December 2006

A Little Help can Be A Bad Thing: Anchoring and Adjustment in Adaptive Query Reuse

Gove Allen
Tulane University

B. Jeffrey Parsons
Memorial University of Newfoundland

A LITTLE HELP CAN BE A BAD THING: ANCHORING AND ADJUSTMENT IN ADAPTIVE QUERY REUSE

Human-Computer Interaction

Gove Allen
Tulane University
New Orleans, LA
gallen@tulane.edu

Jeffrey Parsons
Memorial University of Newfoundland
St. John's, NL, Canada
jeffreyp@mun.ca

Abstract

The anchoring and adjustment heuristic has been shown to be a pervasive technique that people use in judgment, decision-making, and problem-solving tasks to reduce cognitive burden. However, reliance on the anchoring heuristic often leads to a systematic adjustment bias, in which people fail to make sufficient adjustments for a particular task. In a study involving 157 subjects from six universities, we examined the effect of this bias on SQL query formulation under varying levels of domain familiarity. Subjects were asked to formulate SQL queries to respond to six information requests in a familiar domain and six information requests in an unfamiliar domain. For some, subjects were also provided with sample queries that answered similar information requests. To adequately adjust a sample query, a subject needed to make both surface-structure modifications that required little cognitive effort and deep-structure modifications that required substantially more cognitive effort. We found that reuse can lead to poorer quality query results and greater overconfidence in the correctness of results. We also show that the strength of the adjustment bias depends on domain familiarity. This study demonstrates that anchoring and adjustment extends to an important area in information systems use that has not been previously studied. We also expand the notion of anchoring and adjustment to include the role of domain familiarity.

Keywords: Reuse, anchoring and adjustment, SQL, query formulation

Introduction

The opportunities for, and potential benefits of, reusing various systems development artifacts has received a great deal of attention in the information systems and software engineering fields (e.g. Frakes and Terry 1996; Kim and Stohr 1998; Mili et al. 1995). Interest in reuse originally focused on code (Cox 1990), but in recent years has broadened to include issues surrounding the reuse of other systems development artifacts, such as analysis and design products. Frakes and Terry (1996) go as far as to suggest 10 kinds of artifacts that can be reused in systems development.

Notwithstanding the broad interest in reusing many information systems artifacts, research on reuse has ignored issues associated with reusing database queries. This is surprising given the ubiquity of relational databases and of SQL as a query language standard. Queries written to satisfy one information request can be adapted to satisfy similar requests on the same database. Moreover, queries are frequently embedded in a variety of programming languages (hence, they can be reused across applications written in the same or different languages) and are even portable across database management systems to some extent. Finally, SQL queries can be written by people who have relatively limited training in the language (Fagan and Corley 1998). In view of these considerations, query formulation is an area in which the potential for reuse is tremendous.

Reuse has been recognized to offer the prospect of numerous benefits, including improved quality (Frakes and Succi 2001), decreased development time and cost (Griss 1993), and greater user satisfaction (Succi et al. 2001). However, there is much disagreement in the literature on the success of reuse initiatives, with conflicting evidence pointing to success (Lim 1994) and failure (Fichman and Kemerer 1997), and some authors pointing to the need for healthy skepticism in evaluating the potential for reuse (Irwin 2002).

Challenges to successful reuse have generally been considered in terms of technical issues such as the development and maintenance of repositories to store reusable artifacts and the provision of effective classification and search mechanisms to retrieve potentially reusable artifacts, as well as management issues related to providing incentives for reuse (Fichman and Kemerer 1997; Kim and Stohr 1998; Morisio et al. 2002; Pittman 1993; Purao et al. 2003).

While these challenges no doubt apply in the context of database queries (e.g. Fagan and Corley [1998] describe a case-based reasoning system developed to locate relevant reusable queries in an SQL repository), we believe that cognitive factors constitute a compelling challenge to effective reuse. The use of cognitive heuristics that lead to cognitive biases recently has been found to affect the reuse of both source code and conceptual schemas. In a laboratory experiment, Parsons and Saunders (2004) found that subjects tended to anchor to artifacts they were attempting to reuse, and failed to make adequate adjustments to deal with errors and omissions (with respect to stated requirements) in these artifacts.

SQL requires much less training to use than traditional programming languages and is used in day-to-day work by people who are not computing professionals. However, studies have shown that novices have difficulty formulating queries correctly from scratch (e.g. Chan et al. 1993; Welty 1985). In that context, adapting existing queries is a viable approach to satisfying information requests. At the same time, the potential risk associated with the impact of anchoring and adjustment on query reuse is high.

This paper examines the impact of anchoring and adjustment on adaptive SQL query reuse. We begin by describing the anchoring heuristic and adjustment bias, and indicate how they can be manifested in information systems development. We then consider the context of database query reuse. Next, we describe an experiment designed to examine anchoring and adjustment in the database context. We present the results of the study and conclude by considering the implications of our findings.

Anchoring & Adjustment in Systems Development

Humans adopt strategies to facilitate judgments and simplify problem solving in complex situations. Such strategies are generally referred to as cognitive heuristics. Heuristics reduce the cognitive burden associated with a decision-making or problem-solving task, and are important coping mechanisms (Plous 1993). However, the use of heuristics has been shown to result in systematic biases in judgments. Although widely studied in cognitive psychology (Tversky and Kahneman 1974), there has been little attention paid to the impacts of cognitive heuristics and biases on information systems development (Stacy and MacMillan 1995).

A recent study (Parsons and Saunders 2004) applied and extended the *anchoring and adjustment* heuristic to the adaptive reuse of code and design artifacts in systems development, that is, reuse requiring the modification of an artifact rather than the "black box" reuse of an artifact that performs a specific task. The anchoring heuristic applies to situations in which people are given a problem and an initial solution (or starting point) and asked to provide a final solution. In such cases, there is considerable evidence that people tend to provide final estimates that are close to the initial estimate, even when the situation calls for significant adjustments to the starting point (Plous 1993). In the context of artifact reuse in systems development, adjustment bias occurs when the final artifact is very close to the starting point, even which the requirements of the situation call for greater changes to the artifact being reused.

Parsons and Saunders proposed three potential forms of anchoring and adjustment in the context of artifact reuse in software development:

1. ***Errors*** involve situations in which the reuse artifact contains functionality that is incorrect with respect to requirements. For example, code that uses an incorrect parameter value such as a tax rate will produce erroneous calculations. Such errors have to be fixed in modifying an artifact to meet the requirements of a new problem.

2. ***Omissions*** involve situations in which the reuse artifact does not contain required functionality as described in a specification. A reuse artifact must be modified or extended to include such omitted requirements if it is to serve its purpose.
3. ***Extraneous*** elements involve functionality provided by a reuse artifact that is not explicitly requested in a requirements specification.

Anchoring to either errors or omissions clearly can result in an adjustment bias, a direct application of the earlier uses of these terms in cognitive psychology. In contrast, extraneous functionality may or may not be a problem in artifact reuse, depending on whether the additional functionality is deemed useful by users. Thus, it constitutes an extension of the anchoring and adjustment phenomenon to the context of artifact reuse in systems development.

Parsons and Saunders (2004) provided empirical evidence of the tendency of developers to include extraneous functionality in their solutions, as well as some evidence of a tendency for errors and omissions in a reuse artifact to propagate to solutions. In short, anchoring and adjustment is a meaningful phenomenon posing a potential challenge that needs to be managed to facilitate effective adaptive reuse in systems development. Surprisingly, however, it has not been studied in the context of query reuse.

Query Reuse

Scope

The ability to use organizational data resources effectively is a source of competitive advantage. Both transactional databases and data warehouses commonly provide end-user access to company data in support of strategy formulation, decision-making, and other management activities (Borthick et al. 2001). Relational database management systems provide the underlying technology for accessing such data resources. Although there has been significant research on multidimensional and graphical interfaces to databases (Speier and Morris 2003), Structured Query Language (SQL) remains the standard for ad hoc query formulation. Accurately composing queries in SQL is a challenging task (Chan et al. 1993; Leitheiser and March 1996), and the detrimental effects of using data from inappropriately formulated queries can be significant.

Since SQL queries are specified using text rather than graphical elements, they are highly reusable. An individual can easily copy one query to use as the starting point to compose a query to answer a different (but similar) information request within a given querying environment. Moreover, since SQL is a standard, queries can be copied between environments on vastly different operational platforms, provided the database schemas have semantically similar elements. Beyond ad hoc query formulation, SQL is commonly embedded in other programming environments such Java, C++, and Visual Basic. When organizations use such environments to access central data repositories, the benefits of using a standard query language increase, as similar queries can be reused in varied development efforts. Many companies now seek the benefits promised by code reuse in general by focusing on SQL query reuse. There are now several commercial and open source tools for building and managing SQL repositories (e.g. netlegger.net, quest.com, keeptool.com, plnet.org).

Dimensions of Anchoring and Adjustment

Describing the potential manifestations of anchoring and adjustment in terms of errors, omissions, and extraneous functionality, consider how these concepts might be exemplified in the context of SQL queries. An SQL query consists of three primary components: projection (attributes specified in the 'select' clause), restriction (conditions specified in the 'where' clause), and join.

Table 1 presents a taxonomy of manifestations of anchoring and adjustment in SQL queries. Note that we consider an error to be a feature of a query that produces incorrect query results. Thus, for example, if an information request calls for a listing of the names and salaries of managers whose salary is greater than $100,000, a query contains an omission if only the managers' names appear in the result but contains an error if the condition on salary is either incorrectly specified or omitted from the query.

Table 1: Manifestations of Anchoring and Adjustment in SQL Queries			
	Error	**Omission**	**Extraneous**
Projection	Incorrect aggregation functions	Requested attributes missing	Unrequested attributes included
Restriction	Incorrect parameter value Incorrect logical operator Missing where condition Extra where condition		Redundant condition
Join	Too few tables Too many tables Incorrect join attributes		Extra join that does not change the result
Presentation	Incorrect ordering of results	Missing order of records returned Incorrect ordering of attributes	Unrequested ordering

As this analysis shows, potential manifestations of adjustment bias in query reuse occur largely with respect to the propagation of errors in the query that is being reused with respect to the information request. Many omissions in a query result in errors, such as omitting a where clause. We do not consider presentation issues presented in Table 1, as these do not reflect problems in the semantics of the query. In addition, we do not consider the presence of extraneous data (e.g. retention of unrequested attributes) in the query result, because it is unclear whether this is undesirable in a reuse artifact (Parsons and Saunders 2004).

Anchoring and adjustment in query reuse has a dimension that has not been examined in previous studies of the heuristic. In general, one's level of familiarity with a domain affects the ability to understand a database containing information about that domain (Khatri et al. 2006; Parsons and Cole 2005). The impact of domain familiarity on anchoring and adjustment has not been considered in previous research, most of which has focused on narrow tasks in domains that are presumed to be unfamiliar. For example, when asking someone to estimate the percentage of a population having a particular blood type and providing an excessively high or low anchor, the presumption underlying an adjustment bias is that the respondent does not know the correct value. In this context, it is an open question whether anchoring and adjustment is a robust phenomenon in familiar domains. Thus, a secondary objective of this research is to examine whether and how domain familiarity influences the tendency to anchor.

Research Methodology

Anchoring and adjustment have been shown to affect reuse behavior in both programming and database design tasks (Parsons and Saunders 2004). We propose that database query tasks are generally more constrained than programming and database design. In particular, the requirements associated with an information request in a query context are typically more explicit and have fewer degrees of freedom than requirements associated with a programming or data modeling task. We predict that the more limited scope of query formulation will result in a clearer understanding of the impact of the adjustment bias in adaptive code reuse than we have seen in prior studies.

Research Model

The research model for this study, presented in Figure 1, is adapted from prior research in the area of user performance in database systems use (Allen and March 2006; Chan et al. 1993). The model asserts that user performance in query formulation is influenced by the characteristics of the data model, the task, the user, and the system.

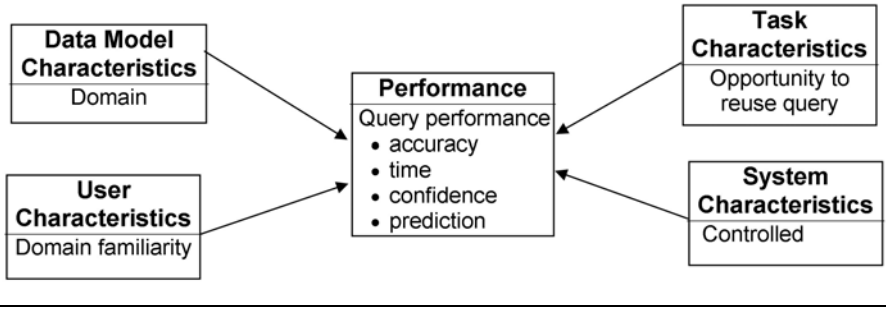


Figure 1. Research Model

This study examines user performance in SQL query formulation under different levels of familiarity for two different domains. Specifically, it examines the effect that the opportunity to modify an existing query to meet the requirements of a new information request has on the accuracy of the query, the time taken to construct the query, the confidence a user has in the accuracy of the query, and the degree to which that confidence predicts the query's accuracy.

Hypotheses

The anchoring and adjustment heuristic and its associated bias suggests that participants who reuse existing queries will anchor to the available query and fail to make all necessary changes to adapt the query to the context of a related information request. This, in turn, will increase the likelihood of errors when reusing a query by adapting it versus when starting from scratch. To observe the anchoring and adjustment phenomenon, two conditions must exist. First, a reasonable anchor must be available. Second, the anchor must be outside the solution's plausible range as judged by the individual (Epley and Gilovich 2006). Simply put, the anchor must neither be too different nor too similar to the correct solution. Accordingly, the scope of each of our hypotheses is limited by the requirement that the query available for adaptive reuse be neither radically different from the correct solution nor very nearly correct. We acknowledge that there may be interesting empirical questions relating to the adaptive reuse of queries that do not meet these constraints; however, these questions are outside the scope of the current study.

Following Epley and Gilovich (2006), we expect that once an adapted query produces results and appears similar to the information request in terms of tables, attributes, and restriction conditions, participants will view the query as a plausible solution to the information request and will cease comparison between the query semantics and those of the information request. When no example is available for modification, participants are forced to examine the semantics of both the information request and the database schema to produce a plausible solution. Of course, some users will introduce their own errors as they construct plausible solutions without assistance; however, as long as the required query is not beyond the user's capability[1], we expect the presence of the anchor to result in lower performance than users will achieve when formulating queries without assistance. Thus, we propose:

H1: Query writers will be less likely to produce queries that correctly satisfy given information requests when they modify existing queries than when they compose queries without assistance.

We further expect that when individuals have a query that satisfies a similar information request to use as a starting point to respond to a separate information request, they will be able to arrive at their solution more quickly because a substantial portion of the time spent in query formulation is taken by mapping the terms in the information request to the terms in the database schema. If a reused query produces results, and is similar to the information request in terms of tables, attributes, and restriction conditions, little time will be spent on verifying the correctness of the query. Moreover, users will be able to copy terms from the query or the query in its entirety rather than re-typing them, yielding additional time savings. In contrast, the effort needed to construct from scratch a query that executes and appears to satisfy the information request can be substantial. Accordingly we propose:

[1] Subjects were students in database courses and were invited to participate only after their professors attested that the required queries were within student capability.

H2: Query writers will formulate queries to answer information requests more quickly when they reuse existing queries than when they compose queries without assistance.

Additionally, we expect that when query writers reuse an existing query, the cognitive burden to develop the query will be lower than when they build a query from scratch. They will be able to verify that the query executes and produces results by running it and will have evidence it is related to the information request in the current task. Accordingly, the task will seem easier and will appear to have been adequately discharged. This perception will result in individuals having higher confidence in the correctness of their queries, much as subjects tend to have higher confidence in easier queries than they do in more difficult queries (Chan et al., 1993). Based on this reasoning, we propose:

H3: Query writers will be more confident in the correctness of their queries when they reuse existing queries than when they compose queries without assistance.

Although confidence has been traditionally studied in query formulation studies (e.g. Borthick et al. 2001; Chan et al. 1993; Leitheiser and March 1996), its value (considered in isolation) is of questionable managerial significance. High confidence is good only when the answer a query produces is the correct one. It will lead a decision maker to accurately weight the quality of the query's result. However, if the answer is incorrect, high confidence can be dangerous, as it may lead to unwarranted confidence in decisions based on bad data. It is only when confidence is considered in conjunction with the accuracy of a query that meaningful conclusions can be drawn. When a query produces an incorrect answer (or answers a different question), low confidence is much preferred to high confidence because it will lead the decision maker to place lower value on the quality of a query's result. Accordingly, we are interested in how well an individual's confidence in the correctness of a query predicts its true correctness.

In the context of query reuse, we expect that the cognitive ease with which a working query is developed when an existing query is reused will lead to more overconfidence than when a query is built from scratch. That is, an individual will be more likely to have high confidence in an incorrect query under conditions of reuse. We state this hypothesis formally as follows:

H4: Confidence in query correctness will better predict actual query correctness when queries are written without assistance than when existing queries are reused

Domain familiarity has been shown to be a factor affecting artifact creation and use in information systems development (Khatri et al. 2006; Parsons and Cole 2005). In the context of query reuse, we expect domain familiarity to mitigate the adjustment bias when participants choose to reuse queries. Information requests over a familiar domain allow query writers to evaluate the plausibility of a query that is being reused. They can focus attention on the match between the familiar domain constructs expressed in the information request and those in the query that is being reused, and thereby be better able to make necessary changes. In contrast, when a domain is unfamiliar, query writers are less able to evaluate the suitability of the reused query, making an executing query that produces results (that appear to match constructs in the domain) more plausible. Thus, we propose:

H5: Query writers will be more likely to adequately adjust an existing query to meet new information requirements in a familiar domain than they will be in an unfamiliar domain.

As Table 1 suggests, there may be varying levels of difficulty in detecting and making appropriate adjustments to anchors present in a query that is being reused. For example, an incorrect parameter value (e.g. Salary > 50000 instead of Salary > 10000) is a relatively simple, or "surface-structure," anchor that should be easy to identify, and to adjust. Moreover, attention is naturally drawn to such superficial differences when comparing parameters in an information request to a query being considered for adaptive reuse. In contrast, a required change in a join expression is a "deep-structure" anchor that will require more cognitive effort both to identify and to modify correctly. Thus, we expect:

H6: Query writers will be better able to identify, and to adjust from, surface-structure anchors than deep-structure anchors.

Research Method

To test these hypotheses, we conducted an Internet-mediated experiment involving subjects from six universities in the United States. To administer the experiment, we developed a query formulation system that allows individuals using a web browser to build, execute, and review the results of queries. The system allows researchers to control

the databases to which subjects have access, as well as the database management system (e.g. Oracle, SQL Server) that processes submitted queries. The system was used in two pilot studies and many homework exercises. After each pilot study, modifications were made to both the functionality of the system and the experimental treatments.

Ultimately the experimental tool administered 12 information requests. Four of these were used to allow subjects to become familiar with the query formulation environment and the database schemata. As such, they contain no experimental manipulation beyond the fact that they exist in different domains, and results from these four queries were used only in calculating the tendency for subjects to anchor as well as their ability to adjust correctly from the anchor.

Independent Variable: Opportunity to Reuse Query

The primary goal of this study is to evaluate the degree to which the anchoring and adjustment cognitive heuristic and its associated bias affect user performance in the adaptive reuse of SQL queries. Accordingly, the ability of the experimental environment to manipulate the opportunity to reuse a query is critical. To meet this need, the query formulation system has the ability to display an arbitrary number of completed queries that a subject might use as a starting point for a given information request. The complete set of information requests is reproduced in Appendix A and the complete set of example queries is reproduced in Appendix B. For the eight information requests that were used in the manipulation of this independent variable, four were given to each subject without any accompanying query upon which the subject might anchor. The other four information requests were presented along with two possible queries that the subject could reference and use in the process of composing a query to answer the information request. Two queries were presented so that the subject had the option to choose a query to use (if any). Subjects were informed that the example queries were written by a competent employee. We felt it inappropriate to present the subject with a single potential anchor query because subjects may have taken the presence of a single query as a cue that it *should* be used, and been more likely to accept its validity at face value. Presenting two queries required participants to engage in some level of analysis of the queries to decide which was a better fit to the information request. Each query in the pair presented to a user required the same transformation if it was to be successfully modified to meet the information request. The two queries differed only in layout and in the syntax used to accomplish the join. One used the SQL '87 syntax that lists tables in the "from" clause and specifies the join conditions in the “where” clause; the other used the SQL '92 syntax that specifies the complete join in the “from” clause.

The information requests were constructed in pairs of equal complexity. Consider the following pair:

> How many times have professors employed students from Arizona (state='AZ')?
>
> How many sponsors are the primary sponsors for courses that can be taken for credits ranging from 1 to 3 (credit_range='1-3')?

As seen below in the pair of queries that satisfies these two requests, they can be answered by queries that differ only in the tables and fields that are referenced. They both require a single join, a single restriction, a single summary, and a single projection.

```
select count(*) from student s
join employment e  on s.id=e.student_id
where s.state='AZ'

select count (*) from course c
join sponsor s on s.id =c.primary_sponsor_id
where credit_range='1-3'[2]
```

For these two information requests, half the subjects received two example queries for the first and half received two sample queries for the second. To demonstrate, we show the pair of example queries that was presented for the first:

```
select max(wage)
from student s,  employment e, work_experience w
```

[2] Credit_range is a text field. Courses that can be taken for varying credits have a text value indicating range.

```
where s.id=e.student_id
  and w.student_id=s.id
  and s.state='AK'


select max(wage)
from student s
  join employment e  on s.id=e.student_id
  join work_experience w on w.student_id=s.id
where s.state='AK'
```

To correctly modify either of these example queries to meet the requirements of the first information request, a subject needs to change the "Max(wage)" function to "count(*)," change "AK" to "AZ" and remove the reference and join to the "work_experience" table. In this way, each subject had the opportunity to reuse an existing query for an information request of a given complexity and was required to formulate a query of identical complexity for a parallel information request without exposure to a set of example queries. Because subjects were randomly assigned to a treatment that determined which example queries were presented, the possibility that one of the information requests is inherently more easily satisfied than another is controlled.

Independent Variable: Domain Familiarity

Because this study hypothesizes that the effect of anchoring and adjustment on query reuse will be different for different levels of domain familiarity, we needed a way to evaluate subjects' performance under different levels of domain familiarity. Although a few studies have examined the effect of domain familiarity on the ability of an individual to reason about a conceptual representation of that domain (Khatri et al., 2006; Parsons and Cole, 2005; Burton-Jones and Weber, 1999), we are aware of no studies that have tested the effect of domain familiarity on the performance of users' query formulation. Indeed, it is difficult to study empirically because it is very difficult to randomly assign subjects to a level of domain knowledge. In one possible approach to examine the influence of domain familiarity, a researcher randomly assigns domain novices to a given domain and then gives them enough training and experience to develop their domain knowledge to a level significantly higher than other subjects. Then the researcher could have the newly created domain experts formulate queries to fulfill the same information requests as are presented to the domain novices and evaluate the difference in their performance. This approach is extremely time-consuming, and it is difficult to get subjects to participate.

We have developed a technique that not only allows a researcher to randomly assign subjects to a level of domain familiarity without the need for domain training, but it also allows us to make that assignment on a within-subjects basis. Our approach builds on a similar approach used to study the role of domain familiarity in understanding conceptual schemata (Khatri et al., 2006).

To accomplish this, we began with a real database that is expected to be of low domain familiarity for our subjects. We selected the Gene Ontology database (http://www.geneontology.org) because the number of tables is not so small that it is trivial nor so large that it is incomprehensible. This database is freely available to the scientific community, and it is used by genetic researchers as a controlled vocabulary to describe gene and gene product attributes in organisms. We implemented a copy of the database on one of our database servers and constructed a schema diagram to reflect its contents (Appendix C). Tables in the database that had no data were omitted from the diagram and dropped from the schema.

Next, we took the schema diagram and removed the names of the tables and attributes, leaving the structure (position of tables, number of attributes in each table, relationships between tables, cardinality of relationships) intact. We used this structure as the framework to construct a database schema for university registration system, a domain with which we expected our subjects (all university students) to have moderate to high familiarity. We mapped semantics of the university domain to the structure taken from the gene ontology schema without modification. To reduce the likelihood that subjects would recognize the schemata as structurally identical, we converted the university schema diagram to its mirror image (Appendix D). We then built the university database and populated it with data taken from a real university registration system. The amount of data is similar in both implemented databases; that is, both have some tables with records numbering in the hundreds on the low end, and both have tables with records numbering in the hundreds of thousands on the high end.

For the purpose of this study, we have two databases that are structurally identical, differing only in the semantics of the domains they represent. This allows us to formulate a query in one database and to easily construct its counterpart in the other. Structurally the two queries are identical; they reference tables in analogous positions in the two schema diagrams. However, they answer different questions because their semantics are based on different domains. With the two parallel queries, we can compose an information request that is satisfied by each. Effectively, we have a single structural information request expressed using the semantics of different domains in parallel schemata. These two information requests can now be given to a single user. Since the requests and schemata differ only in domain semantics, we can essentially ask that user to formulate the same query twice, once under a condition of low domain familiarity and once under a condition of high domain familiarity. To the extent that the subject does not realize that the domains are structurally equivalent, we can examine the difference in performance between the two query formulation tasks as determined by differences in domain familiarity.

In our study, each information request presented to a subject has an analog in the parallel domain. To protect against the possibility that learning in one domain might affect performance in the other domain, the order in which the subjects experienced the domains was randomized. Subjects completed all the information requests in one domain before moving on to the other domain.

Controlled Variables

The two remaining factors from Figure 1 that affect query formulation performance (user characteristics and system characteristics) are controlled in the current study. User characteristics are controlled by the within-subject nature of the experimental design. Since comparisons between experimental treatments will be made on an individual basis, the way in which different user characteristics affect the treatments will be completely balanced. Subjects were drawn from introductory database management courses; therefore, they have only limited experience in formulating ad hoc queries. To encourage participation, students were given course extra credit for participating in the study. They were either given credit for their actual performance or they were instructed that they would receive credit as long as their performance met a certain level. Ultimately, 157 subjects from six different universities in the United States produced usable results.

All subjects used the same system and were exposed to the same information requests, so there was no variation on this variable. The use of a query formulation system that allows subjects to execute and get results to queries is important for the current study because it allows subjects to evaluate the efficacy of modification they make to any query they might be reusing. Moreover, it provides a degree of realism for the study because in practice, query formulation is almost universally conducted in an environment capable of executing proposed queries. The use of an experimental environment that integrates the execution and evaluation of queries is also important because it allows for the precise tracking of time that subjects spent working on each part of the experimental task.

Dependent Variables

The first measure of query performance is accuracy. In the current study, accuracy is measured as a dichotomous variable indicating whether the submitted query correctly answers the information request. Historically query formulation studies evaluated accuracy using some kind of scale that indicates "degree of correctness." (e.g. Batra et al. 1990; Borthick et. al. 2001; Bowen et al. 2004 ; Chan et al. 1993). This kind of scale is often helpful in increasing the variability of the measure, which can help in statistical analysis. However, such measures have typically been based on how close the query is to being correct, not how close the query's answer set is to being correct. In such a scoring scheme, the use of a greater-than sign (>) instead of a less-than sign (<) would be scored as a minor error, and yet such an error could yield a result set that is exactly the opposite of what is requested. The effect of relying on the results of an incorrect query can only be considered by an evaluation of the result set itself, not on an evaluation of the query. Accordingly, we evaluate each query as either producing the correct result set or not. As an additional benefit, this evaluation can be automated and does not rely on human judgment for its value. Moreover, in the context of examining the reuse of an existing query, evaluating the correctness of a query based on how many changes are required to produce the correct answer may also be inappropriate because the example queries that are provided to subjects may themselves require very little modification to correctly meet the demands of a given information request.

In this experiment, each subject was given six information requests in each domain (Appendix A). Of the six, the first two contain no experimental manipulation. They are provided to allow subjects to gain some experience

working with the domain and the web interface before they begin the portion of the experiment that is affected by the treatment. Of the four remaining questions two are simple, requiring only a single join, and two are complex, requiring three joins. In each pair of information requests, a subject is provided with sample queries for one of the two, determined by random assignment. Thus, in each domain, a subject is asked to formulate a simple and a complex query in the presence of a set of suggested example queries as well as a simple and a complex query without any suggested example. Although the paired queries within a complexity level were of similar structure (same number of joins, restrictions, projections, and presentation constraints), we acknowledge that one may have been easier than another because it may have involved a portion of the database schema that was more familiar to subjects.

The second dependent variable used to measure query performance is the amount of time taken to produce queries to respond to the information requests. This variable is simply measured as the cumulative number of minutes a subject spends to respond to individual information requests. The third variable used to measure query performance is confidence. Confidence is measured on a self-reported, five-point Likert scale. After subjects record their query as answering a particular information request, the system prompts them to express their confidence in the correctness of that query. A subject cannot advance until the confidence has been asserted.

The final measure of query formulation performance is the degree to which subjects' confidence in the correctness of their queries predicts the queries' actual correctness. To measure this, we use mean probability score (Yates 1990). This measure is bounded by zero and one where zero equals perfect prediction.

Experimental Results

Hypotheses Tests

Because each hypothesis is evaluated on a within-subjects basis, we calculate the performance differential for each subject on each measure. In Hypothesis 1, for example, each subject gets a score that is calculated as the difference between the number of correct answers produced in the presence of a set of reuse artifacts (example queries that could be selected and adapted to meet the information request) and the number of correct answers produced without assistance. Student's T-test statistic is then used to determine if the mean within-subject variation is different than zero. The results are presented in Table 2.

Table 2. Summary of Tests of Hypotheses

Hypothesis		Student's t	P-value	Result
1.	Reuse of queries results in more errors than starting from scratch	9.4	.0001	Supported
2.	Reuse of queries results in less time to prepare queries than starting from scratch	10.5	.0001	Supported
3.	Reuse of queries results in higher confidence in query correctness	0.4	0.6981	Not Supported
4.	Reuse of queries leads to poorer relationship between confidence and correctness	5.3	.0001	Supported
5.	Domain familiarity reduces adjustment bias	0.1	.8878	Not Supported*
6.	Surface-structure anchors result in less adjustment bias than deep-structure anchors	21.6	.0001	Supported

* see post hoc analysis for clarification

Hypothesis 1 is supported. This result is extremely compelling. It means that when individuals compose queries from scratch, they are more likely to produce the correct answer than when they modify an existing query, even when, as is the case in this study, the query they modify contains most of the required semantic elements of the information request. Average performance in our sample showed that subjects generated the correct answer 44.3

percent of the time when no sample query was presented and only 21.5 percent of the time when an example was available.

Hypothesis 2 is supported. This result is clearly what would be expected. Given a functioning query that nearly meets the requirements of a specific information request, users are clearly able to formulate their solutions more quickly. Over the eight queries that comprise the experimental treatment, the average time difference was about 10.5 minutes. Given that the mean time to complete all eight queries was about 25.5 minutes, the presence of a sample query results in a 40 percent time savings on average.

Hypothesis 3 is not supported. We did not observe a significant difference in the confidence expressed by subjects when they had an example query as compared to when no sample was present. In our sample, the average confidence expressed for queries when there was an example query available was 3.53 on a scale of one to five, and the average confidence expressed for queries without an example was 3.55.

Hypothesis 4 is supported. The results show that when users modify an existing query to meet new information requirements, they are less likely to correctly assess their own query's accuracy. This result arises because users are overconfident in general. As a result, given a constant level of confidence, the treatment that produces higher accuracy will result in the confidence better predicting query correctness.

Hypothesis 5 not supported. This means that overall, users are not significantly better at adjusting from anchors under conditions of domain familiarity than under conditions of domain unfamiliarity. This hypothesis was proposed without regard to the kind of anchor (surface-structure versus deep-structure). As discussed below in our report of post hoc analysis, this distinction has major implications on the ability to adjust successfully under conditions of varying domain familiarity. For the finding of Hypothesis 5 to be meaningful, we must be confident that our manipulation of subjects' level of domain familiarity was successful. In an exit survey using a scale of 1 to 5, subjects indicated their prior level of familiarity with the Gene Ontology domain to be low (mean=1.3, standard deviation=.59) and their prior familiarity with the university domain to be moderate (mean=2.9, standard deviation=1.19). This difference is significant at $p < .0001$.

Hypothesis 6 is supported. In our sample, subjects correctly adjusted from 85 percent of surface-structure anchors and only 31 percent of deep-structure anchors. Although the hypothesis that surface-structure modifications can be made more readily than deep-structure modifications seems self-evident, the empirical results demonstrate it convincingly.

Post Hoc Analysis

As is often the case, in testing the hypotheses that motivated this study, we came up with new questions that we could investigate with the data collected. In some cases, these analyses were conducted to understand more clearly the implications of the findings for a particular hypothesis test; in others, the tests were conducted to answer ancillary questions that arose during hypothesis testing. In all cases, the tests help us better understand the implications of the anchoring and adjustment heuristic and bias on adaptive query reuse under differing levels of domain familiarity.

Table 3. Decomposition of Hypothesis Test 5

Post hoc Test 1	Student's t	P-value	Result
Effect of domain familiarity on ability to adjust from **surface-structure** anchors	4.3	.0001	Users adjust better for **unfamiliar** domain
Effect of domain familiarity on ability to adjust from **deep-structure** anchors	-2.5	.0148	Users adjust better for **familiar** domain

When we found Hypothesis 5 (that domain familiarity increases ability to adjust) to be unsupported, we wondered if there might be a domain-driven, significant difference in the ability to adjust from either surface-structure anchors or from deep-structure anchors. Accordingly, we conducted two tests similar to those for Hypothesis 5. One test examines the ability of users to adjust from surface-structure anchors under different levels of domain familiarity, and the other examines performance for deep-structure anchors. The results are presented in Table 3.

These results indicate that the reason no significance was found for Hypothesis 5 is that there is a counterbalancing effect. Users are better able to adjust surface-structure anchors under conditions of domain unfamiliarity, and they are better able to adjust deep-structure anchors under conditions of domain familiarity. In our sample, subjects correctly adjusted 88 percent of surface-structure anchors in the unfamiliar domain and 82 percent of surface-structure anchors in the familiar domain. For deep-structure anchors, subjects correctly adjusted 28 percent in the unfamiliar domain and 35 percent in the familiar domain. This finding was unexpected; however, there is some intuition to it. It may be that subjects in an experimental environment expected to spend some time in the adjustment. In the case of domain unfamiliarity, they tend to accept the validity of the joins because the cognitive effort required to validate them is high. As a result, they spend their allotted time identifying surface-structure changes. Such changes involve relatively simple mappings from the information request to values present in the example query, or to attribute names in the database schema. However, when the domain is familiar, users are more willing to spend some of their allotted time validating the joins. The result is that they are able to adjust successfully from the deep-structure anchors to some degree; however, in attempting this validation, they pay less attention to the surface structure. Accordingly, their performance at adjusting surface-structure conditions is better under conditions of domain unfamiliarity.

Although the primary purpose of this study is not to study the effect of domain familiarity on user query formulation performance per se, we believe it to be the first empirical study with treatments sufficient to evaluate this effect rigorously. As such, we examine the effect of domain familiarity on user performance in query formulation on only those queries that were composed without the availability of an example query[3].

A priori, one would expect higher domain familiarity to lead to greater accuracy, higher confidence, and shorter time spent. It is not clear that higher domain familiarity should lead to better self prediction of accuracy. On one hand, a better understanding of the domain should increase a subject's capacity to reason about the domain and lead to a better accuracy at self assessment. On the other hand, since users tend to be overconfident in the accuracy of their own queries, the reduced confidence expected from domain unfamiliarity might result in a stronger relationship between confidence and accuracy. We examine the effect of domain familiarity on each of the performance measures from Figure 1. The results are shown in Table 4.

Table 4. Examination of Domain Familiarity on Query Formulation Performance.

Post hoc Test		Student's t	P-value	Support
2.	Domain familiarity leads to increased accuracy	4.3	.0011	Yes
3.	Domain familiarity leads to increased confidence	5.8	.0001	Yes
4.	Domain familiarity leads to different levels of the degree to which confidence predicts accuracy	0.08	.9337	No
5.	Domain familiarity leads to decreased time	0.28	.7835	No

The support for post hoc tests 2 and 3 are as would be expected. It is somewhat interesting that we did not observe a difference in the ability of users' confidence to predict their own queries' correctness. That we do not observe a difference in time spent may be an artifact of the experimental environment. It is possible that under the test-like conditions of the experiment, subjects thought that they should allocate a certain amount of time for each question. This finding supports our conjecture for subjects' superior performance at identifying and adjusting from surface-structure anchors under conditions of low domain familiarity.

An important contribution of this paper is the technique used to evaluate the effect of domain familiarity in query formulation exercises. However, the efficacy of the use of parallel database structures in different domains hinges on the proposition that subjects do not realize that the domains are identical. To examine this proposition, we asked four questions in the exit survey. These questions required subjects to assert which domain schema had more tables,

[3] The analysis was also conducted with the "example query" questions included. This analysis did not change the significance of any finding, so we present only findings for queries formulated without an example available to increase consistency with prior query formulation studies.

which had more relationships, and which had more attributes. We then asked how confident subjects were that their assessment of the number of tables, relationships and attributes was accurate. Only subjects who answered that there were the same number of tables, relationships, and attributes in both domains and that they were highly confident in their assessment could have determined that the domains were structurally identical. Of our 157 subjects, only one met these criteria. With the permission of that student's professor, we contacted him to inquire if he realized the parallel nature of the domains. He indicated that he had not even considered that a possibility. This indicates convincingly that subjects do not realize the parallel nature of the domains when they are of sufficient complexity and rendered as strict mirror images.

Discussion

This study clearly demonstrates that adaptive reuse of queries is affected by the phenomenon of anchoring and adjustment. Reuse can increase the likelihood of errors relative to writing queries from scratch. Despite having access to a functioning query requiring only slight modifications to address a current information request, it is possible to achieve higher quality by writing a query without reference to an existing one.

In addition, the study shows that not only can query reuse lead to higher error rates, the reuse of queries leads to unwarranted confidence in the correctness of queries. Even though reused queries are more likely to have errors, query writers are not correspondingly less confident of their correctness. This is particularly troubling, since the unwarranted overconfidence increases the likelihood that decisions may be made based on incorrect query results. Consequently, query reuse has the potential to lead to bad outcomes for an organization. This result has strong implications for the appropriateness of strategies involving adapting existing queries to satisfy new requirements. It shows that adaptive reuse cannot be accepted unquestioningly as a strategy for effective support of query writing.

"Time taken" to compose queries is a standard measure of query writing performance; as such, we have formally hypothesized about this measure. However its efficacy in understanding the effects of adapting queries for reuse needs to be tempered. The value of saving a few minutes of a decision-maker's time may seem inconsequential when compared to the potential costs of making decisions on erroneous data. Ceteris paribus, reduced time would be a desirable benefit; however, in the current study its value as a performance measure is dubious.

The implications of these finding extend to the teaching of SQL. Based on our experience teaching SQL, we believe it is normal for students to attempt to solve information requests by adapting existing queries. This is particularly true when learning a new concept, in which case queries that address similar tasks in different domains may be modified for the current task.

The study also demonstrates that the adjustment bias is stronger in the presence of “deep structure” anchors than “surface structure” anchors. That is, participants in our study were less able to adjust from anchors involving joins than from anchors involving projections, restriction conditions, or functions. Thus, if it is known in advance that a query to be reused simply involves, for example, changing a parameter value in a selection condition, the risk associated with reuse may be less than if the query is known to require a change in a join condition, or changes of an unknown nature. Perhaps more importantly, the risk associated with queries involving multiple changes may be high, as people may not look beyond surface structure changes to other parts of a query that may need to be modified.

In addition to demonstrating that anchoring and adjustment is robust when applied to database querying, this research also helps build a deeper understanding of the phenomenon. As noted above, when applied to a complex problem-solving task (as contrasted with simple estimation tasks used in earlier studies of anchoring and adjustment), a distinction can be made between surface-structure anchors and deep-structure anchors. This research shows that adjustment to surface-structure anchors tends to be more successful than adjustment to deep-structure anchors. More work is needed to examine this issue in greater depth.

A further contribution to our understanding of anchoring and adjustment comes in the introduction of domain familiarity as a salient factor that interacts with the type of anchor. In particular, post hoc analysis showed that participants were better able to adjust to deep-structure anchors in a familiar domain than in an unfamiliar domain, and better able to adjust to surface-structure anchors in an unfamiliar domain than in a familiar domain. Thus, we have shown that the notion of an anchor is more complex than previously thought.

A final important contribution of this study is the introduction of a technique to manipulate domain familiarity within the confines of an experimental setting without requiring participants to learn about a domain, as well as to

study this effect on a within-subjects basis. Domain familiarity has been recognized as important in recent studies in information systems development.

The findings of this study should be considered in terms of several limitations. First, the study used students in an artificial setting. Although they have experience in writing SQL queries, the use of students means the results of this study may not generalize to organizational settings in which employees reuse queries. Second, the reuse opportunities in this study do not reflect the range of real-world scenarios in which queries might be reused. Participants were offered only two queries as possible reuse artifacts for each information request. No search was required to locate or match available queries to a particular information request. It is possible the act of searching and matching may help overcome some of the anchoring and adjustment found in this study. Third, the manipulation of domain familiarity, although effective, was not as strong as it might have been. Although students were quite unfamiliar with the gene ontology domain (1.3 on a 5-point scale), they were at best moderately familiar with the university domain (2.9).

Conclusions

This study has extended our knowledge of the role of the anchoring and adjustment heuristic and bias in adaptive code reuse and added specific insights with respect to query formulation. Although subjects complete query formulation tasks more quickly when modifying a query that satisfies a similar information request, the speed comes at a cost of decreased accuracy and increased overconfidence.

This study also extends our understanding of the anchoring and adjustment phenomenon by considering its effects under differing levels of domain familiarity and by examining the differences between surface-structure and deep-structure anchors. Although users are better able to adjust from surface anchors than deep anchors in general, the effect of domain familiarity on these kinds of adjustments is not clear-cut.

To continue to improve our understanding of the role of anchoring and adjustment in adaptive query reuse, more research is needed. We expect that expertise and experience in formulating SQL queries will moderate the insufficient adjustment bias. This study should be extended through the use of subjects with substantial experience in SQL query formulation. Moreover, it is unclear how the results of this study apply to the adaptive reuse of a query writer's own queries.

This study has shown that query writers do not always adjust adequately from surface-structure anchors, but it lacks the power to make any claim about the relative performance of writing queries from scratch and the adaptive reuse of queries that require only surface-structure modification. Along these same lines, it is unclear how a warning about the pitfalls of modifying queries that may require adaptation to join expressions may be able to improve user performance in adjustment from deep-structure anchors.

References

Allen, G. and S. March. "The Effects of State-Based and Event-Based Data Representations on User Performance in Query Formulation Tasks," *MIS Quarterly* (30:2), 2006, pp. 269-290.

Batra, D., J. Hoffer, and R. Bostrom. "A Comparison of User Performance between the Relational and the Extended Entity Relationship Models in the Discovery Phase of Database Design," *Communications of the ACM* (33:2), 1990, pp. 126-139.

Borthick, A., P. Bowen, D. Jones and M. Tse. "The Effects of Information Request Ambiguity and Construct Incongruence on Query Development," *Decision Support Systems* (32), 2001, pp. 33-56.

Bowen, P., R. O'Farrell, and F. Rohde. "How Does Your Model Grow? An Empirical Investigation of the Effects of Ontological Clarity and Application Domain Size on Query Performance," Proceedings of the 2004 International Conference on Information Systems, December, 2004, pp. 77-90.

Burton-Jones, A. and R. Weber. "Understanding relationships with attributes in entity-relationship diagrams," Proceedings of the Twentieth International Conference on Information Systems, 1999, pp. 214-228.

Chan, C., K.K.Wei and K. Siau. "User-Database Interface: The Effect of Abstraction Levels on Query Performance," *MIS Quarterly* (17:4), 1993, pp. 441-464.

Cox, B. "Planning the Software Industrial Revolution," *IEEE Software* (7:6), 1990, pp. 25-33.

Epley, N. and T. Gilovich. "The Anchoring-and-Adjustment Heuristic: Why Adjustments are Insufficient," *Psychological Science* (17:4), 2006, pp. 311-318.

Fagan, M. and S. Corley. "CBR for the Reuse of Corporate SQL Knowledge," *European Workshop on Case Based Reasoning (EWCBR'98)*, *Lecture Notes in Artificial Intelligence*, vol. 1488, 1998, pp. 382-392.

Frakes, W.B. and G. Succi. "An Industrial Study of Reuse, Quality and Productivity," *Journal of Systems and Software*.(57:2), 2001, pp. 99-106.

Frakes, W.B. and C. Terry. "Software Reuse: Metrics and Models," *ACM Computing Surveys* (28:2), 1996, pp. 415-435.

Griss, M. "Software Reuse: From Library to Factory," *IBM Systems Journal*.(32:4), 1993, pp. 548-566.

Irwin, G. "The Role of Similarity in the Reuse of Object-Oriented Analysis Models," *Journal of Management Information Systems* (19:2), 2002, pp. 221-250.

Khatri, V., I. Vessey, V. Ramesh, P. Clay, S. Park. " Understanding Conceptual Schemas: Exploring the Role of Application and IS Domain Knowledge," *Information Systems Research* (17:3), 2006, pp. 81-99.

Kim, Y. and E.A. Stohr. "Software Reuse: Survey and Research Directions," *Journal of Management Information Systems* (14:4), 1998, pp. 113-147.

Leitheiser, R. and S. March. "The Influence of Database Structure Representation on Database System Learning and Use," *Journal of Management Information Systems*, (12:4), 1996, pp. 187-213.

Mili, H., F. Mili, and A. Mili. "Reusing Software: Issues and Research Direction," *IEEE Transactions on Software Engineering*.(21:6), 1995, pp. 528-561, 1995.

Morisio, M., M. Erzan, and C. Tully. "Success and Failure Factors in Software Reuse," *IEEE Transactions on Software Engineering* (28:4), 2002, 340-357.

Parsons, J. and L. Cole. "What Do the Pictures Mean? Guidelines for Experimental Evaluation of Representation Fidelity in Diagrammatical Conceptual Modeling Techniques," *Data and Knowledge Engineering* (55:3), 2005, pp. 327-342.

Parsons, J. and C. Saunders. "Cognitive Heuristics in Software Engineering: Applying and Extending Anchoring and Adjustment to Artifact Reuse," *IEEE Transactions on Software Engineering* (30:12), 2004, pp. 873-888.

Pittman, M. "Lessons Learned in Managing Object-Oriented Development," *IEEE Software* (10:1), 1993, pp. 43-53.

Plous, S. The Psychology of Judgment and Decision Making, New York: McGraw-Hill, 1993.

Purao, S., V. Storey, and T. Han. "Improving Pattern Reuse in Conceptual Design: Augmenting Automated Processes with Supervised Learning," *Information Systems Research* (14:3), 2003, 269-290.

Siau, K., Y. Wand, and I. Benbasat. "A Psychological Study on the Use of Relationship Concept -- Some Preliminary Findings," Proceedings of the 7th International Conference on Information Systems Engineering (CAiSE), Jyvaskyla, Finland, 1995, pp. 341-354.

Speier, C. and M. Morris. "The Influence of Query Interface Design on Decision-Making Performance," *MIS Quarterly* (27:3), 2003, pp. 397-423.

Stacy, W. and J. MacMillian. "Cognitive Bias in Software Engineering," *Communications of the ACM*, (38:6), 1995, pp. 57-69.

Succi, G., L. Benedicenti, and T. Vernazza. "Analysis of the Effects of Software Reuse on Customer Satisfaction in an RPG Environment," *IEEE Transactions on Software Engineering*, vol. 27, no. 5, 2001, pp. 473-479.

Tversky, A. and D. Kahneman. "Judgment Under Uncertainty: Heuristics and Biases," *Science* (185:4157), 1974, pp. 1124-1131.

Welty, C. "Correcting User Errors in SQL," *International Journal of Man-Machine Studies* (22:4), 1985, pp. 463-477

Yates, J. *Judgment and Decision Making*, Prentice-Hall, Englewood Cliffs, NJ, 1990

Appendix A: Information Requests

University Domain:

1. How many rooms are in building 'Gist Hall'?
2. Which courses (id and name) have as overseeing professor 'Young, Todd F.'?
3. How many times have professors employed students from Arizona (state='AZ')?
4. How many sponsors are the primary sponsors for courses that can be taken for credits ranging from 1 to 3 (credit_range='1-3')?
5. List the name and title of professors who have taught sections which had a grader with the last name of 'Bennett' from the city of 'New York'. Remove any duplicates from the list.
6. List the work experience (company and job title) of students of who have also been employed by professors whose title is currently 'Assistant' in the 'University College'. Remove any duplicates from the list.

Gene Ontology Domain:

1. How many species are there of the genus 'Influenza'?
2. Which gene products (id and full_name) are referenced by dbxref 'GeneDB_Lmajor'?
3. How many definitions are there for obsolete terms (is_obsolete=1)?
4. How many species are the primary species for gene products with a symbol of ADH2 (symbol='ADH2')?
5. List the database cross references (xref_key and xref_dbname) that provide evidence of associations involving the term name 'ethanol metabolism' with a term_type of 'biological_process'. Remove any duplicates from the list.
6. List the spiecesdbname and product_count for terms with definitions cross referenced in the 'GOA' database (xref_dbname = 'GOA') with non-listed key types (xref_keytype='none listed'). Remove any duplicates from the list.

Appendix B: Example Queries

All subjects were shown example queries for information requests numbered 1 and 2. For all others, subjects either received examples for odd numbered information requests or for even numbered information requests.

Example Queries for University Domain	
1	select count(*) from room where building = 'University Annex'
2	select course_name from course c, professor p where p.id = c.overseeing_professor_id and name = 'Avina, Kathleen' select course_name from course c join professor p on p.id = c.overseeing_professor_id where name = 'Avina, Kathleen'

3	select max(wage)
	from student s, employment e, work_experience w
	where s.id=e.student_id
	  and w.student_id=s.id
	  and s.state='AK'
	
	select max(wage)
	from student s
	join employment e on s.id=e.student_id
	join work_experience w on w.student_id=s.id
	where s.state='AK'

4	select max(sponsor_name)
	from course c, sponsor s, cross_list l
	where c.primary_sponsor_id=s.id
	  and l.course_id = c.id
	  and credit_range='3'
	
	select max(sponsor_name)
	from course c
	join sponsor s on c.primary_sponsor_id=s.id
	join cross_list l on l.course_id = c.id
	where credit_range='3'

5	select distinct name
	from student st, section s, course c, professor p
	where st.id=s.grader_id
	  and c.id = s.course_id
	  and p.id = c.overseeing_professor_id
	  and last_name = 'Barnett' and city = 'New York'
	
	select distinct name
	from student st
	join section s on st.id=s.grader_id
	join course c on c.id = s.course_id
	join professor p on p.id = c.overseeing_professor_id
	where last_name = 'Barnett' and city = 'New York'

6	select distinct company from work_experience w, student s, supervising v, professor p

	where s.id=w.student_id
	  and v.student_id = s.id
	  and p.id=v.professor_id
	  and p.title='Associate' and college='University College'
	
	select distinct company
	from work_experience w
	join student s on s.id=w.student_id
	join supervising v on v.student_id = s.id
	join professor p on p.id=v.professor_id
	where p.title='Associate' and college='University College'

Example Queries for the Gene Ontology Domain	
1	select count(*) from species where genus='Rana'
2	select full_name from dbxref, gene_product where dbxref_id=dbxref.id and xref_dbname='TIGR_Tba1' select full_name from dbxref join gene_product on dbxref_id=dbxref.id where xref_dbname='TIGR_Tba1'

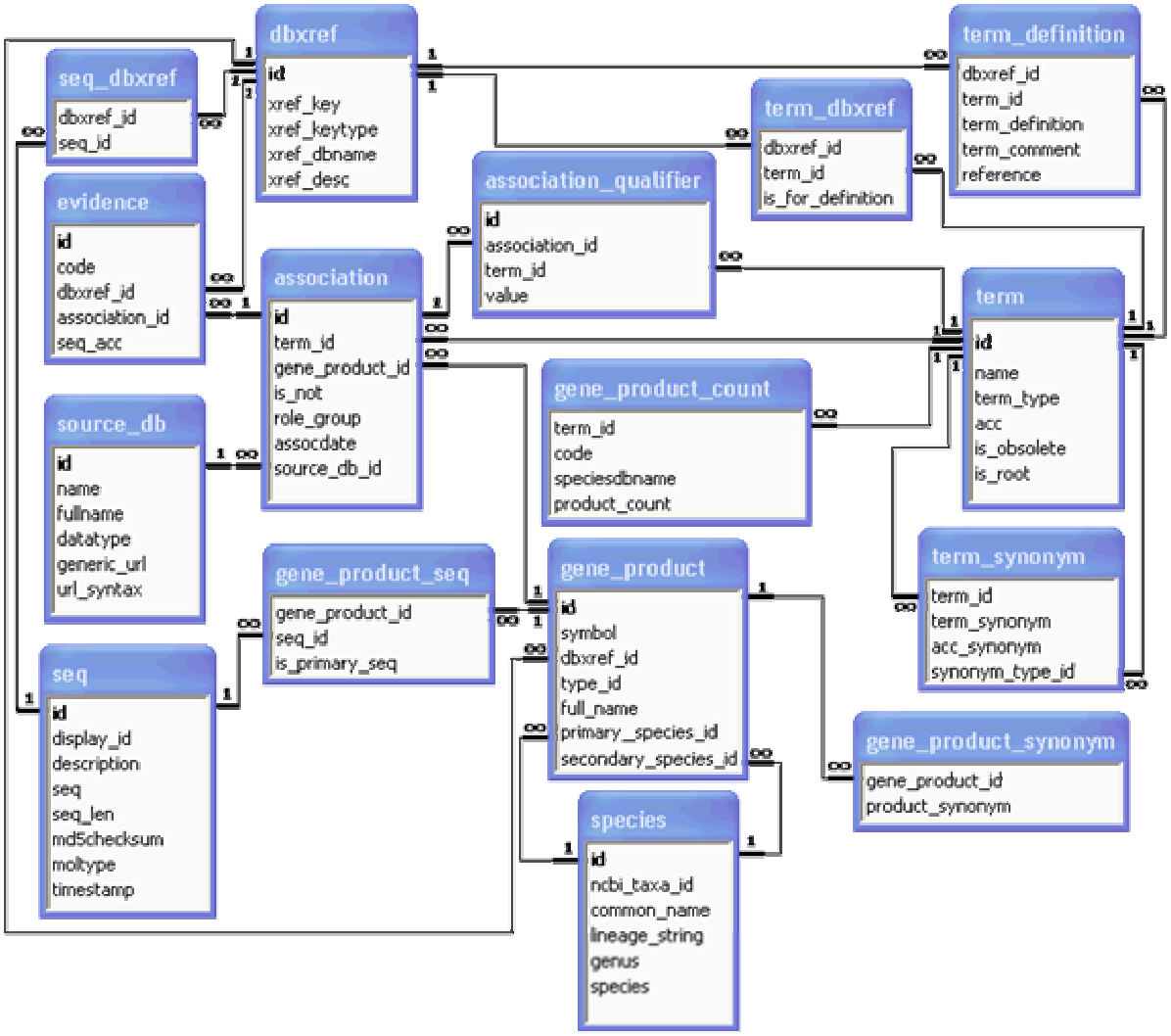
3	select max(term_definition)
	from term t, term_definition d, gene_product_count g
	where t.id=d.term_id
	  and g.term_id=t.id
	  and t.is_obsolete=0
	
	select max(term_definition)
	from term t
	join term_definition d on t.id=d.term_id
	join gene_product_count g on g.term_id=t.id
	where t.is_obsolete=0

4	select max(common_name)
	from gene_product g , species s, gene_product_synonym p
	where g.primary_species_id=s.id
	  and p.gene_product_id = g.id
	  and symbol='ADH1'
	
	select max(common_name)
	from gene_product g
	join species s on g.primary_species_id=s.id
	join gene_product_synonym p on p.gene_product_id = g.id
	where symbol='ADH1'

5	select distinct d.xref_dbname
	from association a, gene_product g, dbxref d, term t
	where g.id = a.gene_product_id
	  and d.id=g.dbxref_id
	  and t.id=a.term_id
	  and t.name='ethanol fermentation' and term_type='biological_process'

	select distinct d.xref_dbname from association a join gene_product g on g.id = a.gene_product_id join dbxref d on d.id=g.dbxref_id join term t on t.id=a.term_id where t.name='ethanol fermentation' and term_type='biological_process'
6	select distinct speciesdbname from dbxref d, term_dbxref td, term t, gene_product_count g Where d.id=td.dbxref_id and t.id = td.term_id and g.term_id = t.id and xref_dbname='PAMGO' and xref_keytype = 'none listed' select distinct speciesdbname from dbxref d join term_dbxref td on d.id=td.dbxref_id join term t on t.id = td.term_id join gene_product_count g on g.term_id = t.id where xref_dbname='PAMGO' and xref_keytype = 'none listed'

Appendix C: Gene Ontology Diagram



Appendix D: University Diagram