THE INDEX IS KEY

"...words are not...carriers of complete meanings, but are instead more like index terms or cues that a speaker uses to induce a listener to extract shared memories and knowledge. The degree of detail and number of units needed to express the speaker's knowledge and intent and the hearer's understanding are vastly greater than the number of words used to communicate."


Long before computers learn to understand text, they can do a passable job of determining what a given text passage is about. Of course, explaining what it's about in few words is also a delicate matter, as noted above: What words should you use? And what words will the computer use?

There's an answer with a name familiar from databases -- "query by example." It's easier to show than to describe. Right now a growing group of companies is working on this field of so-called "similarity searching," where you show the computer a piece of text and ask it to find matches. Soon, others will be working on its logical extension into text applications -- not just systems that retrieve text by classification, but systems that can do something based on such classifications (see page 8). Profiles of two leaders in similarity searching follow.

The old way

But first, consider the widely used standard Boolean search, where a system finds all documents that contain a specified word. Doing a search on a single word is quick (and generally produces far too many hits); doing one on several requires combining the results of the searches and producing the Boolean intersection (and, with both A and B) or combination (or, with A or B or both) of those results, or more ornate combinations (with A but without B, and so forth). These searches are straightforward: With small target files, everything can be held in memory; for larger systems, disks must be searched. If there's time to index the texts beforehand, performance can be improved dramatically, because the index has a known sequence, and points to the locations of the desired words.
These indexes also list the locations within the documents, so that you can specify "A within five words of B." The closeness query looks for all instances of one word, then of the other, and checks for proximity based on their locations within a document. Theoretically you could use this approach to measure similarity as well, but the explosion of combinations renders that impractical.

Quality of document retrieval is measured by two coefficients: Of 100 documents that you want, how many do you get? That first coefficient is 0 at worst and 100 percent at best -- 30 is poor but typical for Boolean searches on commercial systems. And how many do you get that you don't want? The second coefficient is 0 at best and the whole database at worst. Unfortunately, all a Boolean search gives you is a list of hits, with no ranking. But the world is not quite so black and white; some documents are more relevant than others, or more similar to an ideal (the "model"). Now read on...

THINKING MACHINES

Dow Jones has just plunked down $5 million for two Connection Machines from Thinking Machines Corp. of Cambridge, MA. Dow Jones' aim is to make it easier for its customers to use its on-line text services, a fast-growing part of its fast-growing News Retrieval business. There are three things to improve: the interface, which all agree is clumsy, confusing and rigid; the range of data available, which DJ is broadening by making deals with third-party suppliers; and the quality of retrieval.

The main goal addressed by the company's use of Connection Machines is speed and accuracy of retrieval (DJ is handling the others itself, although allowing similarity searching certainly improves the interface by avoiding the need for Boolean formulations). By using the Thinking Machines system, familiarly called "Finder," Dow Jones is basically applying brute force to the problem of text retrieval, with a parallel architecture uniquely suited to performing the same function on 65,536 ($2^{16}$) processors and data sets. Dow Jones is the only announced customer Thinking Machines has in this field, although we imagine there are several more, including the usual suspect near Washington.

Each processor in the Finder's Connection Machine has its own 8K of memory (upgradable to 32K with the advent of 1-megabit chips), which can store approximately 1800 different words in compressed form ("surrogate code"; see page 5) -- or enough to represent about a 7000-word chunk of text stripped of stop words such as a, an, the, etc. The model document -- the item that you want something similar to -- is likewise compressed (on the fly if necessary) and is then compared to each data cell held by the Finder.

While most systems index text as a list of words with pointers to the occurrence of the words, Finder simply compresses the text, holds it all in memory, and goes through it all each time a search is needed, with a performance 80 times that of a 10-MIPS Sun 4 (although not strictly comparable). New documents are added to the text base discretely, whereas in index-based systems the entire index must be updated each time a document is added.

Typically, the user will start out with a short query. Experienced users tend to enter a Boolean query, while novices may enter a question. Since
MEASURES OF SIMILARITY

What makes one document similar to another? Is it the color of his eyes, her hair? The measures vary, but there are a number of accepted formulas, most of them relying on word weight without regard to proximity. Words in certain locations (titles, for example) could count double, or constitute another dimension, at a user's option. Generally, stop words (a, an, the, etc.) are discarded. Also, synonymous words or phrases representing similar concepts could be combined into a single dimension.

Now, think of each "word" as a dimension. Each document is then represented by a vector (list) exactly as long as the number of discrete words (dimensions) in the text base. If the document doesn't include the word, the vector has a zero for that dimension; if it does, it has either a 1 (for a binary measure) or a value for the weight of that word in the document.

That's each document. Now, how do you measure the similarity of two such vectors? Imagine a vector with just two dimensions -- two words -- to understand the concept, and then imagine a vector with hundreds or thousands of dimensions -- one for each word, phrase, or other item. The simplest measures generally involve multiplying the vectors: If both A and B have a value in a given dimension, the product for that dimension will be 1, or some larger number in a non-binary system. The sum of the products represents the strength of the match; dividing it by some figure representing the length of each vector/document normalizes it.

Consider the examples below. A is all about Juan; B is all about Alice. C is about both of them, and about tigers, equally. D has as much about Alice as does C, but it's mostly about Juan -- and tells more about him than A does. By these measures, A and B are totally dissimilar, while both are somewhat like C (a vector product of 6 in both cases). But C and D are far more similar because they both cover all three topics (with a vector product of 26). Yet if you account for intensity, A and D are more similar, both focusing on Juan, and with a vector product of 30. Is D more similar to B than C is? Yes, if you go by raw numbers; perhaps not if you divide by a factor that accounts for D's greater length (it has more about Alice than C, but doesn't focus on Alice). What proportion of the text is similar? And there's inclusion: Does A match B closely, or does it cover B and a lot else besides? We won't go into the math further here, but you can see how there are different kinds of similarity, and different ways of adjusting for the lengths of the documents containing the information. Those who want the details can consult "An introduction to modern information retrieval," the industry-standard text (see Resources, page 19).

<table>
<thead>
<tr>
<th>Documents</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>A*B</th>
<th>A*C</th>
<th>A*D</th>
<th>B*C</th>
<th>B*D</th>
<th>C*D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Juan</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>10</td>
<td>0</td>
<td>6</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>Alice</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>Tigers</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Sum</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>14</td>
<td>0</td>
<td>6</td>
<td>30</td>
<td>6</td>
<td>9</td>
<td>28</td>
</tr>
</tbody>
</table>

A*B is the product of A and B for a given dimension (topic).

Release 1.0

31 January 1988
Finder simply matches key words, it doesn't really matter: Both kinds of users get more or less what they want -- a list of article titles. From that list the user can mark the ones that look promising. All the words in those selections are sent through Finder's sieve, and chunks with matches are selected, sorted and presented back to the user in a ranked list of documents. Finder doesn't perform "true" Boolean searches because it may find something that's not there (if an overlap of several sets of 10 bits happens to correspond to the word being searched for). In practice, a front-end system that highlights the specified word can notice if it's absent, and discard the spuriously selected text.

At the user's option, any selected article can be displayed, at which point the actual text is downloaded (more slowly) from an auxiliary database where the full text is stored. Dave Waltz, senior scientist and director of advanced information systems at Thinking Machines, notes that Finder is precisely optimized for its current range of database size and search style. With larger databases, you could increase the text per processor, but you'd eventually tax the capabilities of each processor. Alternatively, you could move to inverted lists (cf. Third Eye, below) and get greater speed on more text without increasing either memory or I/O requirements.

Strategic importance to DJ

Dow Jones evp Bill Dunn is jumping for joy at the prospect of unleashing his Connection Machine on Dow Jones' massive text bases. While DJ's own people are cleaning up News Retrieval's notoriously cryptic front-end, Finder will give it power on the back-end to be far more responsive as well as friendly to customers' requests. The system should be able to handle 40 active user sessions ("40 people hitting buttons," says Dunn) at a time. That number will multiply as front-end capacity is increased, because the current set-up doesn't yet fully exploit the the Connection Machines' power. Moreover, these aren't simple Boolean searches but richer, more powerful query-by-example searches.

Initially, DJ's Connection Machines (one for use and one for development and back-up) will have 32,768 processors and can each hold 256 megabytes, equivalent to 1 gigabyte of raw text. The initial DJ database will consist of several recent months of material from sources such as Dow Jones' own Wall Street Journal, Barron's, the Dow Jones wire and American Demographics; as well as selections from Forbes, Fortune, Financial World, Money, the PR News-wire, and 140 regional papers. In time, Dunn's goal is to add a substantial number of sources and to extend the time periods. (Dow Jones is also an early customer for TM's DataVault, five (upgradable to 20) gigabytes for backup or eventually for fast swapping of auxiliary databases. Some material could be stored in the DataVault and swapped in and out of a reserved sector of the Connection Machine with a few seconds' delay.)

The Dow Jones version of Finder will run considerably faster than the prototype Thinking Machines shows visitors, replacing the Symbolics with a VAX-PC combination, although it will lose some of the flavor of the Symbolics version, especially the ability to select text with a mouse. It's difficult to predict the impact the system's greater power and ease of use will have on people's usage patterns, and Dunn and crew haven't yet figured out the pricing. Long-term, Dunn would like to store more data (from more third parties) at customers' sites, with the Connection Machines as backup. The possibilities are intriguing.

Release 1.0

31 January 1988
How Thinking Machines "surrogate-codes" its text

The key to Finder's performance is partly in the raw power of the Connection Machine, and partly in the clever way it prepares the text it searches. Finder divides all the documents up into chunks of about 50 words each, and assigns 64 chunks to each of its 65,536 processors. As the text is "coded," it is compared against a system dictionary that discards stop words, checks for phrases, and performs other tasks. Optionally, at this stage other information such as capitalization or location can be noted. Each text chunk is represented not as words, but as a 1000-bit string. The bits in the string default to 0 and are set to 1 as follows: Each unique word in the database (say 200,000) is represented as a random but consistent set of 10 bits along a 1000-bit vector (or one of $2^{13}$ possible combinations). Finder turns on the appropriate bits in the vector of any paragraph where each word is present. Thus the 30 unique words (minus the, and, etc.) in a paragraph will affect 300 bits, of which probably 200 will be unique. The diagram below shows an example of a vector representing two words:

Juan

Alice

The 4-bit codes for Juan and Alice have one bit in common, circled in the illustration at left.

Statistically, it turns out, this system is effective at determining the presence of a given word. The bits for some words may overlap slightly, but spurious hits are uncommon, and negligible when you check for similarity of 30 words or more. (The only place spurious hits are noticeable is when you try to do a Boolean query on just a couple of words -- and you won't miss anything that contains the word sought.)

When a match is requested, Finder sends the requested words, similarly rendered as bit patterns, to each of the 65,536 processors, each of which tests it against its 64 1000-bit strings representing 64 text chunks (or a text base of 208 million words -- less for DJ). Each word that hits is multiplied by a score representing the weight of the word in the model query, and that score is allocated to the appropriate chunk of text among the processor's 64 chunks. At the end, scores are totted up, and combined to rank the documents they comprise. A document's score is the maximum score of the chunks it comprises, increased slightly if it contains several high-scoring chunks. Other weighting schemes are selectable by users or OEMs.

Although Finder does not do proximity searches per se, it is sensitive to proximity and to frequency of words in a statistical way based on its way of combining text chunks into documents. As currently designed, Finder merely notes the presence of a word in text chunk, but a document that contains several chunks will get a higher ranking if a given word appears in more than one chunk, providing some measure of sensitivity to frequency and proximity.

When the user makes selections from the ranked list, Finder can display the relevant paragraphs (usually more useful) or start at the top of the selected document by following a reference to that actual text, which is stored on disk drives managed by a VAX.
THIRD EYE

By contrast, Third Eye software, a three-person company that makes its money by selling the UNIX world's leading independent C debugger, CDB, is focused on minimizing resources and selling to people whose ratio of time to money is higher than that of TM's customers. Third Eye, in Menlo Park across a fence from SRI, was founded in 1982 by Peter Rowell, a former Xerox PARCer who worked on BravoX, the precursor to today's WYSIWIG editing environments. Third Eye's Elexir system is written in C and runs on UNIX on Suns and 386es (with mainframes and VAXen in the plans), performing adequately in a far more constrained environment than a Connection Machine, with a cost difference of two orders of magnitude ($20,000 vs. $2.5 million).

Where TM's Finder stores images of all the words in each chunk of text and searches them fast, Elexir takes an opposite approach, storing an inverted list of "topic" vectors representing the documents' topics and using that list to select the matches.¹ This list contains all the important words (stripped of suffixes) and phrases (identified by a proprietary algorithm) in the document base, along with the documents within which the word or phrase appears and its weight for each document (instead of its precise location). The weight of each word in each document topic-vector is determined by factors such as word frequency, capitalization, location (title or beginning of a paragraph counts more), type style (boldface counts more), and inclusion within a noun phrase. All these factors are selectable by the user; they cost more in indexing time, but provide more precise similarity rankings. In the end, information about position and other attributes is discarded, reduced simply to a weighting factor for each key word in a document. Each document topic-vector (and each query topic-vector) is represented as a listing of dimensions/words and weights rather than as a long vector of, say, 30,000 dimensions (words), most of them 0.

In essence, Elexir takes something of the approach that you might use with a Boolean system, searching for words (dimensions) that match the query and limiting its work to manipulating a subset of the data. Thus, Elexir computes weights only for matching words and sums them for each document, rather than computing and summing thousands of products that are mostly zero. For example, in the Juan and Alice example, a query for similarity to document A would select only the values for Juan in documents C and D. If there were other words in the query, their weights would be computed too, and added to the Juan scores for C and D respectively. The entire document vector would never be examined; nor would the underlying text be touched unless the user asked to see it.

Elexir offers a number of similarity measures from which the system-builder can choose, with a variety of ways of adjusting for document length and other factors. We have left out some details; Third Eye considers them proprietary. But in summary, the "similarity search" effectively compares the model vector against all the target vectors, and selects and ranks all those that exceed a specified cutoff point.

¹Elexir performs Boolean searches separately, using a simple inverted-list approach.
Since the company is looking for OEM business, it provides the maximum amount of information from which a system builder can select the appropriate measures for his customer set. (God forbid that an end-user should ever see the Jaccard [sic] coefficient of the texts he selects!) The overhead for Elexir's indexes is currently approximately 20 for the Boolean inverted list and 12 to 15 percent for the similarity vectors.

While its CDB business pays the rent (with customers such as Teknowledge, HP, Wang, and Siemens), Third Eye has slowly been readying its product for full-scale marketing. The company has sold a search tool to Informix, and is now talking to other customers we can't mention. It will make a company presentation at the PC Forum in Naples next month.

**COMPARE AND CONTRAST: DISSIMILAR STYLES**

<table>
<thead>
<tr>
<th><strong>Thinking Machines</strong></th>
<th><strong>Third Eye</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Documents broken into chunks</td>
<td>Document handled as a whole</td>
</tr>
<tr>
<td>Document chunks represented in long surrogate-code vectors</td>
<td>Documents represented as multi-dimensional vectors for similarity search</td>
</tr>
<tr>
<td>Search addresses each item separately</td>
<td>Search addressed text base through inverted list of vectors</td>
</tr>
<tr>
<td>Optimized for speed</td>
<td>Optimized for resources</td>
</tr>
<tr>
<td>Frequency, proximity data incomplete but statistically valid</td>
<td>Lots of optional data on proximity, frequency, etc., with a time penalty</td>
</tr>
<tr>
<td>Performance steady, optimized for large text bases</td>
<td>Performance unpredictable by user</td>
</tr>
</tbody>
</table>

Although it takes longer to do its indexing (and searching!), Third Eye's Elexir offers more precision and economy than the Thinking Machines Finder approach, which relies on statistical measures and allows spurious matches (which are easy to discard). Because Finder is stunningly fast, Thinking Machines hasn't had to bother with the clever tricks Third Eye specializes in -- in essence, smart indexing that makes subsequent searches easier and faster. In the long run, of course, this means that the Connection Machine's performance could be improved substantially when its brute force and Elexir's clever techniques are combined.

The Dow Jones system is no faster than any other in downloading full text over a wire, a process which has little to do with search time. However, the Dow Jones system should support a substantially higher number of users than Elexir. Moreover, Elexir's performance varies according to how susceptible a particular query is to its optimization techniques, while Finder's is fairly steady. (Elexir's speed is poorest when a query uses lots of frequently occurring words so that the resulting search space is large.) The big difference is that Third Eye uses indexes to minimize I/O, while Thinking Machines uses a high ratio of processors to memory (and a high ratio of overall resources to data) for the same reason.

*Release 1.0*  
31 January 1988
Classification on demand

The essence of smart indexing is to refine a large number of words into a searchable set of concepts accessible to humans. Similarity searching is merely a way of describing what we want when we have no clear taxonomy to use -- or when the text is not classified by the taxonomy we prefer. Traditional taxonomies -- tables of contents or abstracts -- tend to be fixed and to obscure hidden discriminations that may have been meaningless to the indexer but might be valuable to another reader. (That is, one person's poetry is another person's iambic pentameter, sonnets, couplets, and other distinct forms of rhapsody.) It is to overcome these fixed structures that we use word or similarity searches.

Indeed, when indexing, classifying and structuring are done by people, they are so slow and difficult that you can do them only once. But if you can do it again and again from different perspectives, classifying text by concepts is theoretically appealing.

The promise of the new generation of similarity tools is that they are so fast that they can indeed be flexible, since it takes just a few seconds to match a set of documents against any concept hierarchy you care to construct. That is, you could create a concept hierarchy -- a table of contents, say -- and assign documents to locations within it by measuring their similarity to a reference set of documents pre-classified along these lines by a human. Another person could come along with a different model and quickly reclassify everything. In the future, with a cleverly constructed thesaurus and fast indexing, you could create quite small indexes to large text bases, yet retain the flexibility of re-indexing to accommodate different users' conceptual spaces. (Information Access of Boulder, CO, is doing interesting work in building such conceptual indexes with a methodology for interviewing appropriate users to create what it calls J-spaces.)

On beyond retrieval

How will these systems be used? Thinking Machines is already finding high demand for its text-search capabilities, even among customers who were originally interested in other, engineering-type applications. Long run, the power of text will be realized when we don't consider things "text applications," but simply applications that include text manipulation as a function along with calculation, sorting, etc. IZE, with a simple algorithm (see Release 1.0, 12 May), is a first step: It organizes text chunks into a hierarchical (outline) structure, producing a sort of table of contents. Other systems (Lotus Agenda or MIT's Information Lens; see Release 1.0, 28 October 1987 and 24 September 1986, respectively) can sort your mail or take other actions based on some fairly simple criteria, generally the presence or absence of certain words, sometimes in defined fields. Other such applications might include problem-tracking systems that send out letters to aggrieved customers, marketing databases that track structured data and less structured published materials about competitors' moves, employee files, insurance records, etc.

As text retrieval and classification become faster and more accurate, they will constitute part of automated systems rather than remain standalone aids in finding material for human consumption.

Release 1.0 31 January 1988
NEWS OF THE WEEK: BALANCE OF POWER

Last month Ashton-Tate, the leading provider of personal computer database management systems, called a press conference that stunned the industry: A-T president Ed Esber shared the podium with Microsoft chairman Bill Gates to announce SQL Server, the next-generation dbms for the next-generation operating system, OS/2. Instead of competing tooth and nail, as everyone had expected, Ashton-Tate and Microsoft will now be working together to make the world safe for Ashton-Tate’s new version of dBASE, a dbms that until now looked to be on its last legs. For a second irony, the underlying technology doesn’t belong to Microsoft either: It is licensed from the hot new Berkeley start-up, Sybase (see Release 1.0, 1 July 1986).

Well, such alliances aren’t so unusual anymore — not in an era that includes alliances between DEC and Apple (development only, but a first date with big implications), Sun and AT&T, Oracle and Novell. (While alliances are grease that help separate corporate structures work in concert with minimal friction, too much corporate glue makes things sticky; hence IBM’s recent move to decentralize.)

In fact, Microsoft is not only a leader in technology, but a canny corporate strategist experienced in the geopolitics of the software industry. To recap: Microsoft developed the operating system for IBM’s original PC and sells its own version to virtually every maker of clones and compatibles (except in the Far East and Brazil?); at the same time, it is a major supplier of software for Apple’s Macintosh. Microsoft also sells spreadsheets, word-processors and other software in competition with applications vendors who rely on Microsoft’s DOS as the foundation of their programs.

Microsoft strengthened its alliance with IBM with an agreement to make and sell the follow-on OS/2 operating system released late last year: IBM sells it for its own PCs and PS/2s, while Microsoft sells it to other hardware vendors to include with their PC and (soon) PS/2 clones. That way, IBM can own a widely used "standard" without having to sell it to other vendors itself. But the operating system is only part of the support applications builders need today: Standards for data structure and for communications are becoming vital as networking and connectivity proliferate. While competitive confusion of solutions leads to more and better options in the short run, in the long run a combination of forces to foster standards makes economic and technological sense. There is intrinsic value to standards apart from the virtues of any particular one...until a new standard compelling enough to overturn the status quo arrives.

Filling the gap that IBM left

So where does the deal with Ashton-Tate fit in? Well, Microsoft’s joint effort with IBM doesn’t cover everything. Later this year, IBM will deliver OS/2 Extended Edition, a version of OS/2 that includes communications and database facilities developed by IBM alone.

With IBM set to offer OS/2 Extended without its pal Microsoft, what choice had Microsoft but to offer its own version (still lacking mainframe communications, however) to all those hardware vendors who cannot buy the complete version from IBM, and to all those end-user customers who want a second source? And how could Microsoft be sure of winning other than by teaming up
with its potential rivals in this market, seeing as it lacked the technology on the one hand and the installed base on the other?

The deal benefits almost everyone except everyone's arch-rival, IBM, which will now have to deal with a single strong competitor in the OS/2 database business rather than a group of weaklings. The team has three players:

**Sybase**, a start-up with a brilliantly conceived and executed product, free from the constraints of history and thus able to learn from the successes and mistakes of its predecessors.

**Microsoft**, a successful software firm, co-developer and co-marketer with IBM of the industry's next software "platform," but with little database expertise and a fairly full plate, working hard to get OS/2 1.1 out on time.

**Ashton-Tate**, long the leader in pc dbms, but now saddled with an aging product and a lagging development effort. Ashton-Tate, however, boasts 2 million users of dBASE II and III line. Those users should be able to move fairly easily to dBASE IV -- and thence to SQL Server.

Of course, this love fest leaves a few people out in the cold -- notably Oracle, currently the leader in IBM-style dbmss, especially on non-IBM hardware, and hopeful of making an impact on OS/2; and Lotus, which has announced and will deliver its own client/server dbms but will also end up supporting the Ashton-Tate/Microsoft/Sybase product as well as IBM's.

The joint product should be stunningly successful. It combines all the ingredients for success: good technology, an installed base, and Microsoft's strong relationship with other hardware vendors -- just as they buy OS/2, so will they buy SQL Server from Microsoft rather than from IBM. A number of other software vendors have already chimed in with their support -- not necessarily because they know anything about the product, but because this array of forces looks unbeatable. Only IBM can hope to compete effectively.

But the agreement is nothing for antitrusters to worry about. First, it takes a strong coalition to compete with IBM. Second, even IBM will be pleased to see software supporting its OS/2 and indirectly PS/2 (there's that software standby, mixed motives). And third, it's comforting to see companies do what they do best rather than duplicate each other's efforts.

**Ashton-Tate**

Ashton-Tate itself will shortly be bringing out its dBASE IV (in DOS and OS/2 versions), which will not initially support SQL Server -- given that it will arrive before SQL Server is ready next fall. By the time SQL Server is there, Ashton-Tate will be there too with dBASE IV "extended" -- a front-end version that supports SQL Server -- and with an existing installed base. Although specs aren't products, the plan is for dBASE files to be painlessly loadable into SQL Server, and for dBASE III applications to move easily into dBASE IV -- although to get the full benefits of SQL Server, obviously, you have to write applications with its client-server architecture in mind. Ashton-Tate's team includes Moshe "Query by example" Zloof from IBM, Harry "SQL" Wong from WordTech, and Mike "Supra/SQL" Benson from Cincom. Their expertise covers front-ends and the SQL interface nicely, but SQL is a way to talk to a dbms, not to build one.

*Release 1.0* 31 January 1988
The other guys

Firms such as Lotus, which is working on its own two-part dbms (front- and back-ends) with Gupta Technologies, are likely to make sure that their back-ends support SQL Server front-ends and their front-ends support SQL Server -- which is a basic goal of the client-server architecture and of the whole notion of layered standards and cooperative processing. One might expect that the Lotus product will be optimized for handling spreadsheet-style data, calculations and graphics, and would sell well in appropriate markets.

Oracle, as befits its current leadership status, is going it alone, although it has teamed up with communications expert Novell, potentially to offer a soup-to-nuts alternative to OS/2 Extended Edition with both database and communications components.

The view from IBM

The party line is that SQL Server won't compete with IBM's Extended Edition, out next fall. A powerful PC version of its best-selling mainframe DB2, the first release of the Extended Edition dbms won't support multiple users (leaving room for IBM's S/3X line of servers). Likewise, EE's communications facilities won't provide a gateway for other PCs on a network to use for reaching outside the network, but will simply let a single PC hook up directly to outside systems such as mainframes, minis, and dial-up links (as well as to a local network, of course). We initially found this a little hard to believe, since OS/2 is so well-suited to be a server operating system -- and as yet, a little rich for most individual workstations. Long run, we fully expect to see OS/2 take over power users' individual workstations -- perhaps when IBM comes out with server software for OS/2.

Microsoft

Microsoft indeed will not be competing with IBM, because it is Ashton-Tate that will actually be selling SQL Server for as long as the contract lasts, which is unknown. Hardware and software OEMs will be directed to negotiate with Microsoft, although they'll no doubt end up talking to Sybase about the technology. Software OEMs might include people with a front-end such as a project management system that would provide different views of project data managed by SQL Server, or perhaps a communications vendor wanting to sell its communications component in conjunction with SQL Server as a full-range server alternative to OS/2 Extended. As owner of the OS/2 SQL Server technology and manager of its use by third parties, Microsoft occupies a powerful position. Although customers will deal with Ashton-Tate for distribution and support, Microsoft will nonetheless have a wedge into A-T's huge installed base that should someday stand Microsoft in good stead.

Sybase

Sybase will now broaden its DataServer line to include an OS/2-386 platform. Customers with a Sun or VAX version of DataServer purchased direct from Sybase will be directed to a retailer for their copies of SQL Server, although Sybase will probably have a stock of shrink-wrapped copies purchased from A-T on hand for good customers. In the database world, Sybase will provide an alternative to (or alternate version of) IBM's Systems Application Architecture, running on everything from VAXen and mainframes (to communi-
cate with old files and applications) to (front-end only) PCs with DOS. More interesting yet, courtesy of Apple's 5 percent investment in Sybase, we can count on a Macintosh version, which will foster connectivity between Apple and the IBM world. Imagine hooking your Macs up to a Sybase DataServer on a VAX (of course) or your PCs up to a Mac-based SQL Server (!), or your Macs with the Sybase front-end DataToolset to DB2 on a mainframe. This last connection isn't yet possible, but it's quite plausible -- as is a PC-DOS version of Sybase's DataToolset, which would compete with dBASE IV for users' workstations.

Technology notes

What makes Sybase's SQL Server so special? Unlike most SQL dbms, it is targeted to transaction-processing (Oracle's coming announcements notwithstanding). Even more interesting is its active nature: It is a database management system. It does not merely store data for manipulation by an application, but it manages the data -- doing some work for the applications and also controlling what they are allowed to do. It does this with triggers (actions programmed into the database itself that occur every time a given data element is inserted, changed, or deleted) and stored procedures (canned routines that can incorporate much of an application's logic, but that are stored within the database for use by many applications). This allows for shared/reusable code (some might call SQL Server object-oriented), and ensures that different applications perform the same functions in a consistent way. Just as access to data can be controlled, so can access to the stored procedures. Thus a user application could be written by stringing together a very few stored procedures, such as posting an order, creating an invoice, and generating a pick list for the warehouse. The stored procedures and triggers would take care of ensuring that the customer met credit requirements, that the goods ordered were in stock, and how the pick list should be formatted. Security measures might enable the customer's credit to be checked without the user knowing any of the actual figures involved, but simply whether the customer had passed. A more senior person with access to the database itself could set those parameters. (This notion of integrity enforced by the database rather than by the application should appeal strongly to MIS types and other business people.)

On a technical level, SQL Server achieves its performance levels by bypassing the operating system (as it also does within UNIX environments, to the consternation of some folks). In effect, it doesn't even make use of OS/2's much-vaunted multi-processing capabilities (except to run concurrently with other applications), and looks like a single process to the operating system. This gives SQL Server much tighter control over the timing and sequence of events, and allows it to manage far more transactions in a given time period than if it paid the overhead of letting the operating system take over and arbitrate among transactions. SQL Server can afford to be optimized for mediating among database transactions, whereas an operating system offers more general facilities less tightly tuned for database functions. However, SQL Server does use OS/2's multi-tasking to manage I/O for access to disks and network facilities.

We pointedly avoid the mistake of calling database back-ends commodities: Some are good and others are better. The client/server architecture implies that you can mix and match, but not that you should be indifferent to what's on either end. Quality still counts at both ends.

Release 1.0 31 January 1988
NUCLEUS: AN ENGINEER'S DATABASE ENGINE

While the world focuses on SQL and user interfaces, one company has spent five years working on data storage techniques several levels lower, invisible even to the programmer, if not to the dbms provider. Specifically, Nucleus International (Santa Monica, CA) hopes to be the next Teradata. At the high end, it intends to compete indirectly with companies such as Teradata and Britton-Lee. At the low end, for its initial implementation, Nucleus will offer a PC AT database engine with a hardware assist.

Ted Glaser, architect of Burroughs' B5000 and of UNIX precursor Multics, and Ray Sanders, founder of Tran Telecommunications (sold to Amdahl), founded the company in 1983. Now their product is almost ready, and they have finally found a president (Derek McLeish, formerly with Monogram Software) and marketing team capable of explaining the product to prospective OEM customers, primarily dbms vendors.

The Nucleus board and software, scheduled for delivery next quarter at about $3000 to end-users, should fit neatly into existing environments, starting specifically with PC ATs running DOS and database applications using C and SQL. Like Teradata, Nucleus is best at handling large, relatively static databases, with superb query performance but so-so updating. (It should work as well with object-oriented dbmses as with traditional files and relational structures.)

Nucleus creates a more efficient data storage structure than traditional files, and offers far faster access times both by compressing the data on disk, and by allowing the system to hold more in memory. Like the text systems described earlier in this issue, Nucleus works by compacting data into vectors, and eliminating redundancy from a database. Overall, typical memory and storage requirements should be reduced to one half to one third of other databases.

Instead of storing the data each time it occurs, it lists each unique data element once and then specifies all the places where it occurs. Obviously, this can save a lot of space if much of the data is repetitive.

Take the concept of a domain, which is the list of all values in a certain field. If it's a key field, those values will be unique (people's Social Security numbers, for example, or their names, with the occasional occurrence of duplicates listed as, say, Alice Haynes 1 and Alice Haynes 2). But the data stored in the fields after those names is generally not unique. In other words, most domains are much smaller than a listing of all the data (including multiple occurrences) that they contain. For example, a company may have thousands of employees but only four divisions, seven salary grades, and so forth. Yet divisions and salary grades are typically stored for each employee.

So now suppose we list the company's four divisions in a domain vector, and then create a set of "row use" vectors which show which employees are in each division. Instead of rows and columns, you have a set of vectors representing the data in each column, with a positive bit indicating which rows use each of the possible values, as shown on the next page. (The first bit of each row use vector corresponds to the first "row" of the domain vector (or column) and so forth.)

Release 1.0

31 January 1988
### The old way

<table>
<thead>
<tr>
<th>Person</th>
<th>Division</th>
<th>Domain vector: Juan, Alice, Fred, Jack, Alex, Mike, ⋯ ⋯</th>
</tr>
</thead>
<tbody>
<tr>
<td>Juan</td>
<td>Sales</td>
<td></td>
</tr>
<tr>
<td>Alice</td>
<td>Marketing</td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td>Admin.</td>
<td></td>
</tr>
<tr>
<td>Jack</td>
<td>R&amp;D</td>
<td></td>
</tr>
<tr>
<td>Alex</td>
<td>R&amp;D</td>
<td></td>
</tr>
<tr>
<td>Mike</td>
<td>Sales</td>
<td></td>
</tr>
</tbody>
</table>

While it is complicated to explain, this is the sort of thing that software (especially software assisted by hardware) can do impeccably and fast. Obviously, it’s a little more complex logically (and takes some time) to update such a system with new data elements: You need to extend the domain vector or add a new row use vector. But the volume of added data is small, just a couple of bits here and there: The zeros don’t count because long sequences of zeros are further compressed by run-length encoding. If a field has multiple values (an employee with two divisions, say) the data compression is even greater.

Standard relational queries are also easy, since records with specified values can easily be retrieved (or joined) simply by finding (and combining) the appropriate row use vectors. (In the example above, "Find all employees where division equals marketing" simply selects all the employees listed in the second row-use vector.)

Will the Nucleus system sell? In a world where interfaces and standards are all the rage, and database engines look like commodities, sometimes it seems possible to ignore performance. Yet if Nucleus hooks up with the right database OEMs, its system could give those vendors quite an edge in performance-sensitive markets.

---

A standards-observant programmer awakened from a nightmare

---

Release 1.0

31 January 1988
In our last issue we asked, "How can the computer industry reconcile customers' technology-averse need for total solutions with their strategic need for differentiated solutions?" In this age of mission-critical systems, software is no longer overhead; it is production equipment. A user company's "solution" is its means of differentiating itself -- hardly something it wants to pick up off a shelf. In the expert system marketplace, this question becomes, "How can you efficiently build expert systems, which by definition contain specific, valuable information that people don't want to share?" The answer is to make problem-specific tools rather than problem-specific solutions.

This is the approach of Coherent Thought Inc., Palo Alto, a start-up founded three former employees of Teknowledge: Barry Plotkin, formerly coo, evp and general manager of knowledge engineering services (consulting); Jim Bennett, formerly principal scientist; and Peter Stokely, formerly a section manager. In all, 10 of the company's 11 people formerly worked at Teknowledge. (To put it bluntly, Plotkin, a controversial person at a controversial company, was fired last summer. Teknowledge recently filed suit against CT and Plotkin for what amounts to alleged employee-stealing, but not for theft of intellectual property.)

At Teknowledge these people worked not on the company's expert system shells (M.1 and S.1), but rather on a number of large custom projects for customers such as General Motors, NCR, Procter & Gamble and Motorola. There they faced and sometimes even solved the practical problems that customers face. Each project required a huge amount of work building data structures, rules, procedures -- most of them far more specific than anything in M.1 or S.1, yet generic across a fairly large problem set.

The particular knowledge for each customer wasn't much use for any other customer in a different field, but the group learned a lot building what amounted to custom tools for custom applications. At Coherent Thought, the goal is to abstract that process one step further back, and to build an environment for creating problem-specific customizable expert systems -- sort of a cross between shells and completed applications. "Ninety-five percent of expert systems vendors have never solved a real problem," says Plotkin. CT aims to embed problem-solving expertise as well as technical smarts into systems for sale to end-user customers.

CT's own environment to build these systems is currently under design, and the resulting execution systems won't be ready for deployment until 1989. The development environment uses Suns and Common LISP; the target environments include workstations running C and mainframes running COBOL under the CICS transaction-processing environment. The development environment will use object-oriented programming, with classes representing data structures, procedures, rules, and even language constructs of the target systems. These classes can be tailored to deal with a specific problem set, benefiting from easy reusability and specialization of code in the object-oriented world. Then, for performance and marketing reasons, each problem-specific set of program-element objects will be bound and compiled, retaining the power but not the flexibility of the original. Each set of system objects will do what it is built to do, but it will no longer easily be transformed and enhanced without the original development environment, which will be
proprietary to Coherent Thought (and possibly licensed to third-party re-sellers). That is, customers can add knowledge and data, but they can’t change the overall structure or the way the system attacks a problem.

For example, a set of objects to handle a generic diagnostic problem would include fault types, test routines and results, rules about the implication of test results, sequences of faults to focus on, groups of problems, relationships between faults, rules governing the sequence of tests based on test costs and likely results, and so forth. The system could be further specialized (by CT or by a third party) to deal specifically with electronic faults (opens and shorts) or with mechanical problems. CT’s other two broad areas of initial expertise will be system configuration and financial risk analysis.

Second-order magic

This approach solves a general problem builder/users have had constructing expert systems. It’s a fairly common misconception that expert systems don’t require programming: You just build a set of rules, and off you go. In fact, any but the simplest expert system needs a structure for efficient representation of its data and execution of its rules. (Structured systems are also much easier to inspect, debug and enhance.)

System builder/users need help in representing and classifying things and situations -- typically a few rules but lots of definitions and knowledge about sequence and organization. CT will provide a richer data structure based on objects, and generic rules where the user/builder need change only the parameters or other details. It’s not just a question of installing different rules into an expert system shell that fires rules in any order, but feeding specific information into something much richer that already knows how to represent a typical problem, and includes a problem-specific language, solution heuristics and problem-solving methods, explanation facilities, and standard sequences of questions, tests, and other data-gathering techniques.

CT’s big challenge now is to build the tools that can cross-compile these dynamic program element objects into static, executable, linkable code modules that will do their jobs smoothly. CT’s environment will illustrate the vision that object-oriented software is a living thing that can reproduce and evolve rather than either a tool or a set of programs. That is true, but since CT will be selling programs, it will have a monstrous version-management problem, keeping track of all the different implementations of its model systems and the compiled versions (to say nothing of the various iterations of its own development environment).

This is an ambitious project, but it makes a lot of sense. An empty database may be a sufficient foundation for a transaction-processing application, but a knowledge-based application is complex enough that it makes sense to use reusable code and object classes (cf. ON Technology, Release 1.0, 25 November and 30 December). Rules in an inference engine are like calculations in a transaction-processing application: They look like the smart part, but the real work is in building the structure that organizes the calculations, transactions, and rule-firings.
ASK DAN ABOUT HIS EXPERT EVADER

We recently spent some time with a tax expert system, Ask Dan About Your Taxes, and with an expert-system-based securities trader's workbench that we can't discuss in detail. Both these systems represent the future of expert system applications (as opposed to today's wizards-in-a-box) in that rules and even data don't constitute an application, they enrich one. Each system does something more than just make decisions or give advice; it helps carry it out. In the trader's workstation case, the application is information presentation and display plus some minor calculations of securities positions and offsets and ultimately trade execution, with securities selected according to certain rules; in Ask Dan, the application is tax filing, with all the data collection, management and calculation that entails.

While traders think in terms of movements and trends, tax advisers (and the IRS) think in terms of definitions and compliance: Does this qualify for such-and-such treatment? Were you truly trying to make a profit, or is building PC software just an expensive hobby? Ask Dan stresses explanations: What does the IRS mean by "dependent," "tax basis," "capital gain"? Tax strategy is an expensive, valuable, and widely applied field of expertise -- a combination of economics, rules of thumb, arcane laws, and investment options. Accountants charge hundreds of dollars per hour based on the minute or so of each hour that's devoted to expertise in an hour of calculation, questioning, etc. A tax accountant's calculations aren't mathematically complex; the trick is knowing what to calculate. By and large that's not reasoning; that's IRS edict.

So what is Ask Dan? In essence, it's a pre-configured tax model with variables and formulas, displayed as tax forms with links from fields to other form, support for linked what-if scenarios, hundreds of rules, and a huge amount of explanatory text. In addition, Ask Dan includes its own forms, to help you run through the logic of determining, say, whether your sister in nursing school is a dependent, how you can defer gains on the sale of a house, whether to file jointly with a spouse.

Or perhaps Ask Dan is a publishing system, organized neither hierarchically nor linearly, but in a hypertext structure built around tax forms. The biggest part of Ask Dan -- 365K out of 900K (it takes 512K to run) -- is devoted to explanatory text, an on-line tax code simplified (by designer Dan Caine himself) to be intelligible, and reorganized to be linked to the relevant part of the tax form. "The Tax Code was written independently of the forms. The skill was figuring out what things in the Code referred to what things in the forms," says tax lawyer Caine, 32.

In short, Ask Dan ($70 on a PC, mail-order) is by far the richest and most useful integration of hypertext, expert systems, calculation and other tools we've seen -- or that we're likely to see in some time. The expert system is only a piece of it: Caine's wife Claire, a Gold Hill employee, designed Ask Dan's inference engine in an hour. In fact, says Caine, the toughest part of the exercise was "taking convoluted tax provisions and putting them into a tree that could be parsed by the expert system."

For the user, the best part is that Caine focused on his goal -- easing the heartbreak of tax season. While Ask Dan explains the rules, it also helps you calculate the results and prepare the appropriate forms: It does the work as well as the reasoning.

Release 1.0
31 January 1988
FORUM DETAILS

The Forum is now sold out, although we are maintaining a waiting list to fill spaces due to cancellations.

Confirmed speakers include (additions and changes in italics):

Victor Alhadeff
Bob Berland
Gordon A. Campbell
Vittorio Cassoni
David Chapman
Peter Coffee
Michael Dell
John Doerr
Eric Drexler
Bob Epstein
Edward M. Esber
Gordon Eubanks
Robert Flast
Robert Frankenberg
Jean-Louis Gassée
William Gates
Jerry Kaplan
Mitch Kapor
Bill Krause
Bill Lowe
Jim Manzi
Mike Maples
Scott McNealy
Peter Miller
Dave Nelson
Bob Orbach
Safi Qureshey
Vern Raburn
John Roach
Ben Rosen
Mort Rosenthal
Mark Teflian
Larry Tesler
Edward Tufte
David S. Wagman
Kenneth R. Waters
Joyce Wrenn
Haviland Wright

Egghead Discount Software
IBM Application Systems
Chips & Technologies
AT&T
Cullinet
Aerospace Corporation
Dell Computers
Kleiner Perkins
Stanford ("Nanotechnology")
Sybase
Ashton-Tate
Symantec
American Express
Hewlett-Packard
Apple Computer
Microsoft Corporation
GO Corporation
ON Technology Inc.
3Com Corporation
IBM Entry Systems
Lotus Development Corp.
IBM Entry Systems
Sun Microsystems
ON Technology Inc.
Apollo Computer
47th Street Computer
AST Research
Cooper & Raburn
Tandy Corporation
Compaq Computer
Corporate Software
Covia (United Airlines)
Apple Computer
Yale University
Softsel Computer Products
ComputerLand
American Airlines
Avalanche Development


Release 1.0 31 January 1988
RESOURCES & PHONE NUMBERS

Ed Esber, Ashton-Tate, (213) 538-7714
Barry Plotkin, Jim Bennett, Coherent Thought, (415) 493-8805
Dan Caine, Legal Knowledge Systems/Ask Dan, (617) 923-2322
Conall Ryan, Ed Belove, Lotus Development, (617) 577-8500
Ray Sanders, Nucleus, (213) 450-1166
Rob Glaser, Bill Gates, Microsoft, (206) 882-8080
Bob Epstein, Sybase, (415) 548-4500
Peter Rowell, Third Eye, (415) 321-0967
Dave Waltz, Thinking Machines, (617) 876-1111
Daedalus, Winter 1988, an entire issue on artificial intelligence,
The American Academy of Arts and Sciences, c/o (617) 491-2600

COMING SOON...

- PC Forum program: Speaker profiles.
- Connectivity: Promises, promises.
- Parallel processing.
- Channels -- Micro and otherwise.
- Nitty-gritty experts: Are they intrinsically friendly?
- Airline experts.
- And much more...

Release 1.0 is published 12 times a year by EDventure Holdings, 375 Park Ave., New York, NY 10152; (212) 758-3434. It covers the pc, software, CASE, groupware, text management and connectivity markets, and artificial intelligence. Editor & publisher: Esther Dyson; associate publisher: Sylvia Franklin; circulation manager: Hyacinth Frederick; copy chief & consulting editor: William M. Kutik. Copyright 1988, EDventure Holdings Inc. All rights reserved. No material in this publication may be reproduced without written permission; however, we will gladly arrange for reprints or bulk purchases. Subscriptions cost $395 per year, $475 overseas; multiple subscription rates on request.
February 8-10  
IFIP conference on computers and law - Santa Monica, CA. Issues that just won't go away: Copyright, contracts, taxation, computer crime, legislative actions. Sponsored by IFIP and Los Angeles County Bar Law and Technology section. Contact: Michael Krieger, (213) 208-2461.

February 8-11  
UNIFORUM - Dallas. Sun-bashing, Apple-polishing, and more. Sponsored by /usr/group. Concurrent with useNIX. Contact: /usr/group, (408) 986-8840, or PEMCo, (312) 299-3131 or (800) 323-5155.

February 16-18  
DEXPO - New York City. Keynote by John Sculley. Can you have a honeymoon without being married? Come see Apple and DEC try. Sponsored by Expoconsul. Contact: Sandy Krueger, Hope Makransky at (800) 433-0880 or (609) 987-9400.

February 17-19  

February 17-19  
CASE Benchmarks - San Francisco. An incisive look at a variety of CASE tools. Rather than just present the companies and tools, moderator Vaughan Merlyn controls the proceedings and compares the various tools on a common scale. "Benchmarks" here doesn't mean raw numbers, but such measures as comprehensiveness, documentation support, and other practical metrics of effectiveness. Repeated March 14 and June 1 (see below). Sponsored by Digital Consulting, Inc. Contact: Scott Dorman, (617) 470-3870.

February 21-24  
ELEVENTH ANNUAL PERSONAL COMPUTING FORUM - Naples, FL. We moved it in search of variety and better weather. For further information, please see page 18 or call Forum director Sylvia Franklin at (212) 758-3434.

February 24-26  

February 25-27  
Workshop on technology and cooperative work - Tucson, AZ. Sponsored by Bell Communications Research and the University of Arizona. Contact: Robert Kraut, (201) 829-4513 or Jolene Galegher, (602) 621-7477.

March 1-3  

March 3-5  
ABCD visions '88 - Newport Beach, CA. Keynote by John Sculley. Sponsored by abcd, the microcomputer industry association. Contact: Bernie Whalen, (312) 240-1818.

Release 1.0  

31 January 1988
<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>March 7-10</td>
<td>IEEE conference on computer workstations - Santa Clara. Sponsored by IEEE. With Sun's Bill Joy, and sessions on distributed systems, computer-supported cooperative work, and OS/2. Contact: Pat Mantey (408) 429-2158 or Robin Williams, (408) 927-1842.</td>
<td></td>
</tr>
<tr>
<td>March 14-16</td>
<td>CASE Benchmarks - Dallas. An incisive look at a variety of CASE tools. Rather than just present the companies and tools, moderator Vaughan Merlyn controls the proceedings and compares the various tools on a common scale. Repeated June 1 in Chicago (see below). Sponsored by Digital Consulting, Inc. Contact: Scott Dorman, (617) 470-3870.</td>
<td></td>
</tr>
<tr>
<td>March 14-18</td>
<td>Artificial intelligence applications - San Diego. Sponsored by IEEE. With Charles Bachman, Eric Bush, John Landry, Jan Aikins (Aion), Walt Scacchi (USC), among others. Call Richard Greene, (301) 468-3210 (exhibits) or IEEE, 371-0101 (program) or Paul Harmon (415) 861-1660.</td>
<td></td>
</tr>
<tr>
<td>March 16-23</td>
<td>Hannover Fair CeBIT - Hanover, West Germany. Contact: Donna Peterson Hyland, Hannover Fairs USA, (609) 987-1202.</td>
<td></td>
</tr>
<tr>
<td>March 20-23</td>
<td>ADAPSO SPRING CONFERENCE - Palm Desert, CA. Software and services vendors at the oasis. Contact: Sheila Wakefield, (703) 522-5055.</td>
<td></td>
</tr>
<tr>
<td>March 27-30</td>
<td>Software Publishers Association spring conference - Berkeley, CA. Contact: Jackie McDonald, (202) 452-1600.</td>
<td></td>
</tr>
</tbody>
</table>

Release 1.0  
31 January 1988
<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 7-10</td>
<td>13th West Coast Computer Faire - San Francisco.</td>
<td></td>
<td>Contact: Jason Chudnofsky at Interface Group, (617) 449-6600.</td>
</tr>
<tr>
<td>April 10-13</td>
<td>AEA Conference - Washington, DC. With companies under $75 million in annual revenues.</td>
<td></td>
<td>Contact: John Baumeister, (408) 987-4200.</td>
</tr>
<tr>
<td>April 11-14</td>
<td>AIIM show - Chicago. Information and image management.</td>
<td></td>
<td>Sponsored by Association for Information and Image Management. Contact: Sue Wolk or Betty Garrett, (301) 587-8202.</td>
</tr>
<tr>
<td>April 11-15</td>
<td>IEEE Tenth international conference on software engineering - Singapore</td>
<td>From an international perspective. Sponsored by IEEE and NCB Singapore. Contact: Tan Chin Nam or Lim Swee Say, (65) 772-0200.</td>
<td></td>
</tr>
<tr>
<td>April 19-21</td>
<td>CEPS/Spring '88 - Chicago. Corporate electronic publishing systems.</td>
<td>Sponsored by Cahners and InterConsult. Call Mike Driscoll, (203) 964-0000, or Paula Wertman, (617) 547-0332.</td>
<td></td>
</tr>
<tr>
<td>April 27-29</td>
<td>Seybold Technology Forum - Cambridge, MA. &quot;Distributed network computing: A journey into the future.&quot;</td>
<td>Sponsored by Patricia Seybold's Office Computing Group. Discussions ranging from communications protocols to computer-supported cooperative work. Call Catherine Cooper, (617) 742-5200.</td>
<td></td>
</tr>
<tr>
<td>May 3-6</td>
<td>CASExpo - Dallas. Managed by Arthur Young &amp; Co.</td>
<td></td>
<td>Contact: Ken Burroughs, (703) 845-1657.</td>
</tr>
<tr>
<td>May 9-12</td>
<td>Comdex Spring - Atlanta. Peaches and PCs.</td>
<td></td>
<td>Contact: Jane Wemyss at the Interface Group, (617) 449-6600.</td>
</tr>
<tr>
<td>June 6-8</td>
<td>Artificial intelligence in electronic publishing - San Jose.</td>
<td>Sponsored by the Graphic Communications Association. Applying AI to design, content, process, etc. Contact: Marion Elledge, (703) 841-8160.</td>
<td></td>
</tr>
<tr>
<td>June 19-22</td>
<td>Congress VI - Paris. The world computing services industry gets together. With speeches by Charles Marshall, AT&amp;T; Max Hopper, American Airlines; Anthony Craig, GEISCO. Meet your potential partners or competitors abroad. Sponsored by national trade organizations, including our own Adapso.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Contact: Phyllis Cockerham, (703) 522-5055, or Diana Kirby, London, (441) 405-2171.

July 12-14
CASE '88 - Cambridge, MA. Second international workshop on computer-aided software engineering. More academics and less hype than most CASE conferences, for better or worse. Sponsored by several academic institutions. Call Pamela Meyer, Index Technology (organizers), (617) 494-8200, x454.

July 24-27
IEEE conference on neural nets - San Diego. The second, because the first was so successful. Contact: Richard Rea (exhibits), (619) 222-7477, Sue Varga, (619) 281-8991, or Nomi Feldmann (papers), (619) 453-6222.

August 1-5
SIGGRAPH - Stanford, CA. Sponsored by IEEE, ACM and SIGGRAPH. Contact: Adele Newton, (519) 888-4534.

August 8-12

August 22-26
AAAI-88 - St. Paul, MN. The seventh annual. Sponsored by the American Association for Artificial Intelligence. Contact: Claudia Mazzetti, (415) 328-3123.

September 14-17

September 25-27

September 25-30
OOPSLA - San Diego. Object-oriented Programming: systems, languages and applications. Sponsored by ACM. Contact: Allen Otis, Servio Logic, (503) 644-4242 or Barbara Noparstak, Digitalk, (213) 645-1082. (The conference section of OOPSLA is Wednesday through Friday (28-30), so you can catch most of CSCW first if you miss the OOPSLA tutorials.)

September 26-28
Second conference on computer-supported cooperative work - Portland, OR. Sponsored by ACM. Contact: Suzanne Sylvia, (617) 225-1860.

October 11-14
Info Show - New York City. Contact: Frank Fazio, Cahners Exposition Group, (203) 984-0000.

October 23-28
Monterey Classic - Monterey, CA. Contact: John Baumeister, (408) 987-4200.

Please let us know of any other events we should include.
-- Anita Fowler

Release 1.0 31 January 1988
SUBSCRIPTION FORM

Please enter my subscription to Release 1.0 at the rate of $395 per year in the U.S. and Canada. Overseas subscriptions are $475, airmail postage included. Payment must be enclosed. Multiple-copy rates on request. Satisfaction guaranteed or your money back.

Name

Title

Company

Address

City State Zip

Telephone

How did you hear about Release 1.0?

Please fill in the information above and send with your check payable to: EDventure Holdings Inc. 375 Park Avenue, Suite 2503 New York, NY 10152

If you have any questions, please call us at (212) 758-3434.

Sylvia Franklin
Associate Publisher

Release 1.0 31 January 1988