AERIAL:
Ad hoc Entity-Relationship Investigation And Learning

L. M. Burns[1]
A. Malhotra[1]
G. Sockut[2]
K.-Y. Whang[3]

[1]IBM Thomas J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598

[2]IBM Santa Teresa Laboratory
P.O. Box 49023, San Jose, CA 95161

[3]Computer Science Department
Korea Advanced Institute of Science and Technology
373-1 Koo-Sung Dong, Daejeon, Korea

Contact: L.M. Burns, (914) 784-7649,
LUANNE@YKTVMH, luanne@watson.ibm.com

Abstract - **This paper discusses Browsing, an alternate style of working with databases, that facilitates the unplanned exploration of the structure and contents of a database by a novice user. We argue that Browsing facilitates learning and helps bring the user's mental model of the problem space into correspondence with the model stored in the database.**

Browsing is illustrated through a learning scenario using AERIAL, a facility that allows the user to browse an Entity-Relationship database. The facilities provided by AERIAL are discussed in detail as well as their use in building a correspondence between the user's model of the problem space and the model stored in the database.

INTRODUCTION

Browsing [1, 3, 4, 9, 10, 16] is a new way of working with databases that has not received as much attention as query. This is probably because, while query mimics an older style of working with databases, browsing is a different style of database access that is made possible by computer-stored databases and visual interface technology. We argue that browsing is much more useful for learning about the structure and content of a database and relating the acquired concepts to existing concepts in the user's mental model. After the user understands the database and has formulated, in detail, the questions that he wants to ask, he can proceed to write queries.

This paper describes AERIAL (Ad hoc E-R Investigation And Learning), a facility to browse Entity-Relationship (E-R) databases. AERIAL is part of a larger system called IRIS (Interactive Repository Interface System) that provides access to a E-R database or repository via a visual interface. Malhotra et al. [13] discuss IRIS and show how the IRIS concepts can be easily and naturally extended to other data models such as Relational.

IRIS uses the schema graph display as the focus of interaction with the user. The window shown in Fig. 1 is the main window for all facilities provided by IRIS. Burns et al. [6] found that database users who spontaneously drew schema diagrams were better able to interact and understand the database than those who did not. Therefore, IRIS provides a tailorable, visual depiction of the database structure in an attempt to bring system interaction closer to the user's cognitive processes.

ENTITY-RELATIONSHIP DATABASES

In the E-R data model [8], data are organized in terms of *entity types* and *relationship types*. All entities belonging to an entity type have the same *attributes*. For example, an **EMPLOYEE** entity type might contain attributes for **Name**, **EmpID**, and **Salary**. In the IRIS version of the E-R model, each relationship type connects two entity types (called *source* and *target*). For each relationship type, the schema specifies whether the relationship is one-to-one, one-to-many, many-to-one, or many-to-many. For example, the **PROJECT_MEMBER** relationship type might be a one-to-many relationship whose source is the **PROJECT** entity type and whose target is the **EMPLOYEE** entity type. *Entity instances* and *relationship instances* (or *occurrences*) contain values, e.g. **EmpID**, = 456, that conform to the specifications of the entity types and relationship types in the schema. For example, a **PROJECT_MEMBER** relationship instance might exist between a **PROJECT** instance with **ProjCode** = 123 and an **EMPLOYEE** instance with **EmpID** = 456. In this paper, when referring to entity or relationship instances, the qualifier 'instances' is often omitted (e.g. en-

ISSN# 0-7803-0233-8/91 $1.00©1991 IEEE

tities); when referring to types, the qualifier 'types' is always explicitly specified.

The structure of the E-R model provides a natural representation for the schema as a graph. Nodes (rectangles) represent entity types and edges (lines) represent relationship types. This graphical representation of the schema is used as a focal point for interacting with the system in IRIS.

WHAT IS AERIAL?

AERIAL is a facility for an inexpert user to easily explore and update the contents of an E-R database in an unplanned manner by navigating from any entity to related entities. The concepts embodied in AERIAL can be extended directly for other types of databases [13]. For relational databases, for example, entity type translates to table, entity to tuple, and relationship to "meaningful, pre-specified joins" [5].

AERIAL is implemented on a windowing system using a visual display terminal (C and Windows under DOS). The user browses by operating directly on a graph representing the database schema and by selecting action options from data windows.

BROWSING VERSUS QUERY

If query is likened to shopping with a list -- with brand names, i.e. with detailed specifications about what to buy, then browsing corresponds to walking the aisles to learn what is available and where things are located. If you go to a new store, some browsing is necessary before you can shop from a list. Sometimes, even if you are familiar with the store but want to buy an item you've never purchased before, you do some browsing (e.g. I want to buy honey and assume it's with the jams and jellies but later discover, by browsing, that it's with the baking supplies). Browsing is slower than query but has greater possibility of serendipitous learning. Not only can it be used to acquire the information you need (e.g. baking supplies are on aisle 6) but it can also bring up other interesting and useful pieces of information that may be available (e.g. you also discover that birthday candles are with baking supplies).

In the context of human development and learning, Lifter and Bloom [12] describe how the ability to "take apart" precedes the ability to "put together." For example, a child is able to remove rings from a peg before he is able to coordinate placement of rings onto a peg. Likewise, we view Browsing as "taking apart," i.e. you start with the overall schema and slowly explore into it, navigating from piece to piece; Query is a process of "building up," i.e. you start with an understanding of how the database is constructed and formulate a query for retrieving data from it. Sometimes, users who are proficient at query formulation can see little difference between query and browsing but novice users, who are struggling to learn database concepts, structure, content, and query language, find it extremely valuable.

We illustrate the use of browsing in a learning situation in the scenario below. Before we do that, we discuss concepts of mental models so that the cognitive development that takes place during browsing may be better understood.

MENTAL MODELS

We postulate that the user starts with an organized body of knowledge about a domain; this is often referred to as a mental model [11]. Several types of mental constructs have been proposed to explain different types of knowledge. The constructs most appropriate for our work seem to be mental schemas [11]. Mental schemas consist of a set of concepts about a domain. Concepts are formed by aggregating propositions. For example, [2] discusses how several propositions related to "house" can be used to form the concept of a house. Further, the concept, which defines a class, has slots which take on values. Instances of the concept may or may not have values for all the slots.

The value for a slot may be taken from the domain of primitive datatypes (numbers, text) e.g. "Number-of-rooms = 5" or it may refer to another concept such as "Dwelling-for = People." Further, slots may take multiple vales such as "Dwelling-for = People and Dogs." In the E-R model, relationships can be considered as attributes that take single or multiple entities as values (to-1 and to-M relationships respectively). Thus, concepts and slots translate quite well into entities and attributes with relationships being treated as special kinds of slots.

Because mental schemas have *isa* relationships, they correspond closely to data models that integrate object-oriented and E-R concepts e.g. [17]. There are, however, two important differences between mental schemas and data models. One is that slots can take fuzzy values such as "near" and "good." The other difference is that

the distinction between types and instances is much less clear cut in mental schemas and data and metadata can be freely intermixed.

Approaching a task, the user has slots in his mental schema that do not have values or have fuzzy values that need to be made more precise. He works with a database to provide values for these slots. Slot values can either be filled directly by declarative knowledge contained in the database or by building up concepts from the declarative knowledge in the database.

To be able to use the database, the user has to set up a correspondence between the concepts in his mental model and the types in the database. The difference between his mental model and the model embodied in the database (or other tool) contributes to what has been termed the *gulf of execution* [14]. In the process of setting up this correspondence the user may extend or modify his mental model. This altering of existing cognitive structures in the subject (e.g. user) to match new, external stimulus objects (e.g. database) was termed "accommodation" by Piaget [15]. Carroll [7] relates these human developmental cognitive processes to human-computer interaction when he states that, "Computer interfaces and accompanying materials can be deliberately cast to stimulate direct comparisons between the current situation (the system itself so to speak) and whatever prior knowledge is engaged by the current situation, thereby highlighting key similarities and differences. These comparisons must be engineered to stimulate inferential processing, hypothesis testing, and active learning." We postulate that the facilities provided by AERIAL, especially the ability to move easily from data item to related data item, facilitates such learning.

In the following section, we present a scenario which illustrates AERIAL's direct manipulation features and the associated cognitive actions and mental schema modifications that may occur during a database browsing session. The italicized sections represent the mental schema actions and can be read sequentially to grasp the cognitive emphasis. The standard typeface sections (with the exception of the first and last paragraphs) represent database actions and the how-to's of AERIAL and may also be read sequentially. Of course, reading the sections alternately provides an integrated picture. The first and last paragraphs provide an introduction and conclusion to the scenario and, for completeness, should be included in both readings.

SCENARIO

You have just been hired by Malibu Foods as marketing manager for their Peanut Butter line. The specific problem you have been asked to address is their declining share of the Peanut Butter market. To help you understand the problem, Malibu Foods provide you access to their product database. There is, however, little computer assistance they can offer you. The previous marketing manager left recently and took all the computer talent with him. You have been briefed on basic E-R concepts and their relation to AERIAL schema diagrams.

You turn on the machine and bring up the product database. The diagram shown in Fig. 1 appears on your screen.

The user starts an AERIAL session by bringing up the schema graph (Fig. 1). Nodes in the schema represent entity types and arcs represent relationships between them. The Connect action allows the user to select a database to work with. The View option allows him to change the display by adding arrows, suppressing relationship names, etc. Zoom allows him to select a portion of the display for magnification. Tailor allows the locations of the nodes to be changed while SchemaDef provides facilities to add entity and relationship types to the database. The query option provides facilities to specify queries using a graphical query language. This is discussed in more detail in Sockut et al. [18].

The user starts a AERIAL session by selecting "Browse" from the action bar and then selecting an entity type from the schema graph. After this, a current set of entities is always identified. This can be the set of:

- All entities of a type (e.g. all instances of **Market**). This is specified by clicking on the entity type rectangle in the schema graph.
- Selected entities of a type, i.e. for which a predicate defined on their properties evaluates TRUE (e.g. all instances of **Market** which are in the Northeast Region -- "Region = 'Northeast' ".)
  Such a set is specified by first selecting all instances of a type as above and then bringing up a window, showing the attributes of that type, into which selection criteria can be entered.

After selecting a current set of entities the user can choose to display the entities in the current set in a data window as follows:

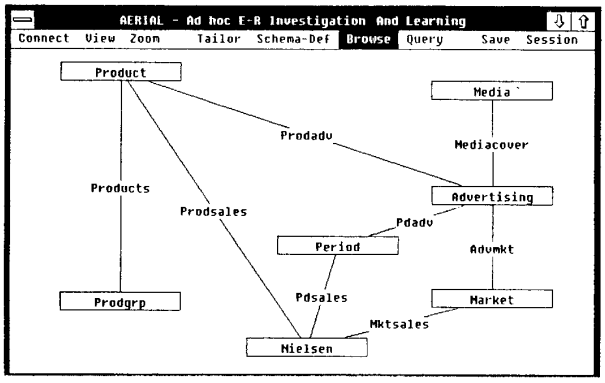

Fig. 1. Schema diagram for Malibu Foods database

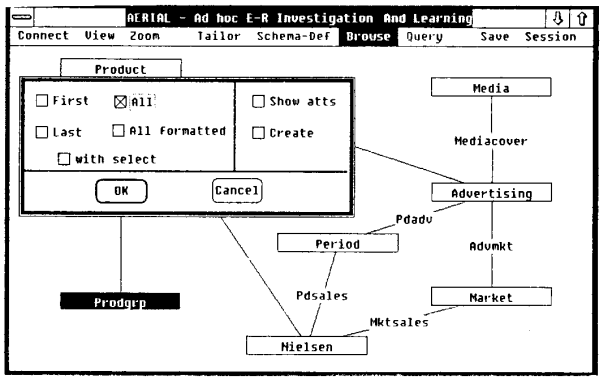

Fig. 2. Malibu Foods schema diagram with **Prodgrp** selected and Browse Options Dialog Box

- One entity at a time in a window -- this is the current entity. A single entity window is, essentially, a list of attribute names and values. The values of attributes are displayed directly as numbers and character strings and can be overtyped for updating. The key attribute for the entity is indicated by a '>' to the left is its name. This type of window will be referred to as a "single entity display."
- As a table in a window. As the table may be very large, the table window displays only a few entities at a time and can be scrolled to display other entities. At any time the user can select a row in the table, double-click to make the selected row/entity the current entity and bring up a single entity display window for it. In the table window, each lines shows the attributes for one entity in rows. The window has action buttons for scrolling up and down and left and right, navigating relationships from the selected entity, closing the browsing session, etc. This window will be referred to as a "table display."

You decide to look at ***Prodgrp*** *because you expect Peanut Butter to be a product group; you are manager for the Peanut Butter line so you assume it must consist of several Peanut Butter products, in other words, a group of Peanut Butter products. You select "Browse" from the action bar and select the* ***Prodgrp*** *entity type with the mouse. The browse options dialog box appears with some options on it, as shown in Fig. 2.*

The browse options dialog box, Fig. 2, appears when an entity type is selected from the schema graph. It allows you to browse the selected entity type either one entity at a time starting at the beginning or the end ("First", "Last" respectively) or by looking at a table of the current set ("All"). In either case, you can specify selection criteria to determine the entities that will appear ("with Select"). This box also allows you to look at the attribute definitions ("Show atts") for the entity type or create an instance of the type ("Create"). Fig. 3 shows the resulting table display after selecting "All" from the browse options dialog box. If the current entity set is displayed as a table, the user can:

- Select a displayed element as the current entity; this is accomplished by double-clicking on the element. The result is a single entity display window with the new current entity.
- Scroll the table to expose other data elements in the set.

You select "All" to look at all ***Prodgrp.*** *This seems most reasonable as you don't really know how many product groups there might be and you don't want to step through them one at a time. Looking through the table that appears you find entries such as 'Cookies', 'Munchies', and 'Sandwich spreads' (Fig. 3).*

"Peanut Butter must be a product then," you say to yourself. "Probably in the 'Sandwich spreads' product group." You double click on the line showing the 'Sandwich spreads' product group to bring up a single entity window for the 'Sandwich spreads' product group (Fig. 4).

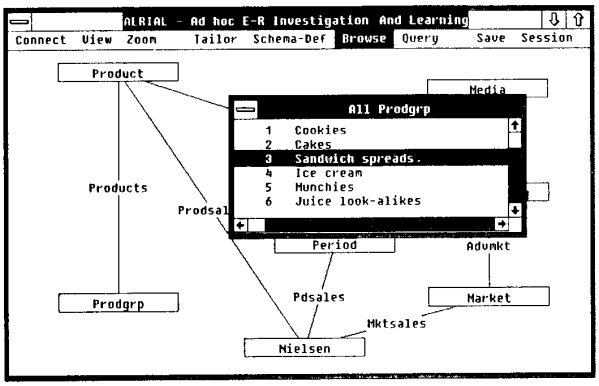

Fig. 3. Malibu Foods schema with **Prodgrp** Table

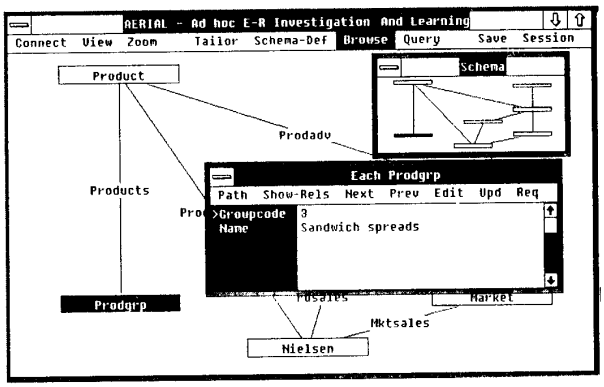

Fig. 4. Malibu Foods schema and Single Entity Display for 'Sandwich Spreads'

In Fig. 4, note that **Prodgrp** is the current entity set and the 'Sandwich spreads' **Prodgrp** is the current entity.

You notice that, in addition to the single entity display window, a window has appeared showing a smaller version of the overall schema window and seems to be mimicking what is on the background schema window. Likewise, the background color of the attribute names on the single entity display matches the color of the highlighted entity type rectangles on the schema window.

The path to the current entity is highlighted on both the background schema window and the miniature schema window; because the background window may become obscured by overlying instance windows and table windows, the miniature schema window is displayed for the purpose of providing a path overview. Corresponding single instance display windows along the path are highlighted in the same color. This makes it easy for the user to see how he arrived at the current entity instance. We describe this in more detail in the section entitled "The Path Window."

The single entity display window has action buttons for scrolling the attributes, if necessary, displaying the next or previous entity in the set, navigating relationships from this entity, closing the browsing session, etc.

From a single entity display window, the user can:

- change the attribute values of the current entity by overtyping the displayed values and asking for the values to be updated in the database. An undo function is also provided.
- display the next or previous entity in the current set and make it the current entity. The new current entity is displayed in the same window overlaying the previously current entity of that set.
- navigate a relationship from the current entity and display all or selected entities that are members of that relationship either an entity at a time or as an entity table (Fig. 5).

At any time the user may:

- select a new current entity set from the schema diagram and display its first or last element in a single entity display window or all of its elements in a table window.
- make a previously displayed entity or table window the current entity or current set.

*You guess that following the **Products** relationship will show you all the products under 'Sandwich spreads'. This makes sense according to the schema graph as well because the relationship **Products** has the entity type **Product** as its target (source of relationship is **Prodgrp**). You select the "Show-Rels" option on this window's action bar and find that the window expands to show the one relationship you can follow. You select the relationship and then choose "Nav-Rel" (Navigate Relationship) from the "Rels" pulldown on the action bar (Fig. 5).*

Selecting the "Show-Rels" option on the action bar of an entity instance window expands the window to include a display of the relationships in which it participates. Note that this corresponds to schema graph lines

into or out of the entity type of the current entity instance. The display of these relationships is broken into two categories and displayed in two separated areas under the attribute values of the entity instance. The upper box describes the "to-1" relationships and the lower box the "to-M" relationships. In Fig. 5, the relationship **Products** is between Many **Products** and a single (1) **Prodgrp,** it appears in the lower box. Selecting the relationship and then choosing "Nav-Rel" (Navigate Relationship) from the "Rels" pulldown on the action bar will cause the browse options dialog box to appear from which you may choose the "instance-at-a-time" or "all-instances-of-a-set" browsing options, i.e. "First/Last," or "All." Selection constraints may also be specified.

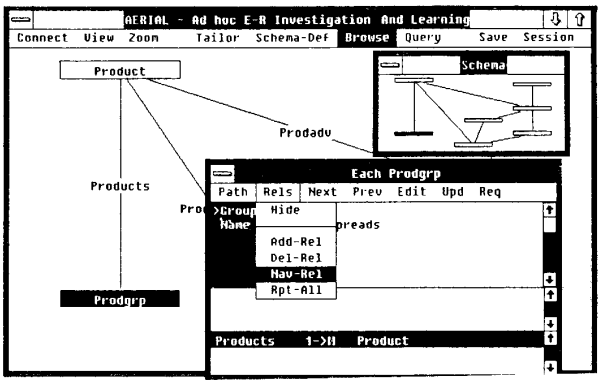

Fig. 5. Navigating the **Products** Relationship

Since you have selected a 1-to-M relationship, i.e. there are many products for each product group, the browse options dialog box appears again. You select "First" and the first product in the "Spreads" product group appears. This turns out to be 'Smooth PB, 8oz.' (Fig. 6).

By selecting "Next" on the product instance window you find other specific descriptions of products, styles, sizes, and packages. You notice too that the miniature schema window is now displaying the path you have followed, i.e. from an entity of type ***Prodgrp*** *across the relationship* ***Products*** *to an entity of type* ***Product.***

Fig. 6 shows the result of navigating from a **Prodgrp** across the relationship **Products** to a **Product.**

At this point you have discovered something important and interesting. You are responsible for neither a product group, nor a product. Instead, you are responsible for a collection of products.

Let us step back for a minute and consider what you have learned so far. Your briefing explained that the schema diagram shows a graph of entity types connected by relationship types. The notion of an entity type is well understood. Some of the entity types represented in the schema, ***Product, Media, Period,...*** *are well named and correspond in the main to familiar concepts. You are not quite sure of what Nielsen represents but have a hunch it may be sales data.*

Relationship type is a more difficult concept but you now understand that the ***Products*** *relationship between Product Group* ***(Prodgrp)*** *and Product* ***(Product)*** *ties together the products that belong to a Product Group.*

In what you have learned so far, the database model corresponds quite well to your existing concepts. The database construct of entity type corresponds quite well with concept as a class except that the database concept is more precise and means a collection of instances all of which have the same attributes. The database construct of relationship type did not have a precise domain concept correlate in your mind. The products relationship type you explored turned out to correspond to the concept called inclusion. Other relationship types that you see on the screen, such as from ***Product*** *to* ***Nielsen*** *and* ***Market*** *to* ***Nielsen,*** *seem to be quite different. For the moment, you decide that relationship type seems to be a portmanteau construct that database people use to represent several different types of real-world relationships. Each type must, therefore, be understood separately, i.e. associated with possibly different domain concepts. You have formed a fuzzy concept and you hope it will clarify later.*

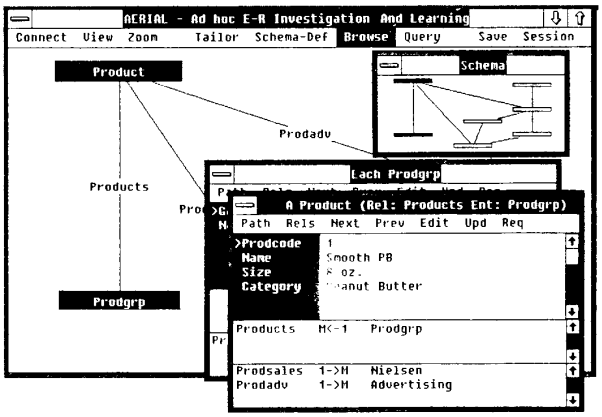

Fig. 6. Browsing **Product** -- Smooth Peanut Butter, 8oz.

You have also acquired skills in exploring the database. You know how to browse instances of an entity type and how to follow relationships. Fearlessly, you plunge on. The domain concept of product has a slot for sales. You need to find values for this slot but you also need to understand how the data are stored. You decide to browse across the ***Prodsales*** relationship (from the entity instance 'Smooth PB') to the entity ***Nielsen*** as you suspect it has sales data in it. Fig. 7 shows the result of navigating this relationship. You find it does have unit and dollar sales and year to date figures but it's not obvious what the time period is. You also find that some of the data are aggregated by category. Looking through a few Nielsen entities you see they have an attribute "category" with a value 'Peanut Butter' (Fig. 7).

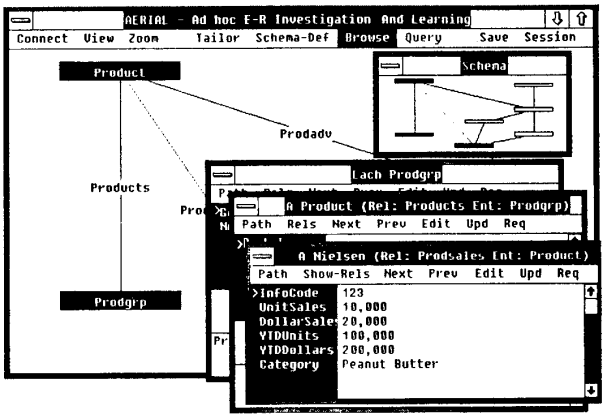

Fig. 7. Nielsen instance showing category Peanut Butter

Fig. 7 shows the result of navigating the **Prodsales** relationship of **Product** ("Smooth PB") to one of its **Nielsen** instances.

You now realize that you are responsible for a 'category' of products. The database aggregation construct 'category' corresponds to your domain concept 'product'. This changes your mental model somewhat, i.e. you have accommodated your mental model to the model in the database. You also now know that sales figures are available by product and by category, only the time period is not clear yet. Thus, the domain concept of sales has been made more precise although the value of the time period slot has not been filled.

You decide to start a new path from ***Period*** *through* ***Nielsen*** *and then to products in the 'Peanut Butter' category. This alternative route seems more preferable to attempting to look through all Periods related to the current Nielsen instance (e.g. there may be Periods related back over many months and many years). You start a new path by returning to the main schema diagram and selecting* ***Period;*** *you discover that periods are months. The sales data are monthly, then, with year-to-date totals. By moving forward among the months you quickly find the current month. "Let's see if they have data for this month," you say and navigate over the* ***PeriodSales*** *relationship to* ***Nielsen.*** *Sure enough, you find sales data for the current month. The numbers are small, as you expect. By going back to* ***Period,*** *getting the entity for the previous month and doing the identical navigation to* ***Nielsen*** *you find that the numbers for this month are roughly a quarter of the number of last month. "These must be the first week's numbers," you say. But since it's early in the second week you can't tell if the data are updated weekly or daily, as done by Famous Potato Chip.*

The "when-updated" slot for sales now has the value 'daily or weekly'.

You have now found that the sales data are monthly and updated either weekly or possibly daily. By browsing the Nielsen figures you have also determined that the sales figures range from 10,000 to 100,000 cases a month. You also find that the larger numbers seem to occur in the Northeast. This is a fuzzy concept that can be validated and refined by writing appropriate queries.

In the same way, you browse the entity type ***Market*** *and discover that it contains individual cities and regional aggregations including the entire country. At this point you know enough to write a query. You decide that the information you need for the sales slot of the domain concept "Product = Peanut Butter" can be acquired by writing a query to extract data from the Nielsen entity type for the aggregation category 'Peanut Butter' by month and/or year to date for selected markets or market aggregations. You use the Graphical Query Language (GRAQULA) [18] to write a query to show the sales of the Peanut Butter category over the last two years for the entire US. As expected, the sales indicate a decline especially in the last six months. "Most people," you reason, "eat Peanut Butter with Jelly. Did Jelly sales also decline?" In a few seconds you change the query to show Jelly sales and find that Jelly sales, while not spectacular did hold their own and not decline.*

"Let's look at advertising," you say, "perhaps that had something to do with this." You start by browsing ***media***

and find 'Billboard', 'Newspaper', 'Magazine', and 'TV'. "Let's look at TV," you say; "that's usually the most important." You follow the ***Mediacover*** *relationship from 'TV' instance of* ***Media*** *and browse the* ***Advertising*** *entity type. Here you find the amount of TV exposure in each market by time slot. ...*

This scenario shows how AERIAL's browsing capabilities can be used to understand the model of information stored in the database and relate the entity types and relationships to the user's mental model. Information that the user needs to know to do his job can now be acquired by writing queries using the corresponding database types. The scenario also shows how fuzzy concepts can be formed by browsing and later refined by queries.

THE PATH WINDOW

The objective of a browsing facility is to make it simple for the user to navigate from entity or entity set to related entity or entity set to reach and display the information that he desires. The flip side of this ease-of-use is that he can rapidly fill up his screen with windows covering other windows, all of which are related in complex ways to each other. In such a situation it is easy to lose track of how a particular data window was reached, i.e. the path that was followed.

AERIAL provides a facility for displaying the path followed via an overview window that shows the schema graph in miniature. Essentially, this is the schema graph reduced and stripped of text and cardinality information on the relationships. This window cannot be overlaid and shows the path to the current entity type by emphasizing the entity types and relationships in the path. AERIAL uses colors on the entity types and relationships to do this although other emphasis techniques such as different box borders and line types are possible.

Note that the overview window shows only the current path, i.e. the navigational path to the current instance window. If the user selects another, previously displayed, window and makes it the current path, the overview window will change the path display to show the path to the new current window.

Any window on the path can be made the current window by clicking on it to make it the active window. The user can then continue navigation from this window. Alternatively, the action bar item, "Path," can be used to ask for specifics about the path followed to arrive at

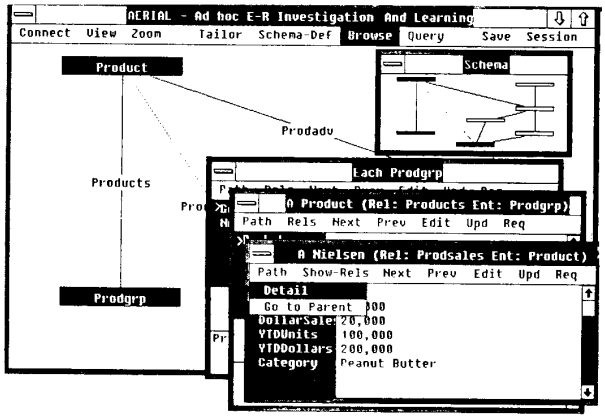

Fig. 8. Using the "Path" pulldown

the given instance. Fig. 8 shows the first path from the scenario, **Prodgrp** across **Products** to **Product;** then across **Prodsales** to **Nielsen.** Choosing "Detail" from the "Path" pulldown (Fig. 8) brings up a new window showing the detail, i.e. giving its ancestors' entity type name and key value as well as the relationship across which navigation occurred. The user can select any item from this list and cause the instance window of that item to be surfaced and the associated instance to become the current entity.

RELATIONSHIP TO EARLIER WORK

ISIS [10] is an early browsing system that presented information about a semantic database in visual form. As such, it initiated a lot of the thinking on browsing and visual presentation that led to AERIAL and other browsing systems. Due to the hardware and presentation facilities available at that time, however, its visual displays now appear primitive.

One of our goals in writing AERIAL was to allow several entity instances and sets to be displayed at the same time with further navigation possible from any of them. This allows the user to pursue multiple threads of exploration. Earlier browsers do not support this. Typically they allow one instance to be displayed at a time [9, 10, 16].

OdeView[1] is a graphical browser for the Object model. It provides the interesting capability of allowing the user to be able to customize the manner in which an object is displayed; which is particularly important with encapsulated objects. It, however, also displays an in-

stance at a time although the instance can be a complex object.

If several instances are to be displayed simultaneously, the individual instance display must be compact. In AERIAL the default instance display is a specially constructed window that shows the names and values of the first five attributes.

AERIAL does not present information from related entities in the instance display. Rogers [16] shows, for example, the title attributes of the related publication entities in an employee instance display. We feel that this is poor utilization of space because it attempts to guess at the user's intentions, i.e. it assumes that you want to navigate publications and is assumes that you want to look at their titles. Further, this makes navigation on any one related instance awkward.

SUMMARY

This paper discusses AERIAL which provides a set of facilities for exploring a database in an unplanned manner. We argue that these facilities, collectively called Browsing, are especially important for the neophyte user and assist him in understanding the content and structure of the database and building a correspondence between his mental model of the problem domain and the model embodied in the database.

In summary, AERIAL provides the following facilities:

- The user can display individual entities and can move from the current entity to all or selected entities that are related to it easily and conveniently.
- Data is presented in multiple windows that can be independently moved, scrolled and sized. This allows the user to retain multiple related data elements on the display. Further, the user can go to any displayed instance and start new navigation paths.
- The overview window describes the path followed to arrive at the current entity.

The AERIAL browsing facility described above is independent of the data model. Thus, it can be used to present a uniform interface to several different data models: hierarchical, entity-relationship, semantic, and relational.

REFERENCES

[1] Agrawal, R., Gehani, N.H. and Srinivasan, J., "OdeView: The Graphical Interface to Ode," In *Proc. SIGMOD, 1990*, pp. 34-43

[2] Anderson, J.R., Cognitive Psychology and Its Implications, Second Edition, W.H Freeman and Co., 1980.

[3] Burns, L.M., Malhotra, A. and Pazel, D.P. "BROWSER: A Visual, Interactive Database Interface," *Research Report RC 10964*, IBM T. J. Watson Research Center, Yorktown Heights, N.Y. 10598, January 1985.

[4] Burns, L.M., Archibald, J.L. and Malhotra, A., "A Graphical Entity-Relationship Database Browser," In *Proc. Hawaii International Conference on System Sciences 1988, Vol. II*, pp. 694-704, 1988.

[5] Burns L., Malhotra A. and Shimmin E., "An Interactive Diagram to Visually Represent Tables of Related Data and Meaningful Joins Between Tables," *IBM Invention Disclosure UK8-89-108*, 1990.

[6] Burns L., Malhotra A. and Black J.B., "Is a Picture is Worth a Thousand Queries?" *RC 16172, October, 1990.* IBM T. J. Watson Research Center, Yorktown Heights, N.Y. 10598, August 1990.

[7] Carroll, J.M. and Rosson, M.B., "Paradox of the Active User," in Interfacing Thought: Cognitive Aspects of Human Computer Interaction, Carroll, J.M. (Ed.), MIT Press: Cambridge, Massachusetts, 1987.

[8] Chen, P. P-S., "The Entity-Relationship Model - Toward a Unified View of Data," *ACM TODS.*, Vol. 1, No. 1, pp.9-36, March, 1976.

[9] Fogg, D., "Lessons from a "Living In a Database" Graphical Query Interface," In *Proc. SIGMOD 1984*, pp.100-106.

[10] Goldman K.J., Goldman S., Kanellakis, P. and Zdonik S.B., "ISIS: Interface for a Semantic Information System," In *Proc. SIGMOD 1985*, pp.328-342.

[11] Johnson-Laird, P.N., Mental Models, Harvard University Press: Cambridge, Massachusetts, 1983.

[12] Lifter, K. and Bloom, L., "Object Knowledge and the Emergence of Language," *Infant Behavior and Development* 12, pp. 395-423, December, 1989.

[13] Malhotra, A., Burns, L.M., Sockut, G.H. and Whang, K.-Y., "IRIS: An Interactive Database Facility," *RC 16945, June 1991*, IBM T. J. Watson Research Center, Yorktown Heights, N.Y. 10598.

[14] Norman, D.A., "Cognitive Engineering," Chapter 3 in User Centered System Design, Edited by Norman, D.A. and Draper, S.W.

[15] Piaget, J., The Construction of Reality in the Child, New York, Basic Books, (1954).

[16] Rogers, T.R. and Cattell, R.G.G., "Entity-Relationship Database User Interfaces," In *Proc. 6th Intl. Conf. on Entity-Relationship Approach*, pp.323-336, 1987.

[17] Rumbaugh, J., "Relations as Semantic Constraints in an Object-Oriented Language," *Proceedings OOPSLA, 1987*, Orlando, Florida.

[18] Sockut, G.H., Burns, L.M., Malhotra, A. and Whang, K.-Y., "GRAQULA: A Graphical Query Language for Entity-Relationship or Relational Databases," *RC 16877, March, 1991.* IBM T. J. Watson Research Center, Yorktown Heights, N.Y. 10598.