CONCEPTUAL LEVEL VERSUS LOGICAL LEVEL USER-DATABASE INTERACTION

H. C. Chan  
National University of Singapore

K. K. Wei  
National University of Singapore

K. L. Siau  
National University of Singapore

Follow this and additional works at: http://aisel.aisnet.org/icis1991

Recommended Citation  
http://aisel.aisnet.org/icis1991/45
CONCEPTUAL LEVEL VERSUS LOGICAL LEVEL
USER-DATABASE INTERACTION

H. C. Chan  
K. K. Wei  
K. L. Siau  
Department of Information Systems and Computer Science  
National University of Singapore

ABSTRACT

User-database interaction can be classified into various abstraction levels. The two higher levels are the conceptual level and the logical level. On the conceptual level, the user talks of things in his world, for example, in terms of entities and relationships. On the logical level, the user talks of things in the database system, for example, in terms of relations and join operations. The user world is something that the user is expected to know, but the data in the database system is something new to the user. Thus, the conceptual level is closer to the user. Therefore, we hypothesized that the conceptual level interaction will be easier for the user. This was tested with the entity-relationship (ER) model for the conceptual level model and the relational model for the logical level model. The results showed that users of the conceptual level had 38% higher accuracy, 16% higher confidence, and took only 35% of the time taken by users of the logical level. The differences were statistically significant with p values better than 0.003. From the user's perspective, the huge differences make it worthwhile to consider the ER model in place of the relational model.

1. INTRODUCTION

There are three major traditional data models: the relational, network and hierarchical. There had been many experiments testing the effect of these models on user query performance (Chamberlain 1980; Reisner 1981; Welty and Stemple 1981; Welty 1985). It is now readily accepted that of these three, the relational model provides the best query language for the user (Date 1982; Goldstein 1982; Lochovsky and Tschritzis 1977). Date (1982, p. 485) writes, "It is generally accepted that non-programming users, at least, will require a relational view of the database."

Because the traditional data models are not expressive enough, many semantic models have been proposed. The most common of these is the entity-relationship model (Chen 1976). There are many claims that the ER model is better than the relational, such as that it is easier to use and understand, or that it is more natural. However, there are very few empirical studies to support such claims.

Instead of considering just specific data models, we group them according to their abstraction levels. An abstraction level model, which is common in the database and information systems fields (Olive 1983), of user-database interaction is described in section 2. The model, though common, has not been used to analyze the empirical results. We use this and argue that the triumph of the relational model over the network and hierarchical models is actually the triumph of one higher abstraction level over another lower abstraction level. We therefore hypothesize that the ER model, which is an example of an even higher level, will be better than the relational model. The hypotheses are verified experimentally.

The paper is organized as follow. Section 2 describes the user-database interaction model. Section 3 gives a review of the few experimental studies that compared the user's performance for the ER and relational models. Section 4 describes the research model, the hypotheses, and the experimental procedure. Section 5 describes and discusses the experimental results. Lastly, section 6 is the conclusion for this study.

2. USER-DATABASE INTERFACE

There are many possible ways to classify user-database interfaces. One may be interested in the mechanics of the interface, such as mouse versus key board or menu versus command-line. In this study, we are interested in the substance of the user-database interaction — what are the things conveyed by the user's input? Are these things about the user's world, or things about memory in the computer? One possible classification of the substance of the user-database interface is by its abstraction level.
There are three main levels: physical, logical and conceptual. The physical level is the lowest while the conceptual level is the highest. This is shown in Figure 1.

At the lowest level, the user is required to know the details of the data structures in the computer memory. He may, for example, have to know the various files and physical pointers that point from data to data, or records to records. His query will typically involve some specification and tracing of physical pointers. This level is exemplified by the interfaces provided by the network and hierarchical data models. Most microcomputer database systems also require the user to know the files that store the data. A database, such as a company database, may be stored in several files and the user must know which file contains what data. The use of indices, the order of the columns or rows also belong to this level.

At the logical level, the user is required to know the layout of the logical data, and the possible relationship among data elements. This is exemplified by the relational database interface (Hawryszkiewycz 1990). There are no physical pointers or files and the order of the columns and rows is not important. But the user knows that by joining relations based on certain fields, it is possible to specify relationships that at the physical level are represented by physical pointers. In fact, it is common to refer to the joins as the logical pointers.

At the conceptual level, the database is supposed to know the user’s world, in terms of entities and relationships. There are no logical pointers for the user to trace. If the user wants to specify a relationship, he may say "where S supplies P" rather than "where s.sno=sp.sno and sp.pno=p.pno." A model that is suitable for this level of interaction is the entity relationship (ER) model. Like the relational model, the ER model is also a formal model. It has many proposed calculi and algebras, such as Atzeni and Chen (1981), Chen (1984), Parent and Spaccapietra (1984), Parent et al. (1989), and Chan (1990). Even the basically relational idea of normalization can be applied with the ER model (Ling 1985).

It should be added that these abstraction levels are different from the three levels proposed by the ANSI/SPARC Study Group on DBMSs. The three levels there — internal, conceptual and external — refer more to the architecture of the DBMS.

The abstraction levels in fact represent some intervals in the semantic range from the user world of entities and relationships to the core of the computer system which is basically in terms of 1's and 0's. For any interaction to occur, the semantic ranges understood by the user and the database system must have some overlap. Hence, the more the database system understands (the higher the level of interaction), the less the user needs to understand. This is the basic reason for hypothesizing that users will perform better at the higher levels.

With this level classification, we can hypothesize that it is not by random design that the relational model is better (for the user) than the network and hierarchical models. It is not just one data model being better than two other data models. It is reasonable to generalize from this and say that the logical level of interaction, which is represented by the relational model, is better than the physical level, which is represented by the hierarchical and network models.

Since the logical level is better than the physical level, it is merely an extrapolation to hypothesize that the conceptual level will be better than the logical level. This, however, needs to be empirically tested.

3. REVIEW

There are very few empirical studies that compare the ER model and the relational model. The study by Batra, Hoffer and Bostrom (1990) compared the user's performance in representing his/her world in the ER model and the relational model. This is about database design. The modeling correctness was measured for different tasks, which were to model various types of concepts,
such as unary and binary relationships. The results showed that for most tasks the ER model allowed significantly more accurate designs. The ER model was perceived by the subjects to be slightly easier to use, though the difference was not statistically significant.

The study by Kenny et al. (1989) was on the user's performance when querying the ER model and the relational model. The users looked at different models (either ER or relational) but they answered with the same query language, SQL. There was no significant difference in the semantic accuracy of the queries. However, users of the relational model took a longer time and made less syntactic errors. The experiment did not make a clear distinction of the level of user-database interaction. SQL is inherently designed for use with the relational model. SQL users must understand the logical pointers and specify the join operations. Hence, in this experiment the users of the ER model, like the users of the relational model, had to go into the logical level and manipulate the logical pointers. This may explain the lack of difference in the semantic accuracy. Furthermore, users answered only four queries, two simple and two complex. A complex query involved a simple join of two relations. This is actually a rather simple query. There are much more complex queries in SQL, such as nested queries and the use of the "GROUP BY" clause.

The Kenney et al. experiment used SQL for both the ER and the relational groups because of the concern that there might be interaction effects between the data model and the query language. However, it is not possible to have one language that suits two data models. If two data models are different, it is because they contain different constructs and their languages must necessarily differ.

For example, let us consider the Unified Database Language (UDL) in Date (1982) designed to provide for relational, network and hierarchical models. It is clearly stated by Date (p. 451) that the language features needed for the relational model form a subset of the language features needed for the hierarchical model; this subset is in turn a subset of the language features needed for the network model. Thus users of different models need to learn the language to different extents and, therefore, interactive effects will not be eliminated even if we use UDL in an experiment.

Another reason against the use of one language for two models is that the language may not be optimal for both. It may be optimal for one, in which case the experiment will be biased. If it is not optimal for either model, then the practical usefulness of the results is doubtful. There will be nagging questions of what will happen if users use the optimal languages in practice.

4. RESEARCH METHODOLOGY

4.1 Research Model

Figure 2 shows the research model for this study. The model asserts that the performance of a database user is likely to be influenced by the following four factors.

1. The characteristics of the data model.
2. The task characteristics.
3. The experience and psychological characteristics of the human user.
4. The physical implementation characteristics of the database system.

![Figure 2. A Research Model](image)

This research model is adapted from Reisner (1981), where a survey of query language studies showed that the laboratory studies on comparison of query languages implicitly considered the following factors: task, data model, human factor and user performance.

The data model affects the way the user views and manipulates the data in the database. It is traditional to say that the user manipulates the data in the database. With a conceptual level interface, it is more accurate to say that the user specifies a query in terms of something
about his/her world, rather than manipulating data in the database. The task can refer to the type of the problem, such as query writing, data modeling or updates, and also the difficulty of the problem, such as simple or complex. The human factor refers to the individual differences, such as computer knowledge, experience or more personal characteristics. The physical implementation characteristics of a database system refer to the actual interaction at the terminal. The variables may include menu versus command line, graphics versus text, color versus black and white, the response time or even the amount of help available.

In this study, these four factors were either manipulated or controlled. The independent variable, which is the abstraction level of the data model, was manipulated to be either conceptual or logical. As we discussed previously, the relational model is an example of the logical model and the ER model is an example of the conceptual level. Unlike the study by Kenny et al., we made the distinction complete by providing an query language, KQL, tailored for the ER model. Hence, the conceptual level was operationalized into the ER model plus the ER query language KQL, and the logical level was operationalized into the relational model plus the ANSI standard relational query language SQL.

SQL was selected also because it is generally considered the best query language for the relational model. For the ER model, there is no commonly used ER query language. Hence KQL was used in this study. Some examples of KQL queries are given in the appendix. Full details of its syntax are available in Chan (1989). It is a definition and query language designed specifically for the ER model, including concepts of is-a (specialization and generalization) relationships and inheritances. The formats of SQL and KQL are very similar. Both are command-line queries with set results. KQL allows for arbitrarily complex queries; it allows nested queries and includes statistical functions such as count and average.

The task, the human, and the database system were controlled. The task was controlled by having the same set of queries for all subjects. The queries covered a very comprehensive range from the very simple to the very difficult. The primary research interest was in the overall query performance. To even out the individual differences, we randomized the subjects into two groups for the two abstraction levels. The database system was controlled by having the same system for both groups. This system was essentially a simple text editor, customized to display queries and record answers and other data. Although there were some valid reasons to use real systems that can parse the answers, point out errors, and return results, there were also some concerns that other factors will invalidate the empirical result. The real systems had to be different and they might not be equally user friendly.

The dependent construct of user performance was operationalized into three variables — the accuracy of the queries, the time taken to formulate the queries, and the subject's confidence in their queries. The accuracy and confidence were measured on a scale from 0 to 5. The time was measured in seconds.

4.2 Hypotheses

Although we suspect that the conceptual level will be better, we nevertheless state the hypotheses in null form. The alternative hypotheses are one-sided.

Hypothesis H1: Subjects in the conceptual level and the logical level groups will show no time difference in formulating the queries.

H1': Subjects in the conceptual level group will take less time than the subjects in the logical level group.

Hypothesis H2: There will be no difference in the accuracy of the queries from the subjects in the conceptual and logical level groups.

H2': The queries from the subjects in the conceptual level group will be more accurate than the queries from the subjects in the logical level group.

Hypothesis H3: There will be no difference in the confidence of subjects in the conceptual and logical level groups.

H3': Subjects in the conceptual level group will show more confidence than subjects in the logical level group.

4.3 Research Method

In this study, an experiment was conducted in a laboratory to investigate the effects of the independent variable on the dependent variables. The experimental plan and the number of subjects successfully completing the experiment is summarized in Table 1.

Independent Variable. This is the level of the data model used in the user-database interaction. It was controlled to be either the conceptual level or the logical level. The conceptual level subjects used the ER model with the ER query language KQL, while the logical level subjects used the relational model with the relational query language SQL.
Table 1. Experimental Plan

<table>
<thead>
<tr>
<th>Abstraction Level of User-database Interaction</th>
<th>Logical Level (Relational Model with SQL)</th>
<th>Conceptual Level (ER model with KQL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 subjects</td>
<td>23 subjects</td>
<td></td>
</tr>
</tbody>
</table>

**Dependent Variables.** These are the time taken to answer the query, the accuracy of the answer and the subject’s confidence in his/her answer. The time was automatically recorded by the computer program. The timing started when the question was displayed on the screen and ended when the subject signaled by typing ctrl z that the answer was completed. The subject’s confidence was a value from 0 to 5, entered by the subject and recorded by the program. The accuracy of the answer was determined separately by two markers. This provided an estimate of its reliability. The accuracy was an overall assessment of the correctness of the answer. Both semantic and syntactic accuracies were considered.

**Training.** Two training booklets were used during the experiment: an SQL booklet for the logical level group and a KQL booklet for the conceptual level group. The training booklets gave a brief overview of the respective data model and the query language. To maintain consistency, the same database domain (suppliers and parts) and the same example queries were used in both booklets.

The same administrator provided separate training for both groups. All examples in the training booklets were discussed. Subjects practiced answering a question after each example. Feedback on the query accuracy was given to improve learning before proceeding to the next example. All questions from the subjects were answered and training continued until the subjects were fully satisfied. The KQL group trained for one hour; the SQL group trained for one and a half hours.

After a ten minute break, the subjects had a practice session writing queries using a practice program similar to the test program. This was to acquaint them with the mechanics of the interface so that, for the real test, they would not have to spend time figuring out how to enter the query, how to report the self-confidence level, or how to get to the next question. The measured time for the real test will then be the real query formulation time. For purposes of consistency, the subjects in both groups were asked to construct the same set of queries using the same training database. The administrator answered all questions from the subjects.

Any biases towards SQL or KQL were eliminated as far as possible. The trainer had no hand in designing SQL or KQL. The subjects were not informed beforehand of the nature of the study. This prevented any subject learning of SQL before the experiment. During the training, the subjects were simply informed that SQL (or KQL) was a database query language. They did not know that SQL was a standard language while KQL was a development product.

**Test.** Finally, the subjects were asked to run the test program. They had to answer ten questions based on a new database domain. The program displayed the questions one by one. Answers were entered directly into the computer via a simple text editor. Subjects could refer to the training material and use paper and pencil to help formulate the answers.

The computer automatically timed the interval between the display of the question and the time when the subject keyed ctrl-Z signifying that the query had been constructed. Immediately after each answer, the subjects were asked for their confidence in their answer. The value ranged from 0 (zero confidence) to 5 (absolute confidence). The answer, the time in seconds, and the confidence level for each question were recorded by the computer. In addition, the subjects had to enter their name at the beginning of the program.

The subjects in both groups were presented identical questions. The conceptual level group had a diagram, on paper, of the ER model. The logical level had a diagram, also on paper, of a set of relations. The ER model and
the set of relations showed the database about departments and employees. The diagrams, the set of ten questions and the sample answers are included in the appendix.

The Subjects Undergraduate students served as subjects for the study. Forty-eight subjects were randomly selected from a population of 480 first year computer science students. They were then randomly assigned to two experimental groups. All of the subjects had used computers before but had no database experience. On average, the students were 20 years old. To motivate the subjects, they were informed that course credit would be awarded based on their speed and accuracy in constructing the queries, and also based on the correlation between the accuracy of their queries and the self-reported confidence level. This encouraged them to report their confidence honestly rather than to show excessive confidence. One subject was absent.

The subjects are representative of users who are intelligent, have some programming experience, and little database training. Users in this group are likely to be at the executive or managerial levels.

5. EXPERIMENTAL RESULTS AND DISCUSSION

5.1 Statistical Results

The accuracy of the subjects' answers was determined separately by two markers. Each answer could get a maximum of five marks and a minimum of zero mark. The marks for the ten questions were totalled to give the score for each subject. The mean scores and the standard deviations (given in brackets) for the subjects in the two groups are shown in Table 2.

The marks were very close; the correlation coefficient for the mean marks of the two markers was 0.96. This showed a high reliability for the measure of accuracy. Hence, only the first set of marks was used for the subsequent tests.

The means and standard deviations (given in brackets) for the three independent variables of time, confidence, and accuracy of the two groups are shown in Table 3.

The t-test was suitable for testing the hypotheses as the independent variable had only two nominal levels. The t values and the significance levels are shown in Table 4. The probability values given are for the one-tailed t-tests.

The two groups had different variances for time, and the same variances for accuracy and confidence. Therefore, the t-test for time was based on the assumption of unequal variances. These tests were done using the SAS software.

The null hypotheses about time and accuracy were rejected at the 0.01% level (which is much better than the commonly acceptable 5%). The null hypothesis about confidence was rejected at the 0.27% level. We therefore accept all of the alternative hypotheses. Users are better at the conceptual level than at the logical level.

5.2 Discussions

The results supported the basic hypothesis that users can perform better at higher abstraction levels. The reason is that the higher levels have semantics closer to the user's world.

A different question is whether the differences are of practical significance. The percentage differences of the KQL group over the SQL group are shown in Table 5.

When compared to subjects of the logical group, the subjects in the conceptual group took 35% of the time, were 16% more confident, and 38% more accurate. These findings should be of practical significance.

The results are in sharp contrast to the results found by Kenny et al. (1989), which showed little difference in the performances using the ER model and the relational model. We believed there are two reasons. The main

<table>
<thead>
<tr>
<th>Table 2. Total Group Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>SQL</td>
</tr>
<tr>
<td>KQL</td>
</tr>
</tbody>
</table>
Table 3. Scores for Dependent Variables

<table>
<thead>
<tr>
<th>Measure</th>
<th>SQL Mean (Std Dev)</th>
<th>KQL Mean (Std Dev)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>2569 (1024)</td>
<td>894 (306)</td>
</tr>
<tr>
<td>Confidence</td>
<td>35.6 (7.5)</td>
<td>41.3 (5.1)</td>
</tr>
<tr>
<td>Accuracy</td>
<td>32.2 (8.3)</td>
<td>44.4 (5.7)</td>
</tr>
</tbody>
</table>

Table 4. T-test Results

<table>
<thead>
<tr>
<th>Measure</th>
<th>t</th>
<th>p (Prob &gt; t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>7.66</td>
<td>0.0001</td>
</tr>
<tr>
<td>Confidence</td>
<td>3.01</td>
<td>0.0027</td>
</tr>
<tr>
<td>Accuracy</td>
<td>5.86</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

Table 5. Group Percentage Differences

<table>
<thead>
<tr>
<th>Measure</th>
<th>Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>-65</td>
</tr>
<tr>
<td>Confidence</td>
<td>16</td>
</tr>
<tr>
<td>Accuracy</td>
<td>38</td>
</tr>
</tbody>
</table>

reason is that a special ER query language, KQI, was used in this study, making the distinction between the ER and the relational model complete.

The other reason is that we tested the subjects with more queries covering a much wider range. In fact, for the simple queries that involve only one relation, our study also showed no significant differences in the accuracies: scores were all close to the maximum score.

It is interesting to note that for all but one question, the SQL group had confidence higher than their accuracy. In contrast, for all but one question, the KQL group had confidence lower than their accuracy. The correlation
coefficient for accuracy and confidence for the two groups are, however, the same. The explanation for this phenomenon may be that, in SQL, it is easier to make mistakes unknowingly. Many previous studies have found that subjects tend to omit the join operations without realizing that these are needed. This was also found in this study. For example, for question 5, one subject specified a join based on employee name and department name directly without the intermediate relation WORK. The accuracy was low (1) but the confidence was high (4). Another subject specified the join using the correct key fields but again without the intermediate relation WORK. The accuracy was low (2) but the confidence was high (5). Some specified one join correctly but omitted the second join and they also reported high confidence levels.

Compared to the study by Kenny et al., our subjects answered the queries much faster. The average time for the SQL group for a single-relation query was only 0.82 minute, compared to the average of 3.53 minutes recorded by Kenny et al. The average for the same queries for the KQL group was 0.49 minute, compared to 2.38 minutes recorded by Kenny et al. Similarly, large differences exist for the one-join queries. It is difficult to explain the large differences since the subjects were all undergraduate students and they had roughly the same amount of training. One possibility may be the different methods of determining the time taken. The study by Kenny et al. used a paper and pencil test, and presumably the timing was self-reported. This study used a computer program to automatically and precisely record the time taken for each query for each individual subject.

This experiment showed that users with relatively little computer experience, immediately after a training session, performed better at the conceptual level than at the logical level. This supports the basic reasoning that users can perform better because they need to understand less at the conceptual level. We think similar results can be expected from other users, such as users with database experience.

6. CONCLUSION

We proposed that the various empirical data on the effect of data models on user performance be generalized according to the abstraction level of the data model. Thus, data concerning the relational model and the network model can be used to support hypotheses about the logical level and the physical level. The new abstraction level that is going to be widely available to the users is the conceptual level. We tested three hypotheses that this level is better than the logical level. The results showed that the conceptual level is actually better. The queries were formulated in 35% of the time taken for logical level queries; they were also 38% more accurate. Furthermore, the users of the conceptual level were 16% more confident. The practical implication is significant. It shows that users can benefit greatly if they can replace the relational interface with an ER interface.

Our study concentrated on the query (retrieval) of information. In addition, Batra, Hoffer and Bostrom (1990) showed that users are better at designing ER models than relational models. Given the few empirical studies in this area, more studies are needed to confirm the findings. Another major aspect of the user–database interaction is the update of information. This aspect also needs to be tested.

Research on data models within an abstraction level is also of interest. Examples include the past studies that compared the hierarchical and the network models, and also studies that compared different query languages for the relational model. Future studies may compare different languages for the ER model, such as comparing a graphical ER language with a command-line ER language, or compare different models that belong to the conceptual level, such as the ER model and other more object-oriented models.

7. REFERENCE


APPENDIX
DATABASE AND QUERIES FOR THE EXPERIMENT

This appendix contains the ER diagram, the relational schema and the set of questions that were given to the subjects. The sample answers to the questions are listed here.

The ER Model:

![ER Diagram](image)

Figure 3. An Entity Relationship Model

The Relational Schema:

Employee(number, name, salary)
Engineer(number, profession)
Manager(number, rank)
Department(number, name, city)
Project(number, name)
Work(employee_number, department_number, date)
Management(manager_number, department_number, date)
Head(engineer_number, project_number, date)
Questionnaire:

1. Show the names and numbers of all employees.
2. Show the department name and city.
3. Show the engineers' numbers, name and professions.
4. Show the name of employees who head any projects.
5. Show the name of employees who work in the research department.
6. Show the name of the employee who work in the same department as Jack.
7. Show the names of the employees with higher salary than Jack.
8. List the names of managers who manages more than one department.
9. List the names of engineers who do not head any projects.
10. List names of engineers who head all projects.

Sample SQL Answers:

1. SELECT NAME, NUMBER FROM EMPLOYEE
2. SELECT NAME, CITY FROM DEPARTMENT
3. SELECT EMPLOYEE.NUMBER, NAME, PROFESSION FROM EMPLOYEE, ENGINEER WHERE EMPLOYEE.NUMBER = ENGINEER.NUMBER
4. SELECT NAME FROM EMPLOYEE, HEAD WHERE EMPLOYEE.NUMBER = HEAD.ENGINEER_NUMBER
5. SELECT EMPLOYEE.NAME FROM EMPLOYEE, WORK, DEPARTMENT WHERE DEPARTMENT.NAME = 'RESEARCH' AND WORK.EMPLOYEE_NUMBER = EMPLOYEE.NUMBER AND WORK.DEPARTMENT_NUMBER = DEPARTMENT.NUMBER
7. SELECT E1.NAME FROM EMPLOYEE E1, EMPLOYEE E2 WHERE E1.SALARY > E2.SALARY AND E2.NAME = 'JACK'.
8. SELECT EMPLOYEE.NAME FROM EMPLOYEE, MANAGEMENT WHERE EMPLOYEE.NUMBER = MANAGEMENT.MANAGER_NUMBER GROUP BY EMPLOYEE.NUMBER HAVING COUNT(*) > 1
9. SELECT NAME FROM EMPLOYEE WHERE NOT EXISTS (SELECT * FROM HEAD WHERE HEAD.ENGINEER_NUMBER = EMPLOYEE.NUMBER)
10. SELECT NAME FROM EMPLOYEE WHERE NOT EXISTS (SELECT * FROM PROJECT WHERE NOT EXISTS (SELECT * FROM HEAD WHERE HEAD.ENGINEER_NUMBER = EMPLOYEE.NUMBER AND HEAD.PROJECT_NUMBER = PROJECT.NUMBER))
Sample KQL Answers:

1. select employee name, number.
2. select department name, city.
3. select engineer number, name, profession.
4. select employee name where employee head project.
5. select employee name
   where employee work department and department name='research'.
6. E1 is employee, E2 is employee
   select E1 name
   where e1 work-related department
   and e2 work-related department
   and e2 name='Jack'.
7. E1 is employee, E2 is employee
   select E1 name
   where E1 salary > E2 salary and E2 name='Jack'.
8. select manager name
   where manager management-related > 1 department
9. select engineer name
   where engineer head-related no project
10. select engineer name
    where engineer head-related all project