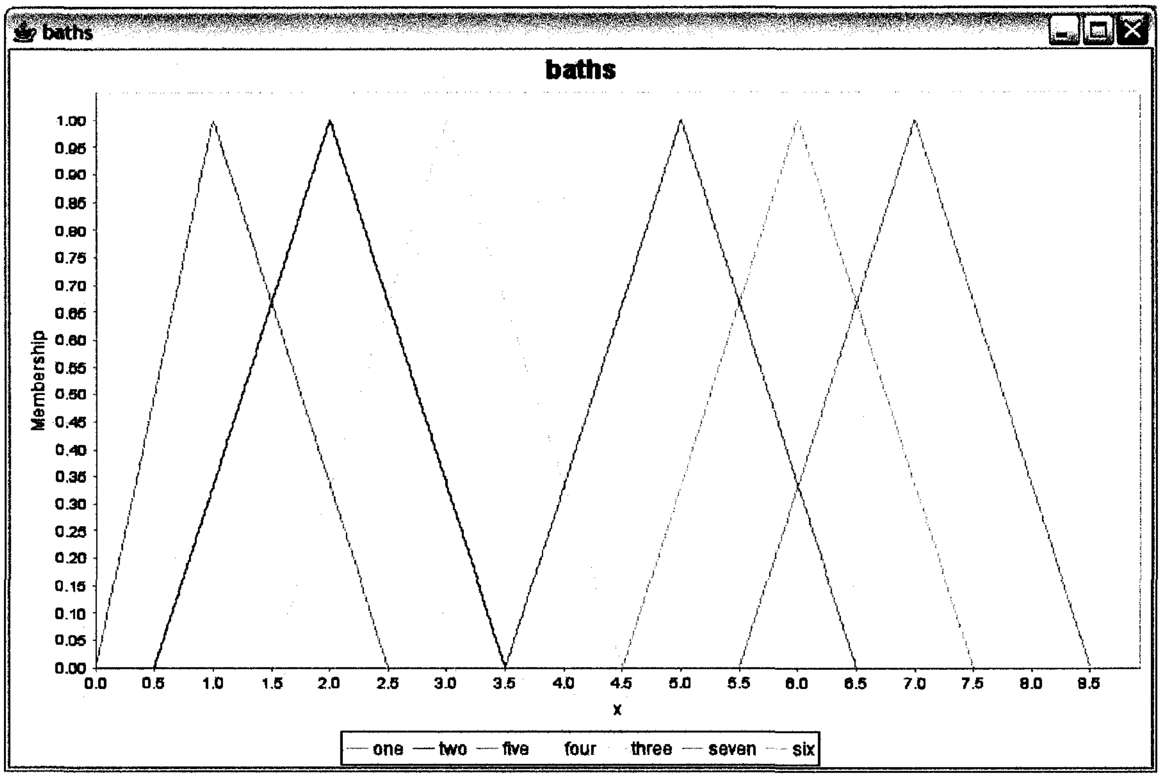


*Figure 8.* Fuzzy Rule Set for Baths Variable

The linguistic term names utilized for the bathrooms variable are the same as for the bedrooms variable: one, two, three, four, five, six, and seven. The justification for using such names as linguistic term names is that the term names signify the middle defined point having that value. It also indicates the lower and upper bound for that term. For example, the middle linguistic term has been named two, which denotes that the middle point for this term is at (2, 1). Based on the design of this variable's linguistic terms, the lower bound has been defined as 1.5 less than the linguistic term value. Hence the lower bound point for linguistic term two has been described at (0.5, 0).

The upper bound for this variable's linguistic terms has been defined to be 1.5 more than the linguistic term value. Consequently the upper bound has been described at (3.5, 0). The same reason as with the bed variables fuzzy rule set has been used for having decimal related lower and upper bounds. It has been used to include partial membership to values exactly one less or one more than the user selected. Going back to the same example, if the user has selected two bathroom properties, houses with one- and three-bathrooms would receive partial membership, hence signaling the newly designed system that along with two bathroom houses, the database also needs to be queried for one- and three-bathroom houses. In the next section, we discuss the fuzzy rule set of the size variable.

### *Fuzzy Rule Set for Size Variable*

The size variable, as shown in Figure 9, has ten linguistic terms, where each linguistic term has been associated to the size of the property. Since there is such a wide range of sizes for any property, there is a need to define this variable by ten linguistic terms. Each linguistic term has been named according to the size range of the house listings. The linguistic terms were chosen based on the data; the cutoffs have been based on the minimum and maximum sizes of the properties listed in the database. This variable also uses the piece-wise linear function as its membership function.

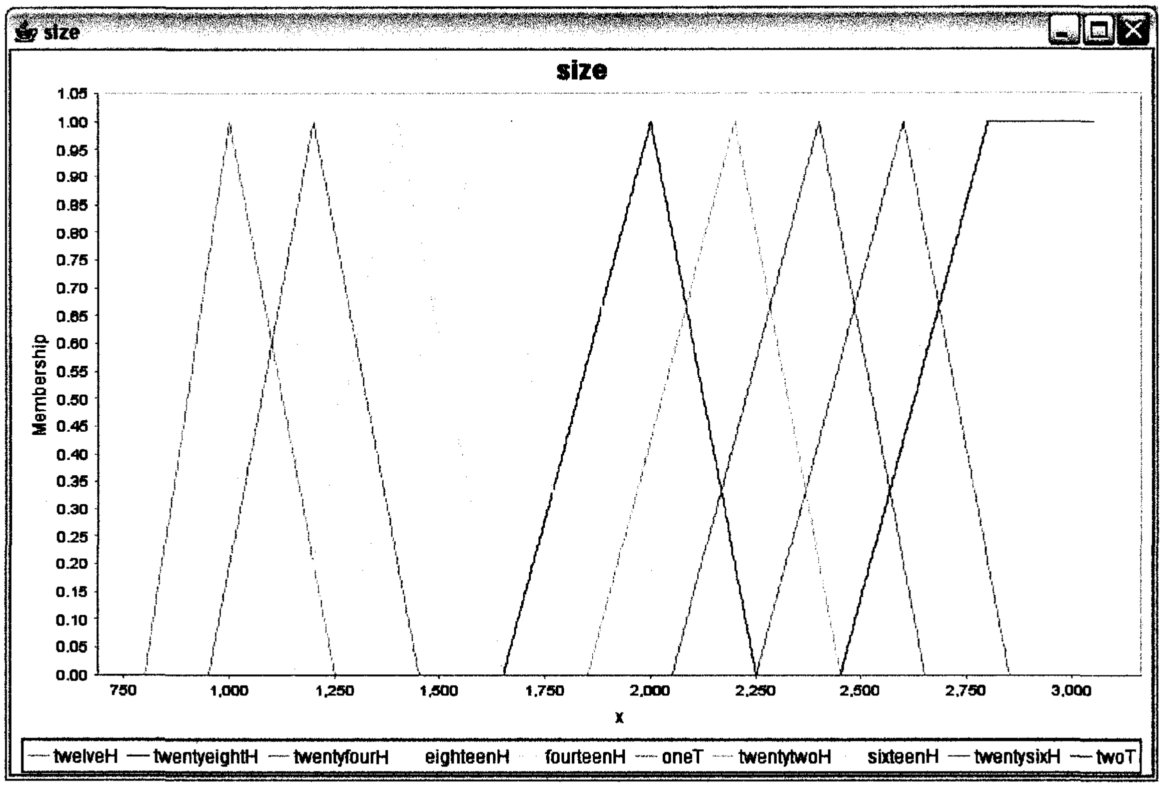


*Figure 9.* Fuzzy Rule Set for Size Variable

The linguistic term names used for the size variable are: oneT, twelveH, fourteenH, sixteenH, eighteenH, twoT, twentytwoH, twentyfourH, twentysixH, and twentyeightH. The T in the linguistic term refers to thousand and the H in the linguistic term refers to hundred. Thus, twelveH equals to twelve hundred and twoT refers to two thousand. The justification for using such names as linguistic term names is that it signifies that the middle defined point has that value. It also indicates the lower and upper bound for that term. For example, one of the middle linguistic terms has been named twoT, which signifies that the middle point for this term is at (2000, 1). Based on the design of this variable's linguistic terms, the lower bound has been classified at being 349 less than the linguistic term value. Hence the lower bound point for linguistic term twoT has been described at (1651, 0).

The upper bound for this variable's linguistic terms has been defined to be 251 more than the linguistic term value. Consequently the upper bound has been described at (600000, 0). Based on all this information, the linguistic terms for the size variable cover the range for the sizes of properties by giving partial membership to any wide range of sizes entered by the user. The next section describes the fuzzy rule set of the lot variable.

*Fuzzy Rule Set for Lot Variable*

The lot variable, shown in Figure 10, has nine linguistic terms, where each linguistic term has been associated to the lot size of the property. As mentioned above for the size variable, since there is such a wide range for a lot size of any property, there was a need to define this variable by nine linguistic terms. Each linguistic name has been uniquely defined according to the lot size range based on the house listings from the database. The linguistic terms were chosen based on the data; the cutoffs have been based on the minimum and maximum lot sizes of the properties listed in the database. This variable also uses the piece-wise linear function as its membership function.

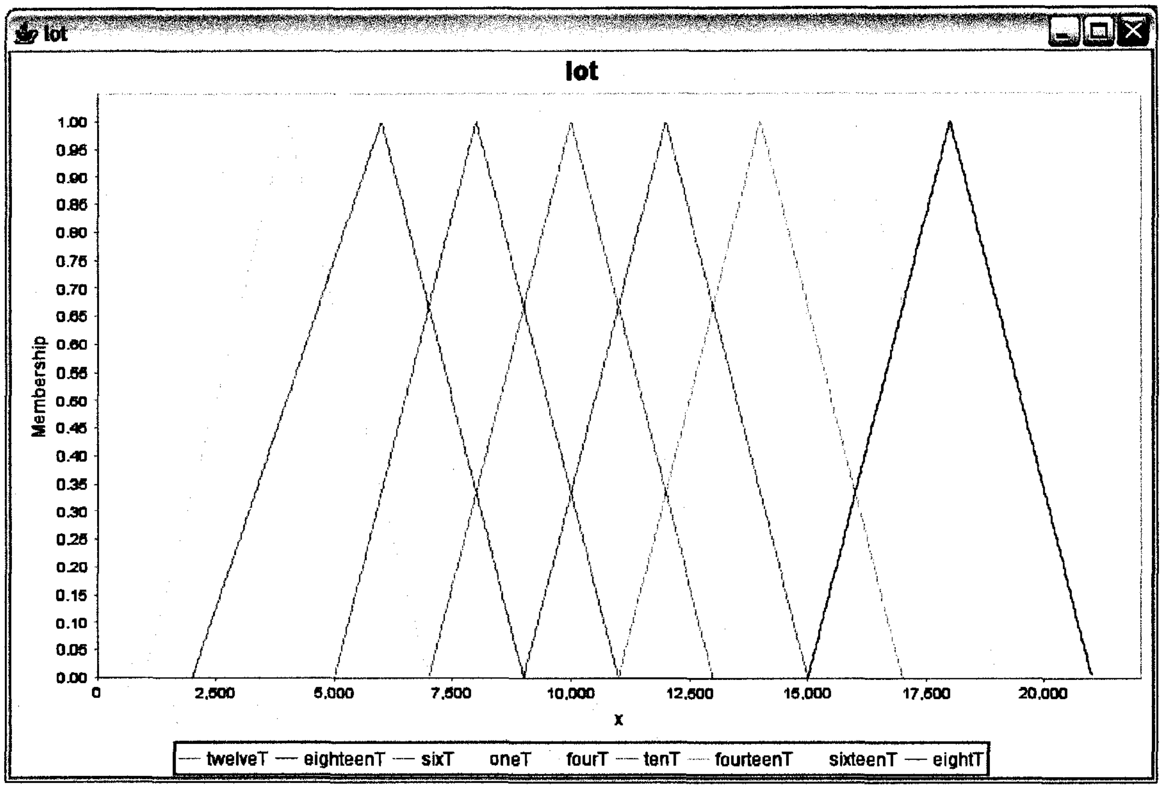


*Figure 10.* Fuzzy Rule Set for Lot Variable

The linguistic term names used for the lot variable are: oneT, fourT, sixT, eightT, tenT, twelveT, fourteenT, sixteenT, and eigthteenT. The T in the linguistic terms refer to thousand, hence, oneT equals to one thousand. The justification for utilizing such names as linguistic term names is that it signifies that the middle defined point has that value. It also denotes the lower and upper bounds for that term. For example, one of the middle linguistic terms has been named eightT, which signifies that the middle point for this term is at (8000, 1). The way this variable's linguistic terms have been defined, the lower bound has been described at 2999 less than the linguistic term value. Hence the lower bound point for linguistic term eightT is identified at (3001, 0). The upper bound for this variable's linguistic terms has been defined to be 3001 more than the linguistic term value. Consequently the upper bound is described at (11001, 0).

The reason for choosing the mentioned lower and upper bound values for each linguistic term was to include those houses whose lot sizes are specified exactly to be that given value by the user. For example, if the user selects houses whose lot size equals to 11,000 square feet, based on the linguistic terms defined, the user defined variable would have partial membership in three linguistic terms: eightT, tenT, and twelveT. Hence this would be returned to the fuzzy intelligence system and the houses which have lot sizes between 8,000 and 12,000 square foot would be queried for accordingly.

The next section describes the FCL file we defined for the jFuzzyLogic software.

## *Fuzzy Control Logic file – house.fcl*

The associated fuzzy control logic (FCL) file that we have defined for the jFuzzyLogic software to use for this intelligent query system is shown below:

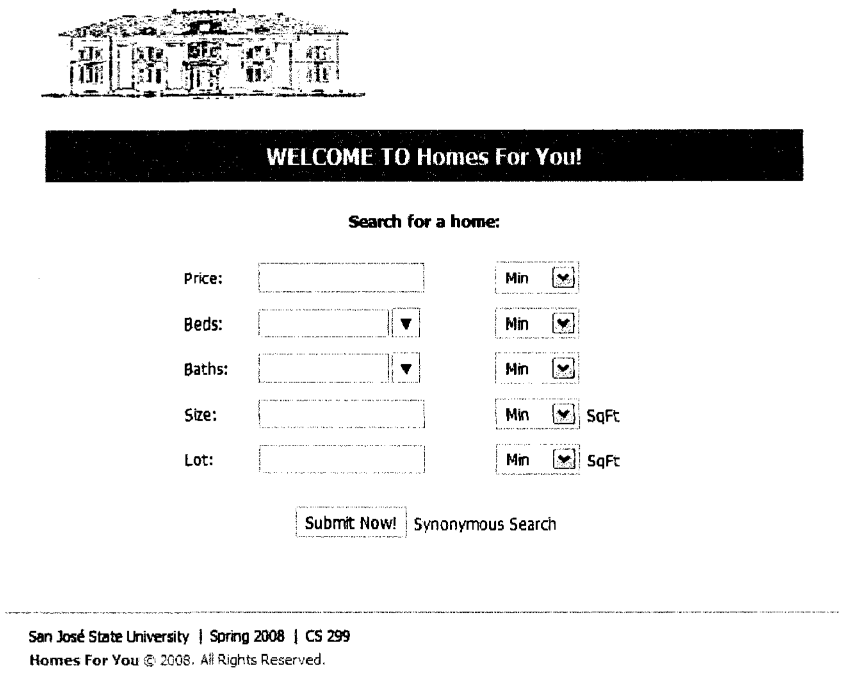
```
FUNCTION_BLOCK house  // Block definition

VAR_INPUT             // Define input variables
    price : REAL;
    beds  : REAL;
    baths : REAL;
    size  : REAL;
    lot   : REAL;
END_VAR

VAR_OUTPUT            // Define output variable
    rating : REAL;
END_VAR

FUZZIFY price         // Fuzzify input variable 'price'
    TERM fourfiftyK     := (350000, 0) (450000, 1) (600000, 0);
    TERM sixfiftyK      := (500000, 0) (650000, 1) (800000, 0);
    TERM eightfiftyK    := (700000, 0) (850000, 1) (1000000,0);
    TERM tenfiftyK      := (900000, 0) (1050000,1) (1200000,0);
    TERM twelvefiftyK   := (1100000,0) (1250000,1) (1400000,0);
    TERM fourteenfiftyK := (1300000,0) (1450000,1) (1600000,0);
    TERM sixteenfiftyK  := (1500000,0) (1650000,1) (1800000,0);
    TERM eighteenfiftyK := (1700000,0) (1850000,1) (2000000,0);
    TERM twentyfiftyK   := (1900000,0) (2050000,1) (2200000,0);
END_FUZZIFY

FUZZIFY beds          // Fuzzify input variable 'beds'
    TERM one   := (0,  0) (1,1) (2.5,0);
    TERM two   := (0.5,0) (2,1) (3.5,0);
    TERM three := (1.5,0) (3,1) (4.5,0);
    TERM four  := (2.5,0) (4,1) (5.5,0);
    TERM five  := (3.5,0) (5,1) (6.5,0);
    TERM six   := (4.5,0) (6,1) (7.5,0);
    TERM seven := (5.5,0) (7,1) (8.5,0);
END_FUZZIFY

FUZZIFY baths         // Fuzzify input variable 'beds'
    TERM one   := (0,  0) (1,1) (2.5,0);
    TERM two   := (0.5,0) (2,1) (3.5,0);
    TERM three := (1.5,0) (3,1) (4.5,0);
    TERM four  := (2.5,0) (4,1) (5.5,0);
    TERM five  := (3.5,0) (5,1) (6.5,0);
    TERM six   := (4.5,0) (6,1) (7.5,0);
    TERM seven := (5.5,0) (7,1) (8.5,0);
END_FUZZIFY

FUZZIFY size          // Fuzzify input variable 'size'
    TERM oneT         := (800,  0) (1000, 1) (1251, 0);
    TERM twelveH      := (951,  0) (1200, 1) (1451, 0);
    TERM fourteenH    := (1151, 0) (1400, 1) (1651, 0);
    TERM sixteenH     := (1251, 0) (1600, 1) (1851, 0);
    TERM eighteenH    := (1451, 0) (1800, 1) (2051, 0);
    TERM twoT         := (1651, 0) (2000, 1) (2251, 0);
    TERM twentytwoH   := (1851, 0) (2200, 1) (2451, 0);
    TERM twentyfourH  := (2051, 0) (2400, 1) (2651, 0);
    TERM twentysixH   := (2251, 0) (2600, 1) (2851, 0);
    TERM twentyeightH := (2451, 0) (2800, 1) (3051, 1);
END_FUZZIFY

FUZZIFY lot              // Fuzzify input variable 'size'
    TERM oneT       := (400,  0) (1000, 1) (2001, 0);
    TERM fourT      := (1001, 0) (4000, 1) (7001, 0);
    TERM sixT       := (2001, 0) (6000, 1) (9001, 0);
    TERM eightT     := (5001, 0) (8000, 1) (11001,0);
    TERM tenT       := (7001, 0) (10000, 1) (13001,0);
    TERM twelveT    := (9001, 0) (12000, 1) (15001,0);
    TERM fourteenT := (11001,0) (14000, 1) (17001,0);
    TERM sixteenT  := (13001,0) (16000, 1) (19001,0);
    TERM eighteenT := (15001,0) (18000, 1) (21001,0);
END_FUZZIFY

DEFUZZIFY rating         // Defzzzify output variable 'rating'
    TERM low         := (0, 0) (1, 1) (2, 0);
    TERM middlelow   := (2, 0) (3, 1) (4, 0);
    TERM middle      := (4, 0) (5, 1) (6, 0);
    TERM middlehigh := (6, 0) (7, 1) (8, 0);
    TERM high        := (8, 0) (9, 1) (10,0);
    ACCU : NSUM;         // Use 'max' accumulation method
    METHOD : COG;        // Use 'Center Of Gravity' defuzzification
                         // method
    DEFAULT := 0;        // Default value is 0 (if no rule
                         // activates defuzzifier)
END_DEFUZZIFY

RULEBLOCK No1
    AND : MIN;           // Use 'min' for 'and' (also implicit use
                         // 'max' for 'or' to fulfill DeMorgan's
                         // Law)
    ACT : MIN;           // Use 'min' activation method

// There are no rules in this rule block because none are needed
// for this project. I am only interested in membership values
// and their associated linguistic terms

END_RULEBLOCK

END_FUNCTION_BLOCK
```

The jFuzzyLogic software package that we utilize for this project provides plotted graph outputs of the FuzzyRuleSet as defined in the FCL file. This is useful for debugging purposes since the software shows the graph output. This allows the user to see if all the linguistic terms in each variable have been defined correctly as compared to each other. This feature is also useful for viewing the FuzzyRuleSet as an image versus the code from the FCL file.

The next section explains how we employed fuzzy logic, from the jFuzzyLogic software, in the standard field search in the *Homes For You* real estate website.

### *Standard Field Search (Fuzzy)*

The user interface design for the standard field search is shown in Figure 11. As displayed, five standard search field selections have been defined: price of house, number of bedrooms and bathrooms in a house, size and lot size of the house (measured in square feet). The user is allowed to type in any appropriate value for each of the five standard search fields.


WELCOME TO Homes For You!

Search for a home:

| | | |
|---|---|---|
| Price: | | Min |
| Beds: | | Min |
| Baths: | | Min |
| Size: | | Min SqFt |
| Lot: | | Min SqFt |

San José State University | Spring 2008 | CS 299
Homes For You © 2008. All Rights Reserved.

*Figure 11.* UI of Standard Field Selection (Fuzzy)

For the beds and baths field selections, a user has the option to click on the drop down list to pick a number or can type in a value in the text field. The reason for having a text box is to help facilitate a range input. The drop down list does not contain any range hence the need for an input text box. The justification for including a drop down list for the beds and baths field selections is to display for the user the range of bedrooms and bathrooms for the entire house listing data set. Thus a user can either select an input value from the drop down list or enter in the input value manually. The next section describes the different constraints defined for the standard search fields.

*Constraints for Standard Fields*

Next to each of the five standard search field input boxes, a user can also modify the input value by selecting a constraint: minimum, maximum, exact, or range. To explain this further, let us look at a very simple example. For instance, suppose that a user wants a four-bedroom house. There are different scenarios for each of the four different constraints as explained in the next four sub-sections.

*Minimum constraint.* If the user picks minimum as the constraint for the input value, this fuzzy intelligence system is defined to utilize the Java classes from jFuzzyLogic to obtain the membership values of the associated linguistic terms for the bedrooms variable. From the example in the previous section, the input value for the bedrooms variable will be four and based on the FuzzyRuleSet discussed above, the input value of four has partial membership in linguistic terms three and five. Since the user has specified the constraint to be minimum and four as the input value, linguistic term three has partial membership. Therefore, the database is also queried for three-bedroom houses.

The result is then ranked according to the membership values. In this case, all of the house properties with four-, five-, six-, and seven- bedrooms are listed first and at the bottom of the results page, three-bedroom houses are displayed.

The next section describes the second constraint option available for a user to select.

*Maximum constraint.* If the user picked maximum instead of minimum as the constraint for the input value, this fuzzy intelligence system is defined to utilize the Java classes from the jFuzzyLogic software package to obtain membership values of the associated linguistic terms for the bedrooms variable. As from the same example before, the input value for the bedrooms variable is still four. The FuzzyRuleSet is still the same and based on Figure 7, the input value four has partial membership in two linguistic fields, namely three and five. In this case since the constraint it maximum, the database will also be queried for five-bedroom houses. The result is once again ranked based on the membership values. The user will see one-, two-, three-, four- and then five-bedroom house listings. Five-bedroom houses will be at the end of the result list, since it had the least membership value amongst all other number of bedrooms.

The next section describes the third constraint option available for the user to select.

*Range constraint.* If the user picked range instead of maximum as the constraint for the input value, this defined fuzzy intelligence system calls the Java class from the jFuzzyLogic package to obtain the membership values of the associated linguistic terms for the bedrooms variable. The example now is slightly changed; the user queries for three-five bedrooms. Thus, the input value for the bedrooms variable is now 3-5, which means three, four, and five are all considered to be the input values.

The same FuzzyRuleSet from Figure 7 is used to acquire membership values. The input variable three has partial membership in linguistic terms two and four. The input variable four has partial membership in linguistic terms three and five. Lastly, the input variable five has partial membership in linguistic terms four and six. Thus, the database would be queried for three-, four-, five-bedroom houses as well as two- and six- bedroom houses. This result will also be ranked, the number of bedrooms with the highest membership values will be listed towards the top of the search result and the bedrooms with the lowest membership values will be listed at the bottom of the search result.

The next section explains the final constraint option available for the user to select.

*Exact constraint.* If the user picks the last constraint, exact, for the input value, then just as before, this defined fuzzy intelligence system calls the Java class from the jFuzzyLogic software to obtain the membership values of the associated linguistic terms. As in the example used in the minimum and maximum constraint sections, the input value for the bedrooms variable is three. Using the FuzzyRuleSet from Figure 7, we obtain the membership values of the linguistic terms. The input variable three has partial membership in linguistic terms two and four. Therefore, the database would be queried for two-, three-, and four-bedroom houses. This result would be ranked according to the membership values, with three-bedroom houses listed at the top of the search results and the two- and four-bedroom houses listed at the bottom of the list.

In a traditional query, as done in the real estate websites mentioned in the first section, any input value for the houses variable would take a crisp number with no partial membership in other numbers. These real estate websites use crisp logic for executing a search query in the database. Thus, the input variable is used to define the crisp set and to find houses listings that are exactly in the set. Therefore, there is no partial membership given to those numbers that are slightly close to the set; either the number is in the set or it is not. For example, a three bedroom house with an exact constraint would only return three-bedroom houses. No partial membership would be taken into consideration in traditional real estate websites and thus a user will not be able achieve as credible and precise results as opposed to the intelligent query system defined in the project.

The next section discusses the second feature of this project, a free keyword search for the user.

## *Synonymous Search*

Another important aspect of this project includes a synonymous search of the house listings. What if a user wants to search for houses but not using the standard field selections? The synonymous search allows the user to type in a keyword and search for real estate properties. Basically, a user is allowed to do a free keyword search for properties. For example, a user can search for *big backyard* houses, and the database would also be queried for *large backyard* houses since large is a synonym for big.

The user interface design of the synonymous search feature is shown in Figure 12. As mentioned, the user types in keyword(s) in the search bar, the keywords are passed into the website, and the synonyms are retrieved. The retrieved synonyms are then used to query the database for matching results. The website utilized to find synonyms of the keywords entered by the user is titled Encarta Thesaurus by MSN. Depending on the keyword, this website displays many synonyms but only the first two sets of synonyms are obtained to use by the database search query. Appendix C contains the PHP script used to obtain the synonyms from the Encarta website. The results are displayed in ranking order, with the supplied keyword input search result at the top and the synonymous search results following the original result.

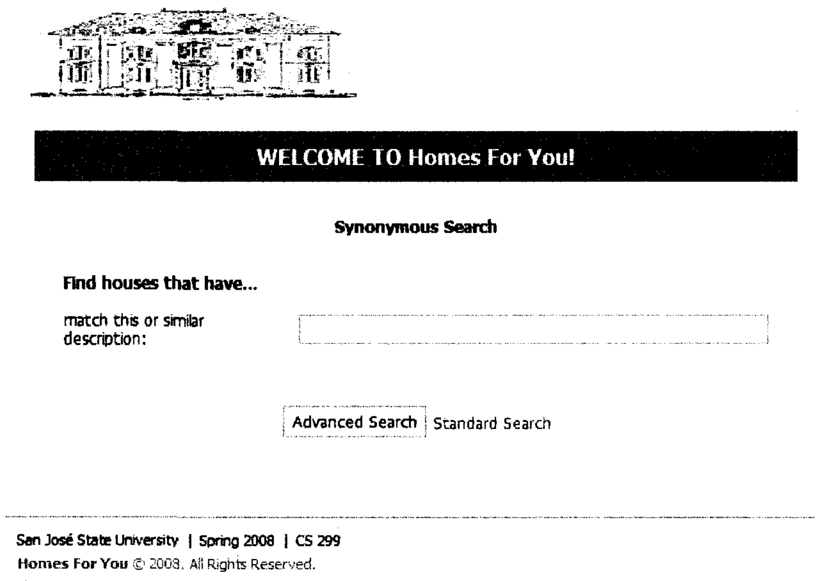

WELCOME TO Homes For You!
Synonymous Search
Find houses that have...
match this or similar description:
Advanced Search
Standard Search
San José State University | Spring 2008 | CS 299
Homes For You © 2008. All Rights Reserved.

*Figure* 12. UI of Synonymous Search

A user can click on a particular house listing from the synonymous search result and see the highlighted keyword(s) or synonymous keyword(s) within that listing to reinforce the result.

In the next section, we discuss the results obtained from this project. The results are shown using several examples.

## RESULTS

Different case examples for the standard field and synonymous search will be presented in this section. This will help demonstrate the capabilities of this intelligent query system. The first five examples will show different cases of utilizing the standard field search feature and the last five examples will show the usage of the synonymous search feature using different keywords as inputs.

### *Example 1 – Standard Field Search*

In this first example, we have searched for a home using just the price standard field selection and the range constraint. The input selected for the price range is 550000 – 750000. Using the fuzzy rule set for the price variable, as shown in Figure 6, we find that this input has partial membership in two linguistic terms, namely fourfiftyK and sixfiftyK. Therefore, the database will be queried for house listing with a price that is between 450000 and 750000. This search result returns twenty-two house listings. If the actual input, 550000 – 750000, was utilized to query the database, the result set would be slightly smaller, with eighteen house listings. The next section describes another example of using the standard field search.

## *Example 2 – Standard Field Search*

For this example, we have searched for a home using only the beds variable and the maximum constraint. The input for the beds variable is two. Based on the fuzzy rule set for the bed variable, as shown in Figure 7, we find that this input value has partial membership in two linguistic terms, namely one and three. Since the maximum constraint has been used, we also include three-bedroom houses in the search query to the database. This query yields eight matching houses. It can be noticed that the search returns houses with three-bedrooms only. This means that there are no houses with only two-bedrooms. So the fuzzy intelligence search returned some houses that would otherwise not be included in the result if a traditional real estate website utilized this same data set. In the next section we illustrate another example using the standard field search.

## *Example 3 – Standard Field Search*

In this next example for the standard field search we have used the baths variable and the minimum constraint. The input for the baths variable is three. Based on the fuzzy rule set for the baths variable, as shown in Figure 8, we find that the input three has partial membership in two linguistic terms, namely two and four. Since the minimum constraint has been utilized, we also include two bathroom houses in the query to the database. This search result returns fifty-six matching houses. There are also two houses with two-bathrooms listed at the end of the list since the input value *three* had partial membership in linguistic term two. In the next section we describe another example using the standard field search.

### *Example 4 – Standard Field Search*

In this example for the standard field search, the size variable with the exact constraint has been utilized to query for houses. The input value for the size variable has been defined to be 1050 square feet. Based on the fuzzy rule set, from Figure 9, for the size variable, we find that the input of 1050 has partial membership in two linguistic terms, namely oneT (one thousand) and twelveH (twelve hundred). Since the exact constraint has been utilized, the database will be queried for houses whose sizes are between one thousand and twelve hundred square feet. This search result returns eleven matching houses. In a non intelligent query system for a real estate website, if the user searches for houses with exactly 1050 square feet in size, it will not result in any matching houses since no house exists with that exact size in square feet from the provided data set. In the next section we discuss the final example using the standard field search.

*Example 5 – Standard Field Search*

For this last standard field search example, four of the standard fields have been utilized to do a search. The price variable has been defined to have range as its constraint with 550000-750000 as its value. The bedrooms variable has been defined to have maximum as its constraint three as its value. The bathrooms variable has been defined to have minimum as its constraint and two as its value. Lastly, the size variable for this search has been defined to have the exact constraint and 1050 square feet as its value. Based on the fuzzy rule sets for the price, bedrooms, bathrooms, and size variables, each variable has some linguistic terms that have some partial membership values. Taking all the partial membership values of the linguistic terms into consideration, this search yields eight matching houses, including a *$430,000, 4-bedroom, 3-bathroom, 1200 square foot* house. In the next section we describe an example using the synonymous search feature.

*Example 6 – Synonymous Search*

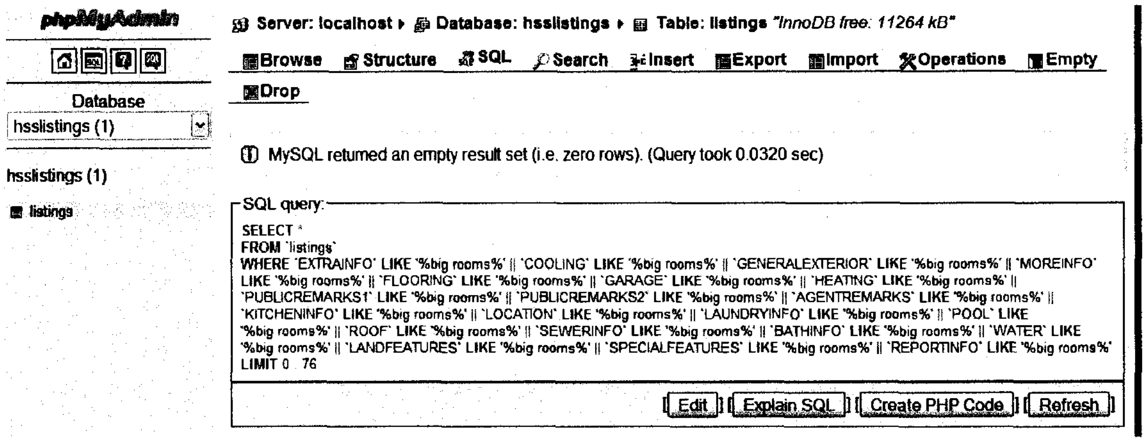
For this synonymous search example, we will input *big rooms* as the keyword search phrase. To show that *big rooms* do not exist in any of the fields in the database, a search query for this keyword phrase will be sent to the database. The MySQL query will be of the following form:

```
SELECT *  FROM `listings` WHERE `EXTRAINFO` LIKE '%%' ||
`COOLING` LIKE '%%' ||  `GENERALEXTERIOR` LIKE '%%' ||
`MOREINFO` LIKE '%%' ||  `FLOORING` LIKE '%%' ||  `GARAGE` LIKE
'%%' ||  `HEATING` LIKE '%%' ||  `PUBLICREMARKS1` LIKE '%%' ||
`PUBLICREMARKS2` LIKE '%%' ||  `AGENTREMARKS` LIKE '%%' ||
`KITCHENINFO` LIKE '%%' ||  `LOCATION` LIKE '%%' ||
`LAUNDRYINFO` LIKE '%%' ||  `POOL` LIKE '%%' ||  `ROOF` LIKE '%%'
||  `SEWERINFO` LIKE '%%' ||  `BATHINFO` LIKE '%%' ||  `WATER`
LIKE '%%' ||  `LANDFEATURES` LIKE '%%' ||  `SPECIALFEATURES` LIKE
'%%' ||  `REPORTINFO` LIKE '%%'
```

The two percent marks match any number of characters, including zero characters. By incorporating the keyword phrase *big rooms* and the LIKE operator, the search query above will do pattern matching of the specified keyword. To illustrate this case, here is a simple query:

```
SELECT * FROM `listings` WHERE `PUBLICREMARKS1` LIKE '%big
rooms%' || `PUBLICREMARKS2` LIKE '%big rooms%' || `AGENTREMARKS`
LIKE '%big rooms%'
```

This query translates to: find all house listings from the listings table where the public remarks 1, public remarks 2, and agent remarks fields contain the key phrase *big rooms*. Executing this query returns an empty result set indicating that the key phrase *big rooms* does not exist in any of the fields of the house listings. This is shown in Figure 13.



MySQL returned an empty result set (i.e. zero rows). (Query took 0.0320 sec)

SQL query:

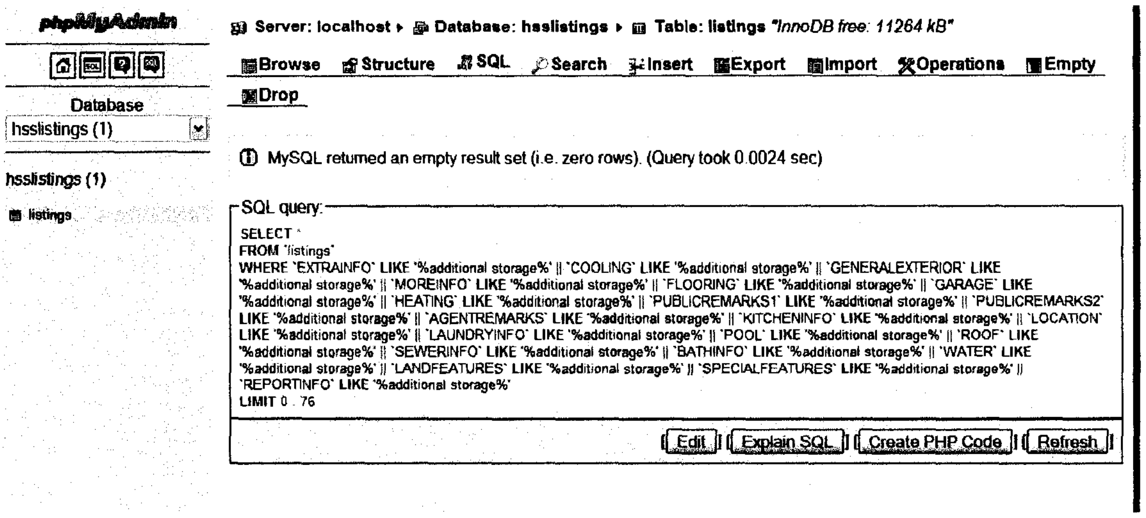
```
SELECT *
FROM `listings`
WHERE `EXTRAINFO` LIKE '%big rooms%' || `COOLING` LIKE '%big rooms%' || `GENERALEXTERIOR` LIKE '%big rooms%' || `MOREINFO`
LIKE '%big rooms%' || `FLOORING` LIKE '%big rooms%' || `GARAGE` LIKE '%big rooms%' || `HEATING` LIKE '%big rooms%' ||
`PUBLICREMARKS1` LIKE '%big rooms%' || `PUBLICREMARKS2` LIKE '%big rooms%' || `AGENTREMARKS` LIKE '%big rooms%' ||
`KITCHENINFO` LIKE '%big rooms%' || `LOCATION` LIKE '%big rooms%' || `LAUNDRYINFO` LIKE '%big rooms%' || `POOL` LIKE
'%big rooms%' || `ROOF` LIKE '%big rooms%' || `SEWERINFO` LIKE '%big rooms%' || `BATHINFO` LIKE '%big rooms%' || `WATER` LIKE
'%big rooms%' || `LANDFEATURES` LIKE '%big rooms%' || `SPECIALFEATURES` LIKE '%big rooms%' || `REPORTINFO` LIKE '%big rooms%'
LIMIT 0 , 76
```

*Figure 13.* Query result for *big rooms* keyword

The synonymous keywords for *big rooms* are the following: large rooms, giant rooms, immense rooms, vast rooms, great rooms, gigantic rooms, great big rooms, huge rooms, enormous rooms, whopping rooms, full-size rooms, life-size rooms, spacious rooms, capacious rooms, roomy rooms, large rooms, and deep rooms. The database is queried for all the synonymous keywords and the search returns two matching houses. Both of the houses listed in the search results contain the key phrase *spacious rooms*. In this example, the synonymous search proved successful. In the next section we discuss a second example using the synonymous search feature.

### *Example 7 – Synonymous Search*

In another example of the synonymous search, we are using the keyword phrase *additional storage*. The synonymous keywords for this phrase include extra storage, added storage, supplementary storage, other storage, further storage, bonus storage, surplus storage, and superfluous storage. This key phrase and its synonymous key phrases are searched in the database and this search results in sixteen matching houses. The query mentioned in Section 5.6 could be used to test if *additional storage* is listed the database. As shown in Figure 14, when this query is executed, it yields an empty result set. This indicates the synonyms for *additional storage* were used to obtain these results of sixteen matching houses.


phpMyAdmin
Database
hsslistings (1)
hsslistings (1)
listings
Server: localhost ▸ Database: hsslistings ▸ Table: listings "InnoDB free: 11264 kB"
Browse Structure SQL Search Insert Export Import Operations Empty
Drop
Edit
Explain SQL
Create PHP Code
Refresh

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0024 sec)

SQL query:

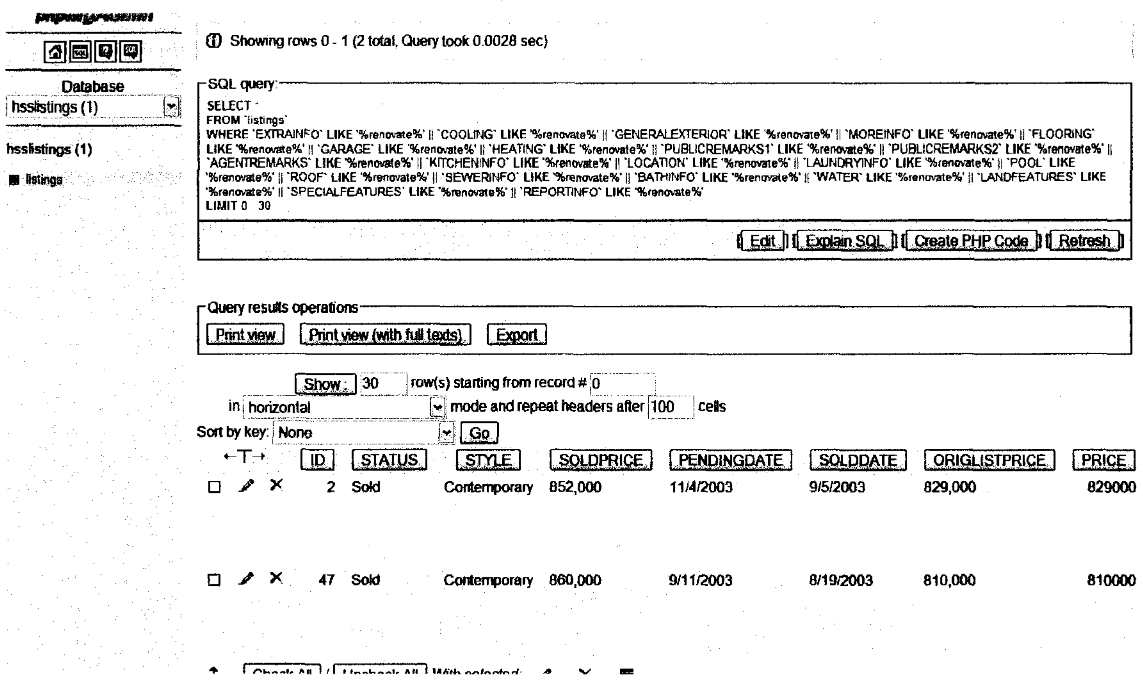
```
SELECT *
FROM `listings`
WHERE `EXTRAINFO` LIKE '%additional storage%' || `COOLING` LIKE '%additional storage%' || `GENERALEXTERIOR` LIKE
'%additional storage%' || `MOREINFO` LIKE '%additional storage%' || `FLOORING` LIKE '%additional storage%' || `GARAGE` LIKE
'%additional storage%' || `HEATING` LIKE '%additional storage%' || `PUBLICREMARKS1` LIKE '%additional storage%' || `PUBLICREMARKS2`
LIKE '%additional storage%' || `AGENTREMARKS` LIKE '%additional storage%' || `KITCHENINFO` LIKE '%additional storage%' || `LOCATION`
LIKE '%additional storage%' || `LAUNDRYINFO` LIKE '%additional storage%' || `POOL` LIKE '%additional storage%' || `ROOF` LIKE
'%additional storage%' || `SEWERINFO` LIKE '%additional storage%' || `BATHINFO` LIKE '%additional storage%' || `WATER` LIKE
'%additional storage%' || `LANDFEATURES` LIKE '%additional storage%' || `SPECIALFEATURES` LIKE '%additional storage%' ||
`REPORTINFO` LIKE '%additional storage%'
LIMIT 0 , 76
```

*Figure 14.* Query result for *additional storage* keyword

In the next section we describe another example using the synonymous search feature.

## *Example 8 – Synonymous Search*

In this example, a single keyword synonymous search will be executed. The keyword is *renovate* and its synonyms are: renew, recondition, modernize, refurbish, repair, restore, mend, fix up, revamp, remodel, redecorate, and do up. The synonymous search for *renovate* yields fifteen matching houses. Just to make sure that only the synonyms of this keyword are responsible for such a large result set, the query mentioned in Section 5.6 is used to match for the keyword *renovate*. The MySQL search query returns two matching rows of data, as shown in Figure 15. This signifies that *renovate* occurs in two of the fifteen synonymous search matches indicating that thirteen of the fifteen matches occurred based on the synonym of *renovate*.



Showing rows 0 - 1 (2 total, Query took 0.0028 sec)

SQL query:

```
SELECT *
FROM `listings`
WHERE `EXTRAINFO` LIKE '%renovate%' || `COOLING` LIKE '%renovate%' || `GENERALEXTERIOR` LIKE '%renovate%' || `MOREINFO` LIKE '%renovate%' || `FLOORING`
LIKE '%renovate%' || `GARAGE` LIKE '%renovate%' || `HEATING` LIKE '%renovate%' || `PUBLICREMARKS1` LIKE '%renovate%' || `PUBLICREMARKS2` LIKE '%renovate%' ||
`AGENTREMARKS` LIKE '%renovate%' || `KITCHENINFO` LIKE '%renovate%' || `LOCATION` LIKE '%renovate%' || `LAUNDRYINFO` LIKE '%renovate%' || `POOL` LIKE
'%renovate%' || `ROOF` LIKE '%renovate%' || `SEWERINFO` LIKE '%renovate%' || `BATHINFO` LIKE '%renovate%' || `WATER` LIKE '%renovate%' || `LANDFEATURES` LIKE
'%renovate%' || `SPECIALFEATURES` LIKE '%renovate%' || `REPORTINFO` LIKE '%renovate%'
LIMIT 0 30
```

Show: 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells
Sort by key: None

| ID | STATUS | STYLE | SOLDPRICE | PENDINGDATE | SOLDDATE | ORIGLISTPRICE | PRICE |
|---|---|---|---|---|---|---|---|
| 2 | Sold | Contemporary | 852,000 | 11/4/2003 | 9/5/2003 | 829,000 | 829000 |
| 47 | Sold | Contemporary | 860,000 | 9/11/2003 | 8/19/2003 | 810,000 | 810000 |

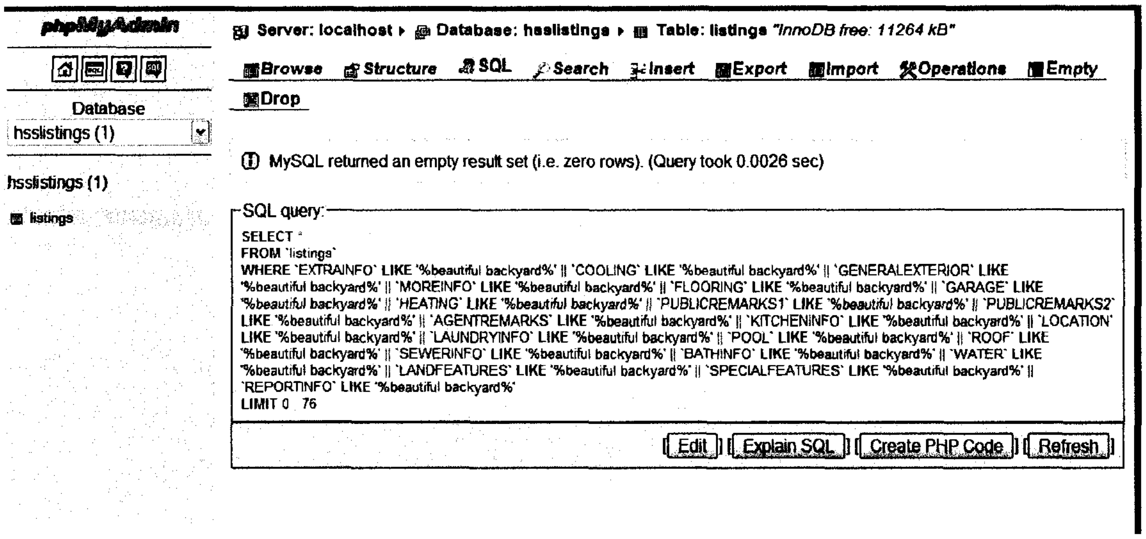
*Figure 15.* Query result for *renovate* keyword

In the next section we illustrate a fourth example using the synonymous search feature.

*Example 9 – Synonymous Search*

In this example of the synonymous search, the key phrase chosen has been *beautiful backyard*. The synonyms of this key phrase include: lovely backyard, attractive backyard, good-looking backyard, gorgeous backyard, stunning backyard, striking backyard, fine-looking backyard, handsome backyard, picturesque backyard, scenic backyard, delightful backyard, charming backyard, wonderful backyard, exquisite backyard, pleasing backyard, superb backyard, and magnificent backyard. This synonymous search yields two matching houses.

The search query from Section 5.6, used to test for the initial key phrase in the database, yields an empty set as shown in Figure 16. Upon clicking on the two listings of houses, it is viewed that the first listing contains the synonymous phrase gorgeous backyard while the second listing contains the synonymous phrase lovely backyard. This example proves that this synonymous search feature is a positive advancement for helping users in finding real estate properties.


phpMyAdmin
Server: localhost ▸ Database: hsslistings ▸ Table: listings "InnoDB free: 11264 kB"
Browse Structure SQL Search Insert Export Import Operations Empty
Drop
Database
hsslistings (1)
hsslistings (1)
listings
Edit
Explain SQL
Create PHP Code
Refresh

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0026 sec)

SQL query:

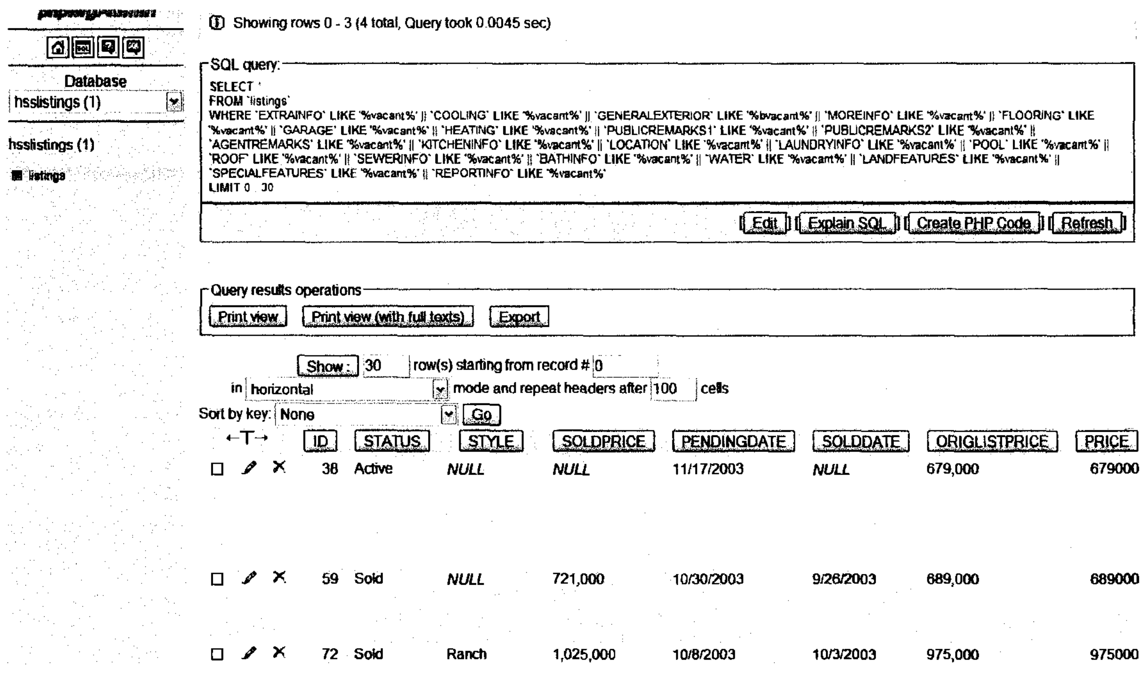
```
SELECT *
FROM `listings`
WHERE `EXTRAINFO` LIKE '%beautiful backyard%' || `COOLING` LIKE '%beautiful backyard%' || `GENERALEXTERIOR` LIKE
'%beautiful backyard%' || `MOREINFO` LIKE '%beautiful backyard%' || `FLOORING` LIKE '%beautiful backyard%' || `GARAGE` LIKE
'%beautiful backyard%' || `HEATING` LIKE '%beautiful backyard%' || `PUBLICREMARKS1` LIKE '%beautiful backyard%' || `PUBLICREMARKS2`
LIKE '%beautiful backyard%' || `AGENTREMARKS` LIKE '%beautiful backyard%' || `KITCHENINFO` LIKE '%beautiful backyard%' || `LOCATION`
LIKE '%beautiful backyard%' || `LAUNDRYINFO` LIKE '%beautiful backyard%' || `POOL` LIKE '%beautiful backyard%' || `ROOF` LIKE
'%beautiful backyard%' || `SEWERINFO` LIKE '%beautiful backyard%' || `BATHINFO` LIKE '%beautiful backyard%' || `WATER` LIKE
'%beautiful backyard%' || `LANDFEATURES` LIKE '%beautiful backyard%' || `SPECIALFEATURES` LIKE '%beautiful backyard%' ||
`REPORTINFO` LIKE '%beautiful backyard%'
LIMIT 0 , 76
```

*Figure 16.* Query result for *beautiful backyard* keyword

In the next section we describe the last example using the synonymous search feature.

*Example 10 – Synonymous Search*

The last synonymous search example uses *vacant* as its keyword. The synonyms associated with *vacant* are empty, available, unoccupied, not in use, unfilled, untaken, free, clear, blank, expressionless, indifferent, vacuous, uncomprehending, and inane. This synonymous search generates fifty-six matching houses. The MySQL query from Section 5.6, which checks for the keyword input in the database, is executed using the keyword and results in four matching rows, as shown in Figure 17. This signifies that four of the fifty-six search result listings contain the word vacant and fifty-two of the remaining search result house listings contain a synonym for *vacant*.



Showing rows 0 - 3 (4 total, Query took 0.0045 sec)

SQL query:

```
SELECT *
FROM `listings`
WHERE `EXTRAINFO` LIKE '%vacant%' || `COOLING` LIKE '%vacant%' || `GENERALEXTERIOR` LIKE '%bvacant%' || `MOREINFO` LIKE '%vacant%' || `FLOORING` LIKE
'%vacant%' || `GARAGE` LIKE '%vacant%' || `HEATING` LIKE '%vacant%' || `PUBLICREMARKS1` LIKE '%vacant%' || `PUBLICREMARKS2` LIKE '%vacant%' ||
`AGENTREMARKS` LIKE '%vacant%' || `KITCHENINFO` LIKE '%vacant%' || `LOCATION` LIKE '%vacant%' || `LAUNDRYINFO` LIKE '%vacant%' || `POOL` LIKE '%vacant%' ||
`ROOF` LIKE '%vacant%' || `SEWERINFO` LIKE '%vacant%' || `BATHINFO` LIKE '%vacant%' || `WATER` LIKE '%vacant%' || `LANDFEATURES` LIKE '%vacant%' ||
`SPECIALFEATURES` LIKE '%vacant%' || `REPORTINFO` LIKE '%vacant%'
LIMIT 0 , 30
```

Show: 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells
Sort by key: None

| ID | STATUS | STYLE | SOLDPRICE | PENDINGDATE | SOLDDATE | ORIGLISTPRICE | PRICE |
|---|---|---|---|---|---|---|---|
| 38 | Active | *NULL* | *NULL* | 11/17/2003 | *NULL* | 679,000 | 679000 |
| 59 | Sold | *NULL* | 721,000 | 10/30/2003 | 9/26/2003 | 689,000 | 689000 |
| 72 | Sold | Ranch | 1,025,000 | 10/8/2003 | 10/3/2003 | 975,000 | 975000 |

*Figure 17.* Query result for *vacant* keyword

In the next section, we analyze the results; namely the ten examples of the standard search and synonymous search queries.

## *Analysis of the Results*

Based on the ten example case studies for this intelligent query system, this new system proves to be a beneficial piece of software that allows users the ability to search for real estate property and obtain meaningful and credible results. By analyzing the data set of seventy six house listings, the various different cases were described and the results were shown to be uniquely acceptable based on the user defined query via the standard field and synonymous search.

The five standard field search query examples compared the results of the fuzzy logic query system with the standard MySQL query using the described input data. As shown, the fuzzy logic system outputted a larger set of house listings data as compared to a traditional query search. The five synonymous search query examples showed the same results; the fuzzy logic query system produced a larger set of house listings data as compared to the traditional query search. The bigger result set from our newly designed fuzzy logic system, *Homes For You,* showed us that the user can receive more house listings to view and potentially pick out a real estate property from the result set.

Based on all this information, our intelligent query system that we have implemented for the real estate website is very beneficial at retrieving meaningful search results. The users are able to view house listings that were close to their precise search; results of the house listings that the user may not have otherwise originally queried for.

## FUTURE WORK

One enhancement that could be made to this intelligent query system is the usage of all the possible synonyms of a given keyword. The drawback to utilizing only a few of the synonyms is that the scope for a synonymous search is limited since not all of the possible synonyms of a keyword are included in the query. The reason for using only a subset of the synonyms is so that a result could be quickly obtained. Linking to the synonym website and parsing for the synonyms takes some time and then using those synonyms to query the database also involves some time usage. So, to reduce the time for processing all the information, only a subset of the synonyms was used for this project.

Another important aspect is to test this intelligent query system extensively with more house listings data. More data would allow this intelligent query system to be fine tuned accordingly and would help to make sure that the membership functions of the standard variables are defined correctly. Along with testing with more data, this system needs to be analyzed for usability with a very large pool of subjects. The user interface design and logic behind this intelligent query system could then be extensively evaluated through the usability testing process. This would help make the *Homes For You* website a simple and easy to use/ navigate site.

# CONCLUSION

This project has attempted a new approach in retrieving search results by using fuzzy logic. The intelligent query in real estate web site, that we have created, is capable of taking in the standard field selection search data and generating results that are more meaningful. This system also incorporates a search bar for users to input more information about a particular house they want. For example, apart from choosing *5-bedroom, 4-bath, 3-car garage* house, the user can also input into the search bar, not *near* highway and *close* to hospital. So the results of this search will provide for even more meaningful information about houses.

The ten standard field search and synonymous search query examples further reinforce the power of utilizing fuzzy logic in the real estate website domain. The ten cases discussed in Section 5 provide great examples of how well this intelligent query system can return search results. With the simple user interface design, the users can easily search for houses. Based on all this information, the implemented system, therefore, increases the ability of the users to query for real estate property with more accuracy.

Some of the challenging and innovative aspects of this project include implementing an effective fuzzy logic real estate website that provides reliable search results: integrating fuzzy logic using jFuzzyLogic, PHP scripts, and Java files; applying fuzzy logic to standard field selections; parsing of online facts data to incorporate into real estate web site; and performing a synonymous search based on several fields from the database. With the use of all this, this new system, therefore, increases the ability of users to query for real estate property with greater precision.

This intelligent query system has proved to be successful in the real estate domain. We can exploit this intelligent query system further by applying it to other domains or applications. This would help obtain just as accurate and credible results, thus making this intelligent query system powerful.

# REFERENCES

Buche, P., Dervin, C., Haemmerle, O., & Thomopoulos, R. (2005). Fuzzy querying of incomplete, imprecise, and heterogeneously structured data in the relational model using ontologies and rules. *IEEE Transactions on Fuzzy Systems, 13*(3), 373–383.

Chang, S., & Ke, J. (1978). Database skeleton and its application to fuzzy query translation. *IEEE Transactions on Software Engineering, 4*(1), 31–44.

Chen, S., & Jong, W. (1997). Fuzzy query translation for relational database systems. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics, 27*(4), 714–721.

Chen, S., & Lin, Y. (2002). A new method for fuzzy query processing in relational database systems. *Cybernetics & Systems, 33*(5), 444–482.

Choi, D. (2003). Enhancing the power of web search engines by means of fuzzy query. *Decision Support Systems, 35*(1), 31–44.

Intan, R., & Mukaidono, M. (2000). Fuzzy functional dependency and its application to approximate data querying. *Database Engineering and Applications Symposium,* 47–54.

Kacprzyk, J. (1995). Fuzzy logic in DBMSs and querying. *Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems,* (20–23), 106–109.

Kacprzyk, J., & Zadrozny, S. (1994). Fuzzy querying for microsoft access. *IEEE World Congress on Computational Intelligence, Proceedings of the Third IEEE Conference on Fuzzy Systems, 1*(26–29), 167–171.

Kacprzyk, J., & Zadrozny, S. (1996). FQUERY for access: towards human consistent querying user interface. *Symposium on Applied Computing, Proceedings of the 1996 ACM symposium on Applied Computing,* 532–536.

Kacprzyk, J., & Zadrozny, S. (2001). Computing with words in intelligent database querying: standalone and internet-based applications. *Information Sciences, 134*(1–4), 71–109.

Kacprzyk, J., Owsinski, J.W., & Zadrozny, S. (2001). Clusterwise data mining within a fuzzy querying interface. *The 10th IEEE International Conference on Fuzzy Systems, 3,* 1239–1242.

MLSListings. (2006). MLSListings. Retrieved September 27, 2007, from http://mlslistings.com/

MSN Encarta. (2008). Thesaurus. Retrieved April 7, 2008, from http://encarta.msn.com/thesaurus /thesaurus.html

NikRavesh, M. (2002). Fuzzy conceptual-based search engine using conceptual semantic indexing. *Fuzzy Information Processing Society,* 146–151.

Penzo, W. (2005). Rewriting rules to permeate complex similarity and fuzzy queries within a relational database system. *IEEE Transactions on Knowledge and Data Engineering, 17*(2), 255–270.

Prudential Realty. (2008). Prudential Realty. Retrieved September 27, 2007, from http://www.prurealty.com/

SourceForge.net. (2006). jFuzzyLogic. Retrieved November 24, 2007, from http://sourceforge.net/projects/jfuzzylogic

SourceForge.net. (2008). PHP/Java Bridge. Retrieved February 4, 2008, from http://sourceforge.net/project/php-java-bridge

U.S. Census Bureau. (2000). American FactFinder. Retrieved January 22, 2008, from http://factfinder.census.gov/

WampServer. (2008). Apache, PHP, MySQL on Windows. Retrieved January 29, 2008, from http://en.wampserver.com/

Zillow Real Estate. (2006). Real Estate Valuations, Homes for Sale, Free Real Estate Information. Retrieved September 27, 2007, from http://www.zillow.com/

ZipRealty Real Estate. (1999). Homes for sale and local real estate agents. Retrieved September 27, 2007, from http://www.ziprealty.com/

APPENDIX A: XML to Database field names conversion

| XML Field Name | Sample Contents of One Entry | Database Table Column Name |
|---|---|---|
| XSTR152 | Single Family Residential | TYPE |
| XSTR57 | Lovely redesigned home with private Zen-like garden | EXTRAINFO |
| XSTR67 | Separate Family Room | MOREINFO |
| XSTR70 | Conventional | OTHERINFO |
| XSTR72 | Wall to Wall Carpeting, Linoleum or Vinyl, Tile | FLOORING |
| XSTR73 | Separate Dining Room | LIVDINROOMINFO |
| XSTR65 | Concrete Perimeter | OUTSIDEINFO |
| XSTR75 | 2 Car Garage, Attached, Uncovered Parking, RV or Boat Parking, Off Street Parking, Garage-Converted | GARAGE |
| XSTR82 | Ceilings insulated, Walls insulted | INSULATION |
| XSTR89 | Built-in Oven/Range Combo, Microwave Oven, 1 Dishwasher, Disposer, 1 Refrigerator | KITCHENINFO |
| XSTR103 | Laundry Area – Inside, Extra Storage | LAUNDRYINFO |
| XSTR121 | Sewer in & Connected | SEWERINFO |
| XSTR122 | 2 or more stall showers, 1 shower over tub | BATHINFO |
| XSTR58 | Geological/Flood report, Press control Report (SPC), Preliminary Title Report. Property Inspection Report, CC&R's | REPORTINFO |
| FEATURE16 | AARONS, BERNARD TR | OWNERNAME |

|  |  |  |
|---|---|---|
| FEATURE17 | 752 Middlefield Rd | PROPERTYADDRESS |
| FEATURE18 | Palo Alto CA | CITY |
| FEATURE19 | 94301 2911 | ZIPCODE |
| FEATURE20 | 155 Heather Ln | MAILINGADDRESS |
| FEATURE21 | Palo Alto CA | MAILINGCITYSTATE |
| FEATURE22 | 94301 2911 | MAILINGZIPCODE |
| FEATURE23 | 5113.98 | CENSUSTRACT |
| FEATURE24 | PA | CITYCODE |
| FEATURE26 | 1788 | APPROXSQFT |
| FEATURE27 | 3332 | PERCENTIMPROVED |
| FEATURE28 | 5/15/1991 | LASTTRANSACTIONDATE |
| FEATURE29 | 0010902238 | DEEDNUMBER |
| FEATURE30 | RM2 | ZONING |
| FEATURE31 | 605454 | TRANSFERVALUE |
| FEATURE32 | 1951 | YEARBUILT |
| FEATURE33 | 313580 | TAXAMOUNT |
| FEATURE34 | 6014 | TAXRATEAREA |
| FEATURE35 | 01 | PROPERTYUSECODE |
| FEATURE36 | 4 | BEDROOMS |
| FEATURE37 | 3 | BATHS |
| FEATURE38 | 7936 | ACTUALLOTSIZE |

## APPENDIX B: GLOSSARY

**Crisp set**: the membership function of a variable with two values, commonly defined as 0 and 1.

**Defuzzification**: modifying a fuzzy set into a numerical value.

**Fuzzification**: conversion of an input value into level of membership for particular membership functions defined for that variable.

**Fuzzy Control**: form of control in which the control algorithm is based on Fuzzy Logic; ex. The FCL file in jFuzzyLogic system has fuzzy control.

**Fuzzy Logic**: compilation of mathematical theories based on the idea of Fuzzy set.

**Linguistic rule**: IF-THEN rule with antecedent and consequent

**Linguistic term**: defined by Fuzzy sets, and given membership function

**Linguistic variable**: variables that takes values in the range of linguistic term

**Membership function**: function that conveys how much an element of a set fits in a given Fuzzy subset.

APPENDIX C: Source Code

**jFuzzyLogic Code to access membership functions**

This java file (House.java) uses the FCL defined file and obtains the membership values of the inputs passed into the specified fuzzy rule set:

```
import net.sourceforge.jFuzzyLogic.FIS;
import net.sourceforge.jFuzzyLogic.rule.FuzzyRuleSet;

/**
 * Test parsing an FCL file
 * @author Mandeep Jandir
 */
public class House {

      public double getRating(String var, String member, int num)
      {
            // Load from 'FCL' file
            String fileName = "C:/Program Files/Apache Group/
                               Apache2/htdocs/realestate/
                               jFuzzyLogic/fcl/house.fcl";

            FIS fis = FIS.load(fileName, false);

            if( fis == null )
            {
                  // Error while loading?
                  System.err.println("Can't load file: '" +
                                          fileName + "'");
                  return 0;
            }

            // Show ruleset
            FuzzyRuleSet fuzzyRuleSet = fis.getFuzzyRuleSet();

            //displays in a GUI, the graphs of the rules
            //fuzzyRuleSet.chart();


            // Evaluate fuzzy set
            fuzzyRuleSet.evaluate();

            return
fuzzyRuleSet.getVariable(var).getMembership(member);
```

```
            //Prints ruleSet
            //System.out.println(fuzzyRuleSet);
      }

      public static void main(String[] args) throws Exception
      {
            House h = new House();
            double r = h.getRating("price", "sixfiftyK", 579000);
            double r = h.getRating("beds","four",3);
            System.out.println(r);
      }
}
```

## SynonymFinder.php file

This file parses the MSN Encarta Thesaurus website for the synonyms of the specified keyword. Using patterns, the synonyms of the input keyword are obtained.

```
<?php

/*
 * Function: get_synonym
 * Purpose : retrieves all synonyms from website
 * Input   : 2 parameter( a word, and web-Link
             http://encarta.msn.com/thesaurus_/)
 * Output  : List of synonyms
 */
function getsynonym($description, $web_url)
{
   $words = array();
   $url = $web_url . $description . ".html";
   $data = file_get_contents($url);
   $newlines = array("\t","\n","\r","\x20\x20","\0","\x0B");

   $content = str_replace($newlines, "",
                             html_entity_decode($data));
   $count=0;

   preg_match('/<font class="wordWheelHighlight">(.*)<td
                colspan="3" class="Copyright">/i', $content,
                $match);
```

```
    $result = $match[1];
    $temp = "";

    for($i = 0; $i < 2; $i++)
    {
        $start = strpos($content,'<b><i>Synonyms</i></b>:',$end);

        $end = strpos($content, '<br><br><b><i>Antonym</i></b>:',
                         $start);

        if($end == "")
            $end = strpos($content, '</span></td></tr></table>',
                                                        $start);
        $table = substr($content,$start,$end-$start);

         // extracting string after symbol ':'
        $pattern1="/Synonyms(.*):\s(.*)/";

        preg_match($pattern1, $table, $arr1);

        if($temp != "")
           $temp .= ", ".strip_tags($arr1[2]);
        else
           $temp .= strip_tags($arr1[2]);
    }

    $words = explode(", ", strip_tags($temp));
    return $words;  // return synonyms
}

$weburl = "http://encarta.msn.com/thesaurus_/";
$description = "renovate";
$res = array();

$res = getsynonym($description,  $weburl);
$count_2 = count($res);

echo "<BR><b>Synonyms for ".$description.":</b>\n";
for($i=0;$i<$count_2;$i++)
{
    echo $res[$i]." \n";
}

?>
```