OBJECT-ORIENTED DATABASE ROUNDPUP

Since our review of object-oriented databases last year (Release 1.0, 88-12, and also 87-8 and 88-9) the playing field has enlarged considerably. In several cases, databases and applications are co-evolving, as corporate partners (the new name for beta customers) work with pre-production models because they don't want to wait. Currently, several new vendors are preparing to join pioneers Servio Logic, Ontologic and Symbolics in the object-oriented database sweepstakes. (Another pioneer, Graphael, no longer answers its phone.) Each reflects its developers' backgrounds -- commercial or research, database or engineering, tools or substrate. Call our previous discussion a set of requirements; here are the specifications, with updates on individual players.

Right now, few people take this market seriously because no one expects to give up DB2 or Oracle or dBASE for the next pretty object-oriented face that comes along. But that's beside the point. Object-oriented databases are not going to replace regular ones anytime soon. Instead, they will replace file systems and provide database support to a new class of applications: CAD/CAM, electronic publishing and other design/configuration processes with intricate data structures, multiple users and multiple applications.

This is an accumulation of technologies, not a replacement cycle: from batch processing with relational predecessors Codasyl and network databases; to relational-based interactive databases for ad hoc queries and analysis and even transaction-processing; and on to object-oriented databases for design and configuration tasks, where you can easily use and navigate the structure as well as the data in a database while maintaining integrity.

While the OODB market once appeared limited to traditional design applications, it is becoming increasingly clear that it will also support office automation in general and task automation in particular. In fact, task automation can be seen as real-time design of a work product -- whether it's generating an answer to a customer's question, a work schedule, an itinerary or a policy statement.

PLEASE ANSWER OUR QUESTIONNAIRE

EDventure Holdings Inc., 375 Park Avenue, New York, NY 10152, (212) 758-3434
Out in the marketplace, the notion of object-oriented databases is gaining more credence, both from people who understand them and from those who do not. Once seen as a tool only for engineers, object-oriented databases are now deemed an appropriate back-end for storage and manipulation of documents, diagrams and other rich data — the role H-P's Iris will play for its NewWave and that IBM's DB2-based Repository (page 25) will play for many CASE tools. However, Index Technology, after thoroughly examining the question, has decided that it needs nothing less than the real thing — a database object-oriented all the way down — and is working closely with Ontologic to shape its Ontos for use as the foundation of Index's next-generation CASE tool suite. Kodak is using MCC's Orion prototype in-house to support configuration of systems for customers.

We believe many other vendors and users will make the same decision for "object-oriented all the way down," for both performance and functionality. It's clear that economics, need and culture make engineering the first and easiest market to crack. But in the long run, competition for engineering and design will raise the appeal of newer markets, where customers will be running Lotus 1-2-3 Release 6 in eight dimensions, Word 7 and GroupSupport 2 under OS/4 on 686 machines and Unified UNIX X.3 on 100-MIPS RISC machines.

Not by relational alone

But first, what do we need object-oriented databases for? Aren't relational ones good enough? Unfortunately, no. In theory, you can do everything with a relational database; call it a Turing data store. You can keep pointers to objects in it, and you can keep the links in it, and assemble whatever data structures you want as you need them. But that's like saying you can keep a car in a file cabinet because you can file the engine components in files in one drawer, and the axles and things in another, and keep a list of how everything fits together. You can, but you wouldn't want to. Nor do you want to keep everything in a pile in the garage (a single file) that has to be assembled each time you want to use it or even just a part of it.

True, if all you want is little pieces of data findable by values (Alice's sales vs. quota for 1988, say), you're better off with a relational database; and if you want to do the same thing to thousands of pieces of data (raise everyone's salary 10 percent, say), you're better off with a traditional ISAM database. An object-oriented database comes into its own when Alice wants to assemble an itinerary based on current pricing and product availability while Juan over in marketing keeps on proposing new customers and the finance department keeps on changing travel restrictions.

A database does more than just store (representations of) things; it manipulates them and maintains their integrity. As with regular data but 20 years later, there will be a proliferation of applications and tools based on

---

1 Traditional relational vendors are trying to head the OODB vendors off at the pass with promises that they too will have objects within a couple of years. Mike Connell, general manager of Servio Logic's data systems division and one of the few OODB vendors actually shipping product, is convinced that these promises are behind the shift in his customer base from commercial accounts to government and aerospace over the past year.

Release 1.0	 22 September 1989
richer data structures that need a shared database on the back-end, with a variety of applications and users on the front-ends. At present these applications simply load entire files and then recreate the structures in memory -- Objectivity president Bob Field calls them file-suckers. With an object-oriented database, these applications could select the components of text or data or images or diagrams they need from shared, structured data that can be used concurrently by other users or applications. These applications and their users need the same assurances of integrity across applications and the same multi-user access as a relational database offers for more "regular" data.

Objective benefits

Relational databases derive a virtual structure at execution time by "relating" sets of data based on values, while object-oriented databases contain explicit structure and assign objects and instances (types and data, so to speak) immutable IDs. The structure as well as the data is permanent (albeit malleable), and needn't be reconstructed at runtime. A "view," a familiar database construct consisting of data selected from multiple tables by "relating" values at runtime and loaded into a single table, has an analogy in object-oriented databases, where a subset network of relationships is selected by traversing pointers from object to object (ideally within clusters) within a larger, incomprehensible web.

This means that there's a lot more information in an object-oriented database. Both data and transactions are defined to/in/by the database, and if you follow the rules you can't fool around with the data (encapsulated in objects) or the structure without the database's "knowledge." In a traditional database, as a user, all you can do is add or delete records and change values (in general); in an object-oriented database, you are dealing with explicit relationships and inheritance schemes and methods, and managing discrete transactions on discrete attributes of objects. The database itself can determine on a fine-grained level which transactions conflict and which can co-occur without breaching integrity. For example, design rules could be enforced within the database, not by an after-the-fact separate application.

Objective challenges

The bad news is that storing and providing easy access and concurrency control for data and code that are linked and constrained by an intricate web of orthogonal relationships is much tougher than what a typical database does now. That gives object-oriented databases a reputation for slowness and overhead, but the question is, "compared to what?" They handle a task not handled before, and use a web of indexes and caching algorithms (never seen by users or even most developers) to make the whole system work.

In any database, data is stored for efficiency. In a relational database, an item is put in the table or file it "belongs" to; applications are separate. In an object-oriented database, data typically "belongs" in many places, and you store not just the item but the related behavior (the equivalent of application code). So you store data (or a pointer to it, akin to a foreign key) in many places, using unique object identifiers for each object (or instance, depending on terminology). With each object are stored its attributes, properties such as age or color, which have no independent
existence (in the database) away from the object that possesses them. Also along with each object are stored pointers to other objects it has relationships with (actually, object IDs) and pointers to the code that implements its behavior. The co-location of the data most likely to be used together is called clustering, and is a key factor in database performance.

As happened with relational databases during their low-performance infancy, much of the OODB work of the past and the next few years will focus on practical techniques for performance: They include indexing and efficient clustering of objects (by class hierarchy or with the other objects they're related to; i.e., do you store sales employees together, along with the sales employee class which contains relevant code, or with the company each one works for?). In addition, clever caching allows you to maintain performance and efficiency regardless of previous physical storage decisions. For example, once the user has loaded a single employee instance along with the employee behavior (retrieved automatically), most caching systems would retain the employee behavior in the expectation that the user or application would turn its attention to another employee instance.

Object story: Statice in action

Houston Power & Light, for example, is using Symbolics' Statice to store and manipulate data for power-plant maintenance schedules (one of very few real-live production uses of an object-oriented database we could find, but many more are in development). Some data come from a mainframe payroll system that does 3000 to 4000 transactions per day; each schedule contains 2000 to 10,000 discrete activities. Scheduling requires manipulation of teams of individual workers each having a unique set of skills, pay rates and work schedules; physical considerations such as distances from one plant to another; kinds of work needed for each component of a generating unit (several per plant); and specific repairs required at each plant. Each power plant has a unique set of equipment, but each instance of that equipment shares many characteristics with others in its class, and some pieces of equipment work together. Do you store all the like equipment together, or all of a plant's components together? (In this case, you keep the plant components together, because plants are scheduled one by one, but the answer might be different in other circumstances.)

Some of the attributes of an object are relationships to other independent objects, such as tools, parts, equipment a particular class of worker is qualified to repair, and groups of items (such as work teams), or complex objects. When an object is selected, the system loads the object ID (a unique number) and the OIDs of the other objects it has relationships with, according to the use the user or application specifies.

How do you combine all these permutations to generate effective schedules? Houston P&L did it so effectively with Statice (plus an expert system) that

---

2 This depends on the view of the system creator. A standalone inventory system would consider "manufacturer" an attribute; a purchasing operation might well consider manufacturers/suppliers to be discrete objects. If both used the same corporate database, an application developer should be able to specify that distinction, which would have an impact on caching behavior.

Release 1.0

22 September 1989
overtime dropped by $442,000 (38 percent) year-to-year. It used to take three hours to load a file to start scheduling one plant (out of 25), but it now takes five minutes because the system is able to load only the specific objects necessary.

The overall process used to take five days (of elapsed time) and now takes just a few hours because the impacts of hypothetical changes can be what-if'd and reflected that much more quickly. The expert system restructures only the relevant data instead of recompiling it all each time a change is made. But the real benefits that scheduling and controls supervisor Steve Geiler hopes for aren't yet visible: He expects better maintenance to lead to fewer blackouts and less "unscheduled maintenance" in the future.

A kaleidoscope for the mind

Object-oriented databases offer a new way of using and thinking about databases. Where applications once used to call data from the database, now they rely on both data and processes in the database, using defined transactions as well as defined data. The database plays an active instead of a passive role. While object-oriented programming is frequently thought of as hierarchies of data types or objects or "nouns with verbs attached," in fact what defines those objects is their relationships with other objects (attributes that are other objects instead of primitive strings or values) and their behaviors -- or allowed transactions. What makes a sales employee different from a generic employee? Not his name (data), but his behavior, or the code routines the object employs. The data is incidental. Object-oriented programming is not the creation of data hierarchies but rather the creation and modification of behavior (read application) hierarchies. The class hierarchy provides a convenient way of organizing and linking code modules (through inheritance) to the data it acts upon. The database holds them together and allows you to take them out a piece at a time.

If you're so smart, how come....

Wonderful as object-oriented databases are, they can't overcome reality: They model complex items and relationships, and representing those is never going to be as simple as modeling a flat vanilla table of names and addresses or checking-account transactions. The achievement of object-oriented databases is to make the impossible possible, rather than to make the possible automatic. Object-oriented programming doesn't guarantee reusable code: It provides a framework for it.

It's not what you don't know; it's what you know that ain't so. Interoperability at a technical level doesn't necessarily mean interoperability at the modeling-reality level. Objects that mean one thing in one application may have quite different meanings (or underlying assumptions) in another. This will become an increasingly important problem as we move from the current examples of computer-oriented objects -- screen icons and buttons, files and data types -- to objects that attempt to mirror reality, such as price-earnings ratios, vice president, workday, and the phrase "let's have lunch." All these items would be defined quite differently by different people. Yes, there is a cost for this lunch, too.
ATTRIBUTES

On page 14 we provide company profiles of major vendors of object-oriented databases. They include (shipping now) Servio Logic (GemStone) and Symbolics (Statice); and (shipping soon) Altair (O2), Artemis (Orion II), Hewlett-Packard (Iris), Juniper (Persist), Object Design (ObjectStore), Objectivity (Objectivity/DB), Object-Sciences (Object-* series) and Ontologic (Ontos, replacing Vbase, first shipped in 1988). But first, a description of the features of the superclass from which the products inherit their behavior:

As you read through these descriptions of the facilities provided by most object-oriented databases, you will notice a continuous tension between transparency/invisibility/automaticity on the one hand, and power/control/flexibility on the other. However, these trade-offs are not always necessary. In general, the best systems handle tough issues invisibly and by default, but leave hooks and tools for sophisticated schema designers and application developers who believe they can improve on the vendor’s defaults. Negotiating the delicate balance between power and ease of use, providing a maximum of customer options beyond the defaults, and supplying high performance to back it all up, will be keys to market success.

- **platform** -- Most object-oriented databases come out of and will be sold into the engineering/technical world, and so they tend to show up first on Suns and VAXen. Statice runs on Symbolics machines; Ontologic, because of its ties with IBM-oriented Index Technology, will move Ontos to OS/2; and Juniper’s first implementation will be on a Mac, but aside from that the primary environment is UNIX and Suns.

- **configuration/architecture** -- Most databases, object-oriented or otherwise, begin life as single-user, single-platform systems. However, they’re not much use that way. The next version is usually a single-server, multiple-client implementation, supporting concurrent access by a variety of users and applications. The ultimate -- and still only a dream even for most relational systems -- is a fully distributed architecture where data can reside on any of a multiplicity of different servers, hooked to a variety of clients, and nobody need know the difference. (Throw in redundancy and guaranteed integrity for good measure.) Obviously, it’s much easier to support different clients than to support different servers one by one -- as Servio Logic does now -- and even more difficult to support different distributed servers working together seamlessly -- as everyone promises to do eventually. Servio also already provides distributed processing (page 23).

- **underlying storage mechanism** -- You can put an object-oriented data model on top of a traditional relational storage mechanism, as Iris does and as the IBM Repository (page 25) will, but an object-oriented storage system is generally necessary for peak performance. It may

---

3 We do not cover DEC’s longstanding Trellis/Owl project because commercial plans for it are unclear. At this point, it is an object-oriented programming environment that can serve as a front-end to a database; it will be shown as a front-end to DEC’s own relational (SQL-based) rdb at OOPSLA. In the future, it could also sit on top of an object-oriented storage system.
seem like overkill for people who do simple transactions, but engineers
don't like to wait two hours for a complex engineering model to load.
(Ideally, one can do both, in a hybrid system with object-oriented
storage as well as access to foreign databases, and thus to existing
relational data. Vendors who do this, such as Servio Logic and soon at
least Artemis, Iris and O₂, will certainly have a market advantage.)

Once the storage system is built (several vendors are using a rewrite
of WISS, a storage system developed at the University of Wisconsin) the
big issue is to determine optimal defaults for clustering and indexes
and caching, while leaving hooks for sophisticated users to modify the
defaults. Are objects stored with their class hierarchy, or with the
complex objects they are part of? What about a single object that
belongs to many different complex objects? What about a class that in-
herits from several other classes?

- **classes, types and objects** -- Are classes treated as just another set
  of objects? In traditional, objects-all-the-way-down systems based on
  Smalltalk or Common LISP, classes are objects, and can be manipulated
  as such. (However, for practical reasons most systems write-protect
  classes from changes by casual users and applications once a schema
  designer has finished building the system.) GemStone, Statice, Orion,
  Persist (by way of its ODL language, even though it's C++-oriented) and
  Object Design follow this model. In the C++ world, by contrast,
  classes are types; Ontologic, Iris and presumably Object-Sciences and
  Objectivity will generally follow this model. It's the usual trade-off
  between power and flexibility (on the class-object side) and robustness
  (on the class-type side).

- **schema modification** -- Can you teach on old object new tricks? The
  reason classes-as-objects matters is that it makes schema modification
  easier. Schema modification is the dangerous practice of restructuring
  a database, which can get very complicated when it already has data/
  instances in it; how do the changes propagate to the old data? This
  will be a big issue when vendors start to release upgrades of OODB-
  based products. The trick is to do as much as possible automatically,
  while maintaining the integrity of existing data, and to provide
  facilities for the resolution of the inevitable conflicts between old
data and new schema. Version management can ease the pain, by letting
you maintain old and new versions of classes simultaneously.

- **object extent** -- When you use or lock an object, how much of it do you
  want -- all its pieces and functions or just a few of them? Some of
  these answers can be determined by default, explicitly by the database
  or application developer, or only at runtime: When you load a letter,
do you want editing functions, printing functions, or access to the ob-
jects it references? (Smart caching may pre-fetch as much data as pos-
sible, but successful implementation pretty much depends on tuning for
usage patterns rather than any general principles. Cf. the capabili-
ties of Borland's VROOMM, Release 1.0, 89-6.)

- **collections and complex objects** -- Another way to extend objects is to
  create a collection or a complex object, which comprises multiple,
heterogeneous objects. Relational databases gave us relational algebra
and sets, but the sets hold only data of the same type.
Collections come in a variety of forms -- both class hierarchies and other less regular, more arbitrary structures that model reality: our office and everything in it; the people, buildings, units and other things that make up a company; the interconnected parts that constitute an airplane; or the suppliers of those parts. Each of those parts also belongs to a class, from which it derives its attributes -- size, creation data, maintenance history -- and it has a set of attributes different from those of other items in the collection. (The Boeing plane is, say, 10 years old, but the engine, made by Pratt & Whitney, is new. The engine has a fuel-consumption rate, while a wing-flap has width.)

Collections underlie many functions: They can be manipulated by users and applications, locked for transaction management, and may also be clustered or indexed for easy access. Queries are real-time generators of collections: All the people who can't drive, or all the parts made by Microdot, or all the people whose spouses work at companies with a particular kind of benefit plan.

- **Language support** -- The market's language of choice is C++. Few vendors endorse it with enthusiasm, but they all promise to support it. The first few efforts had their own languages -- Ontologic's TDL and COP, Servio Logic's Opal. You can still make arguments for those languages' superiority, but it doesn't much matter: People want C++.

Several market contenders have felt (and others have watched from the dressing room) the market's insistence on C++, and are loath to tinker with the language. Note that this market is extremely language-conscious because the quality of "object-oriented" so far has been implemented mostly through languages. Customers are reluctant to go ahead with application development, which would mean committing to a particular object-oriented database. Vendors' assurances of portability aren't enough; the C++ standard is an anchor to cling to (if a tenuous one), and has become a key market factor.

Benefiting from such faith (which no doubt persuaded AT&T of the language's value and encouraged it to improve it), C++ has in fact become more deserving of this support. Release 2.0, has just now become available from AT&T, and has facilities for multiple inheritance and better error-handling.

Each vendor will end up offering similar functions but in different, incompatible ways, which will end up making things tough for users who want to switch and the vendors they want to switch to, but comfortable for the guys in the lead. Although some vendors make a point of "not extending C++," the fact is that there are certain things you cannot do in C++, and the issue is only whether you do them in a separate new language, in language extensions, or in subroutines and methods hidden in and inherited from classes in the database class hierarchy -- or a combination. (The extensions and function calls are open in the sense that every vendor would like its own set to become a standard, but they are all different, just like other vendors' extensions to SQL.)

The extended-C++ approach offers the ability to define objects in C++ and automatically make them persistent and register them with the database. A second benefit of language-extension is that the compiler knows about the data structures and can make everything much more effi-
cient than keeping schema and object definitions separate as subroutine calls or embedded in the objects themselves. Many of the C++-oriented vendors are taking this approach.

Detractors say that they are extending the C++ standard, and prefer an approach where you call database functions from within C++ (or whatever). Database folks know that databases have their own languages, data definition and data manipulation languages, or DDL/DML, and that standardization on SQL (for example) has proceeded independently of the (slow) movement from COBOL to C, C++ and other languages. Whether DDL/DML (either SQL or an object-oriented version) is embedded in extensions of a traditional language with a pre-processor or called separately without "extending" the host language is partly a matter of convenience -- but in the end it works better and faster to embed it in the language, where it can be compiled and optimized. It might make sense to keep database and programming language separate in the old model, but in this new model much of the application set is "inside" the database, and the distinctions are of necessity blurred.

Aside from compilation efficiency and runtime performance, the issue with language support is allowing users (or applications) to create persistent objects easily, and to get them in and out of storage without explicit stores and fetches or puts and gets. This is akin to virtual memory, but managed more cleverly. In general, the store-and-fetch issues concerns application developers, who have a more complicated job when they have to manipulate persistent objects differently and explicitly store or load them. From the end-user's point of view, someone else takes care of the problem and performance is not much affected either way. The advantage of put and fetch comes when you want user control over the extent of locking, but a vendor can provide this capability on an optional basis anyway.

- **multiple inheritance** -- Multiple inheritance gives the object-oriented model much of its value -- and much of the difficulty of implementing it. In practice, multiple inheritance is key to a major benefit of object-oriented programming -- the reusability of code. By assigning an object a new class, you automatically give it all the powers (behaviors and relationships) of that class. In other words, by adding a taxpayer module from a class library and making employees taxpayers, you can add all the applications concerning paying taxes to the employees' repertoire. The advantage is that the taxpayer code can use many of the definitions and data already set up in the employee system, and vice versa. (However, these things can be deceptively easy: The disadvantage comes when the IRS discovers that your employee system and the purchased taxpayer class have different ways of computing income.)

Multiple inheritance mirrors the real world, full of Renaissance objects. You can work around it, for example by creating a collection based on a certain attribute value, but sometimes that gets clumsy. Animals and possessions, for example, are two rich, independent classes where you can't easily simply add an attribute or two to either side to make a new subclass of pets. (Animals have an entire, well-known taxonomy; possessions have a rich set of attributes including owners, prices and previous owners that can't easily be tacked on.)
Not everyone is so enamored of multiple inheritance. Altair's Bancilhon, for one, considers it a check-mark item: "The only reason we have multiple inheritance is so that we can say we do. It's such a visible thing, but the cases where it gives you a big gain are pretty rare."

Multiple metaphors: Here's a (joke) example of multiple inheritance, as well as multiple descendants. In fact, some people might consider it untidy, because the databases in the subclass below are not particular kinds of the databases above, but they do inherit characteristics from them. What we're illustrating here is simply that you can get your characteristics from more than one parent. (When parents' attributes conflict, the conflict is resolved arbitrarily or by rules.)

The diagram above is messy, because reality is messy. It shows each of the object-oriented databases mentioned here, plus some superclasses (some of them are instances as well as classes). Some databases inherit characteristics from several parents. Meanwhile, we have expanded WiSS (Wisconsin Storage System) as just one example to show pointers to two people who also have relationships to other databases. Each other database has similar relationships with a variety of people. We could also have included other classes or added attributes to reflect language support, whether a product is shipped or vaporware, and so forth. The decision whether to reflect these things as individual attributes (which could form the basis for a collection) or to create classes sharing such characteristics is arbitrary, and depends on intended usage. Straight lines show inheritance; dashed lines show the WorksFor-EmployedBy relationships. Database wizards are in rectangular boxes; databases are in rounded boxes.

Release 1.0 22 September 1989
transactions and concurrency management -- The issues above concerned support for objects; transactions and concurrency are a feature of databases -- but rendered more complex when they apply to objects.

Transactions aren't the simple, instantaneous, atomic events they are in regular databases, but can involve a sequence of events or long time periods during which many things may occur. During a traditional transaction, the data you access is assumed to stay the same; during an object-oriented transaction, lots of things may happen in the meantime, and the objects have long tails. A transaction may be the sum of many smaller, quicker ones (nested transactions), and the system may be held in a state of non-integrity for a long time. For example, Juan and Alice could spend days working separately on hypothetical wing designs. Version management can take care of some of these issues, but how could you fine-tune the differences?

Optimistic concurrency management says let anyone (with proper authority) check out whatever he wants; notify other users when someone checks an object back with changes. Control goes to the first person to check it back rather than the first to check it out. Pessimistic control prevents access by others when the first person gets hold of the item. Complications come in defining what is locked, since there is no such simple thing as a file or a record (see "object extent," above): If you lock an object, do you lock all the things it "owns"? These issues must be decided by a schema designer or database creator.

Optimistic concurrency is certainly cheaper to use (it doesn’t lock things up all the time and is rarely even needed), but it’s also more dangerous: People may waste valuable work when changes conflict. In general, you use optimistic concurrency if you think people won’t find it too difficult to reconcile different versions, and when they’re working in close proximity. You can also have optimistic concurrency with notification: i.e., a checked-out object is not locked, but there is a process for notifying either party that someone else has it too.

extent of locks and transaction definition -- In a file-sharing system, consistency is maintained at a large-grained level by locking files whenever someone may be writing (or deleting) a file. In a relational database, operations are performed on sets of data (frequently one row at a time), and so data is generally locked by sets, tables, or rows.

Ideally you want to resolve conflicts without totally backing out one set of changes, and you want to avoid locking up an entire airplane while one person works on a single component. What changes are allowed together, and which ones conflict? Can Juan change the size of the wing while Alice changes the material? Maybe, but both those changes will have an effect on the weight of the aircraft, which affects required engine capacity.

In an object-oriented database, the system potentially has much more knowledge of what’s going on, and can exert smarter locking controls or execute trigger procedures when certain things occur -- handy features in light of the extended-time, extended-reach nature of object-oriented transactions. Objects (including collections of objects) are much more complex to define (and to lock) than simple sets. The database’s
knowledge of these structures ultimately will allow it to control integrity more finely than by simply locking out a large part of a database that might be affected by a transaction.

Semantic constraints -- where you want to lock a composite object, traverse a network of relationships (cf. referential integrity in a relational database), or where you want to lock only some attributes of an object -- will not be defined in a database as sold, but they can be defined by a reseller who tailors an object-oriented database for some specific applications, such as VLSI design, or by end-user schema designers. They can be explicitly defined by a developer when the database vendor provides the tools to do so (usually in the DDL/DML, and less likely to be found in the systems where "the database takes care of everything so we don't have to mess up your language").

Stanley Zdonik, a consultant to Object Design, and Andrea Skarra, both at Brown University, have done substantial research on this issue, but no one has yet found it practical to implement automatic support for such constraints. For now, vendors generally do or shortly will provide locking by object or by page, the physical location that may contain several objects, or only part of one. Most of these benefits and capabilities are still potential.

Ultimately, these issues can't be handled in general in a vendor's object-oriented database management system. They must be handled by schema designers and database creators, but vendors should provide the tools to support these options and context-dependent rules in the database, so that they will hold across applications and for all users.

- **tool suite** -- The actual database part of a database may be -- should be -- almost invisible to the people who use it. In the traditional world, databases are on their way to commodityhood (despite all the ads touting performance and unique capabilities); customers care more about the tools they can use. Savvy vendors are now selling or promoting the availability of tools: Tandem, for example, has lined up deals for tools from a number of ostensible competitors such as Oracle and RTI. Information Builders' Focus front-ends DB2. And in the pc world, Ansa/Borland, Microsoft, Ashton-Tate and Lotus (with all the enthusiasm of an investor) will be providing SQL-based front-ends to Sybase's SQL Server and to IBM's DB2 and OS/2 EE Database Manager. In the object-oriented world, much the same thing will happen. However, there's no simple insulating interface such as SQL which would provide an easy way to keep the database independent of the tools.

For users and application builders, there's need for graphical browsers (so the user can see the structure of an object and its interrelationships with other objects or an inheritance hierarchy), point-and-click development tools, forms and the like. As in other tools, the goal is that the user should be able to select from menus where possible. If the system knows a list of possible answers, it should supply them for the user to pick from rather than wait for him to type one in and then send a cryptic error message when he types it wrong.

In addition to query and reporting tools, schema design and tuning tools will help sophisticated users optimize performance using statis-
tics so that the use of clustering, concurrency, lock extents and other constraints can be optimized. There will also be languages and tools to define transactions and concurrency constraints as well as data structures.

Of the vendors listed here, Object Design is focusing the most heavily on tools. At the other end of the spectrum, Ontologic and Object-Sciences are focusing on providing platforms for other vendors' function-specific tools.

- ship date (promises, promises!) -- Servio Logic, Ontologic and Symbolics beat everyone into the market with their first generations. Each has around 50 installations (plus or minus 20), but only Symbolics can yet point to a successful, production use of its product (at Houston Power & Light, page 4) -- while MCC's Orion prototype has actually been used to do system configuration of customer image systems at Kodak.

The next round of entrants -- the Object-brothers (0-Design, 0-Sciences and 0-tivity), Hewlett-Packard, Juniper, Artemis and Altair -- are in effect starting with the second generation, having learned for free from the experiences of their predecessors. They also enter a market better prepared to receive them.

Juan: What happens to spaghetti code when you go object-oriented? Does that fix everything?

Alice (sigh): No, you just end up with ravioli code.
INSTANCES

We had originally planned to list all the companies and their databases in a table, but of course they are objects and don’t fit neatly into rows and columns (see page 10). They all inherit characteristics from the superclass of object-oriented databases, but it would be misleading to compare them closely when most are still unreleased. One vendor’s support of a feature is another’s non-interference with a user’s doing it himself, and one vendor’s Release 1.0 may be less finished than another vendor’s beta product: Ontologic, for example, is shipping Ontos in October, but many of the goodies will come in February. To summarize: Every vendor plans to support everything in some future release...but none has enough of a track record yet to earn full credence. The dates after the titles show our estimates of commercial ship dates, except where company dates are supplied in the text, and are subject to change, except for those that read 1988.

Altair/O2 -- A commercial consortium (1991)

GIP Altair5 is a French joint venture owned by a commercial software house, IN2 (49 percent); INRIA, the Institut National de Recherche en Informatiques et en Automatique (46 percent); and LRI, the Laboratoire de Recherche en Informatique of Université de Paris-Sud (5 percent), which decided to fund the $20-million, five-year "database for the 90s" research project in 1986. One third of the budget (without equity) comes from the European Community’s Eureka project, through funding it supplied to IN2. IN2 is the Continent’s biggest vendor of Pick applications and hardware (its own 68000 systems and Siemens equipment), with about $200 million in annual sales. It was recently acquired by Siemens, but the project is small-potatoes enough by Siemens standards to have been left alone.

Besides, the end is in sight. Leader François Bancilhon has just set his team of 30 developers (out of 45 people) to work on turning the current prototype into a product, O2, for beta release a year from now and commercial release in mid-1991. Altair combines the best of two worlds, with the pragmatism of its Pick background (a mini-based, friendly operating system that has its own relational database) and the finances and freedom to go out and hire 15 PhDs, including (for a year) WISS project leader Dave DeWitt and Dave Maier, another OODB wizard who consults for Object Design and Servio Logic. Bancilhon once worked with Orion’s Won Kim at MCC (on a predecessor project) and is using a modification of the University of Wisconsin’s WISS (under license) for the underlying storage system.

The initial prototype, V0, was built to be thrown away and was delivered late in 1987. The current prototype, V1, running, needs a rewrite to become a working product: "We will add performance and robustness, other things we know how to do now and didn’t know the first time," says Bancilhon. O2 will

---------

4. Strictly speaking, only classes "inherit," while instances "derive" their behavior and attributes from the classes they instantiate. But allow us this poetic indulgence.

5. The "GIP" stands for Groupement d'Interet Public, a reserved term under French law used to describe public-interest joint ventures.
run on Sun workstations, both as server and clients, and will have a runtime component that supports dumb terminals. However, in its first release it will still run only as a single-server system.

The system works with an extended C language, CO2, or BASIC O2, and an SQL-like query language. Development tools include debuggers, a graphical browser and a toolbox/class library of typical commercial objects such as dates, names, and units of measurement. O2 has an overall commercial flavor to it, starting with its support for BASIC and attention to integration with other languages, databases and application structures.

The system offers object-level pessimistic concurrency control, with some finer-grained concurrency for metadata; for example, it’s okay to add two methods to a class simultaneously, but not for two processes (applications or users) to update an instance simultaneously. The system has enough semantic information to know what’s allowable there, but it can’t really comprehend what a user is doing with the instances, and whether the updates might conflict. "With instances, we just do what a normal relational system would do," says Bancilhon.

Artemis/Orion II -- The commercial version (1990)

Artemis, a Control Data subsidiary, is the first company to attempt to make a product from MCC’s Orion technology (below). The unit has start-up funding from Control Data and was founded by Doug Barry, formerly CDC’s liaison to MCC and a research scientist specializing in database technology. Its first task, says Barry, is to turn Orion from a research prototype into a commercial product. "They left lots of simple things undone," he says, "because they weren’t interesting as research issues." However, they are niceties customers care about -- such as handling media failures as well as software errors, a nice interface (OSF’s Motif), and support for C (with function calls) as well as LISP.

The start-up has a few beta customers, and hopes to ship the commercial version next April. "We’ll push the flexibility it gets from being in LISP," especially dynamic schema modification, and will sell mostly to CAD/CAM customers, says Barry. The other focus for Barry and his two co-founders, also from Control Data, will be to generate powerful class libraries and to work with VARs, so that users needn’t see the LISP. In addition, Orion II will support all the requisite features: multiple inheritance, fully distributed operation on anything that runs Franz LISP, public and private databases (a useful feature in distributed systems, so that you can keep your own data undistributed), versions and so forth. Unlike the other not-yet-shipping vendors, Barry is willing to discuss pricing, which starts before discounts at $4000 per server, plus a $3000 LISP license from Franz.

MCC/Orion -- LISP is the good news and the bad news (research)

A research prototype, Orion first appeared (available to MCC shareholders only) in May of 1987, as a single-user system that nonetheless already had transaction management. A year later it graduated to a client/server architecture (one server/many clients). This fall, as Orion-2, it should work fully distributed, so that multiple users can make queries or transactions
Even non-profit ventures can be disappointing to their backers. MCC, created to foster US competitiveness against the foreign hordes, has turned out more like a slow-starting venture deal, with several management changes and shifts of direction and other digressions. But it has done some respectable R&D that hasn't made its way out into productive, commercial use. Not all of MCC's backers are capable of using its technology (for reasons of culture, finance and focus, among others), and the unit has failed to develop the broad base of support it hoped for. (For historical reasons, MCC stands for Microelectronics and Computer Technology Corporation.)

But under new CEO Grant Dove, and with new inside and outside marketers in place, this earliest of the current generation of consortia is about to go semi-commercial, and open the fruits of its work to anyone with the means and the incentive to spread it around. The organization is "unbundling" and offering its technology to outsiders on a sort of second-option basis once investors have had a crack at it on more favorable terms. Where investors once had to fund an entire division to the tune of several million dollars a year, they can now invest in individual projects.

This pilot program applies initially only to the Advanced Computing Technology Program (formerly Advanced Computer Architecture). The projects include those of the Object-Oriented and Distributed Systems Laboratory: Orion, and three follow-ons that will use Orion as a base: OBCD, object-oriented programming technology for concurrent systems; IOOS, Orion plus frame-based reasoning; and SHDB, Orion extended for heterogeneous operation and transparent access to foreign databases.

ACT Lab participants include Bellcore, Control Data, Digital Equipment, Kodak, Harris, NCR. DARPA, E-Systems, General Dynamics, Honeywell and Motorola have come in under the new structure, although none has opted for the object-oriented and distributed systems projects.

with objects (including voice and images) on multiple platforms. The system runs on either Suns or Symbolics machines, but servers can't be mixed to support a single database. Initial release of this version was July 1, but the final transfer to sponsors will come this fall.

It will also include multiple inheritance, collections, support for both private and shared databases, pessimistic concurrency with optimistic optional, and dynamic schema modification; the only lack is support for nested transactions. However, Orion has only a programmatic interface (no nifty graphical browsers, for example). The first version did, but it was based on Flavors and wasn't reimplemented when the project switched to Common LISP, says project manager Won Kim. The interface is not especially interesting in terms of research, and is being left as an exercise for commercial resellers such as Artemis, above.

Instead, Kim's next goal is to add reasoning into Orion, borrowing tools from MCC's AI Lab projects such as CYC (see Release 1.0, March 86-3) and Proteus, an expert system shell. This new project should take four years,
starting this winter and building on an existing loose interface between Proteus and Orion.

Another group is working on Orion as a front-end to a variety of databases, both supporting objects all the way down and providing transparent access to data held in relational databases, among others. The advantage of this would be to provide retrofit technology (see Release 1.0, 86-5) around existing databases and the data they already hold, and allow the data to be manipulated with jazzy new object-oriented tools. Theoretically, flexible, LISP-based Orion will be able to accommodate any existing data structure as a subset. Of course, there will be no way to avoid some overhead, but the ability to translate from one semantic model to another would be desirable. We expect it can be achieved to some degree, but something always gets lost in translation (cf. French poetry in German, or Japanese haiku in English).

Versions of Orion are currently in productive use at several MCC shareholders. Bellcore is using it to manage an Interactive document-generation tool called Quilt, and Control Data is using it for mechanical CAD and for its "General Data Library." Kodak is using Orion mostly to explore object-oriented databases and as "a feature-set [for others] to beat," says Fred Mellender of Kodak's Advanced Computing Research Lab. "MCC has given us a lot of value," he adds. "Commercial companies are taking less risk in this market because they can improve on the published work done at MCC." But Kodak is also using Orion in real-live work, as a back-end to MCC's Proteus expert system, in a system that configures optical disk information systems for customers. "Scores of installations" have been configured this way in the past year and a half, says Mellender. Other possible applications include a repository for software development (cf. IBM's Repository, page 25).

Hewlett-Packard/IRIS -- Not inherited from NewWave (1990)

Iris is a longstanding research effort at Hewlett-Packard which has just now moved (partially) into a commercial division of the company, the Data and Languages division. Although it has most of the standard features, Iris seems to have been influenced by the need to support other Hewlett-Packard products, specifically NewWave on the one hand and H-P's DBCore, the storage substrate that underlies H-P's relational databases, Image and Allbase. This relational underpinning is the big flaw in Iris, because it doesn't provide the clustering and optimization needed for high performance. For now, it is still a research prototype, more suitable for managing office automation objects and exploring object-oriented architectures, than an industrial-strength system that could manage the kinds of data encountered in H-P's scientific/engineering businesses. However, it does store object IDs, eliminating some of the joins and value-matching required to construct objects out of a vanilla relational database.

6 When a user or application requests an object, the "query translator" module translates that query into relational algebra -- in the case of the car, perhaps a set of items that have "car" in the "part-of" field, plus all the items that have those items in their "part-of" fields, and so forth. There might also be some other joins required to get the functions (behaviors or methods) applicable to each of these items. It could take a long time to get the whole car.

Release 1.0

22 September 1989
Further development on Iris is proceeding in two directions: one for support of H-P's traditional engineering customers, competing with the other companies discussed here, and one for support of NewWave, where it's more likely to meet competition from Oracle and, in combination with Office-Vision, from IBM.

With integrating tweaks, Iris will provide persistent storage for NewWave's Object Management Facility (see Release 1.0, 87-12). Basically, OMF maintains data on all DOS files and links between data and the appropriate applications, and is stored as a file that is loaded and maintained in memory and virtual memory. (Least-recently used data is kept on disk.) As NewWave users create, say, compound documents that consist of multiple kinds of data, the volume of DOS files that the system has to manage (and that the user never sees) can grow dramatically, necessitating some such system. Later on, when NewWave incorporates agents and a more deeply object-oriented architecture with fine-grained objects (instead of data file + application = object), the need for a robust object management system will increase.

For the engineering market, says development manager Bob Perreault, H-P is collecting feedback from early-exposure customers to determine how to strengthen Iris for production use. Specifically, the company plans to add clustering and other optimization techniques, either to DBCore or to a new storage manager, to enhance performance to a competitive level. But it also wants to find out which elements of the OODB feature set are most important. Because Iris has not yet been formally announced, there are no dates and no feature specifics, but we'd expect to see it out in some form in 1990.

Juniper Software/Persist -- One step at a time (1989 - Mac; 1990 - Sun)

Ontologic has already spawned two companies: Object Design (below) and Juniper Software, which will demonstrate its (initially) Mac-based, single-user system at OOPSLA. A multi-user Sun version is to follow.

The project was begun last November by Craig Damon right after he was laid off from Ontologic, where he had implemented Vbase along with Gordon Landis, now at Object Design. Damon was a co-founder of Ontologic and had been with its predecessor Mosaic almost from the beginning.

The two companies have similar long-term technical goals, although Object Design is more ambitious in its hopes to set a standard. Says Damon, "They'll have a fuller product, but we're closer to delivery." He's gotten this far with only his own work and half a dozen part-time technical people, most of them still employed elsewhere until he can get funding on the strength of the Mac version, which is basically a proof of concept that will sell for $1500. ("It's an extreme disadvantage to be on the Mac," Damon says, because most of the potential customers have Suns. But in the long run it may be a good way to find the next group of potential users and trade them up. They're the same kind of people who first used pcs, and ended up building mission-critical systems.)

Unlike the other vendors, Juniper is starting out with a low-end version targeted at exploratory users. It runs underneath MFW (Macintosh Programmer's Workbench) and looks deceptively simple with its user-friendly interface and a graphical browser.

Release 1.0

22 September 1989
Underneath there's an industrial-strength system struggling to get out. Persist already has user-specified clustering and other optimization techniques, schema modification, support for complex objects, and hooks for the addition of distributed objects and optimistic object-level concurrency control. Objects are made persistent by an argument to the constructor when an instance is created -- i.e., by an extension to whatever language is being used -- although a schema designer can also declare a class persistent by default. Thus there is no need for application developers to be aware of persistence, although they can manipulate it specifically if they wish to.

Future Sun-based versions will support multiple users, multiple inheritance, pessimistic concurrency, nested transactions and other niceties. The Sun version will go for $5000, Damon says.

Object Design/ObjectStore -- Running but not yet released (1990)

While people at the other Object... companies bring database or engineering backgrounds, Object Design's people come from the first-generation object-oriented database world: chief architect Dan Weinreb was the principal architect of Statice, and ceo Tom Atwood and chief designer Gordon Landis were major architects of Vbase. Data-model designer Jack Orenstein worked on CCA's Probe research project. Other employees came from the object-oriented programming world.

Their product ObjectStore will be on view in a private suite at OOPSLA. It will be operating, if not fully debugged, with queries and transactions across a network of Sun servers, a set of graphical tools and all the basics -- collections, multiple inheritance, schema designer, and graphical tools (already shown at SIGMOD). Corporate partners including Kodak will get production code to work with early next year, for feedback on features as well as testing, with commercial sales to follow late in the year.

Like the relational database vendors of today, Object Design is spending significant time and effort on its tools as a means of differentiation. But founder Tom Atwood's basic goal is to set the industry standard with ObjectStore's data manipulation/definition language, a set of extensions to C or C++ that are translated into C with a pre-processor. (A future version will translate into C++, but for now C gives greater portability because of the prevalence of C compilers.)

"Our obligation is not to hide under the skirts of C++. We've defined a data model, objects and relationships, and we've defined a DDL/DML language with a compiler to back it up," Atwood asserts. "The issue isn't extending C++; you've got to, one way or another, because C++ doesn't do the job. The issue is who is the standard? The languages may vary syntactically but will probably be similar semantically, and we're prepared to change our syntax if we have to [in response to market sentiment in favor of another standard]."

ObjectStore parameterizes the C++ "new" command (as is now allowed in C++ Release 2.0) to create persistent types, independent of any particular classes, so that persistence is orthogonal to type. From then on, persistence is invisible to users and most applications. The database designer determines whether clustering should be by class or by composite object or by some other arbitrary criterion, and can change this later without too
much difficulty. Although not necessarily in the first release, says database architect Dan Weinreb (also designer of Statice before he left Symbolics), Object Design will provide tools to generate performance metrics to guide schema designers in tuning systems by means of specified clustering characteristics and indexes.

Concurrency is handled at the page level -- not ideal from the point of view of tidiness, Weinreb acknowledges, but he predicts it will result in higher performance: Page-level locking limits overhead, and the cases when a locked page containing several objects would unnecessarily lock out a second user going after a different object are rare.

Object Design's corporate partners include Eastman Kodak's Boston Technology Center (as opposed to the Rochester lab where Mellender works). Assistant manager Bob Gordon is assessing ObjectStore as a potential back-end for high-end publishing applications. That would allow users to store text and images and formats and enable, say, the automatic use of a color photo correctly in both full-color and monochrome publications. In addition, Gordon points out, it would support legal and financial attributes, for tasks such as copyright credit and cost accounting. Overall, he notes, "Object Design is doing a good job of attention to the complex trade-offs between functionality and performance. Features aside, the issue is scalability. We're pretty glib about these things, but you're not going to see dBASE running American Airlines' Sabre system anytime soon." Within a year, he says, Kodak will have several more products in hand and will do a "re-audit" of all of them. Kodak and other corporate partners will get working beta systems this winter.

Objectivity/Objectivity/DB -- By engineers for engineers (1990)

Objectivity's Objectivity/DB remains the system most shrouded in mystery. The company consists of a team mostly out of the engineering world, plus a real-live manager in the person of businessman and venture capitalist Bob Field. "We like them because they're not object fanatics," says one potential customer.

"Calling us an object-oriented database company is like describing GM as a steel-and-vinyl company," says Field. For example, he adds, revealing one product feature, "Versioning is considered optional in an object-oriented database, but it's absolutely vital to an engineering environment. We're blending three technologies: object-oriented things, traditional database concerns -- currency, privacy, heterogeneity -- and a bag of tricks from engineering databases. People who do just two of those won't get the job done. Our tricks aren't Codd's twelve rules; they're not codified or patented. They tend to be speed-related..." All this will be implemented by staff with experience at Daisy, Cadence, Applicon, Prime and ATP. The first platforms will be Sun-3s and DEC MIPS boxes.

Objectivity's emphasis is at the back-end, which is what its CAD-oriented customers want, but it has just hired a user-interface expert (Phil Gust, who worked on the X Toolkit and user interfaces for groupware at H-P Labs) to build some end-user tools for the first release. (The timing of that is undisclosed, but figure at least a few months for the work to be done.)
A great believer in standards, Objectivity has just joined the Open Software Foundation, as well as the Workshop on Object-Oriented Design. However, Field doesn’t promise to use OSF Motif because he isn’t sure it will be a standard in the engineering world. Objectivity’s technical advisory board includes Mark Linton, author of Interviews, a public-domain C++-based alternative to Motif. In the end, says Field, "Our task is to manage the data, not to make it look good."

Object-Sciences/Object-* -- Database influence (1990)

Unlike Objectivity down the road in Menlo Park, Object-Sciences has drawn most of its people from the database world -- Informix, as well as RTI and Oracle, H-P and MCC’s Orion project. Co-founder Mike Seashols, formerly vp of sales and marketing both at Relational Technology (85-88) and at Oracle (82-85), has signed on as full-time ceo. He had worked with founder Kee Ong, who was responsible for the overall architecture of RTI’s Ingres and before that for H-P’s Image database.

"Oracle didn’t invent anything new," asserts Ong. "It just took [IBM’s] System R and implemented it." In the same way, he’s not looking to make any breakthroughs, but rather to deliver a working, high-performance system based on the best of existing database and object-oriented systems research. (The staff includes not only a host of database types, but also Hong-Tai Chou, who wrote much of the Wisconsin Storage System, or WISS, under the guidance of Dave DeWitt, and also worked on Orion at MCC.) But, notes Ong, based on his experience at earnest, high-tech RTI, overshadowed by high-powered marketing-oriented Oracle, "there’s also sheer marketing." Finally, there’s a cache of $3.5 million raised from TA Associates and other sources.

The planned ship date is "early 1990"; if Object-Sciences beats Object Design it gets to keep the name of Object-Store for its storage component, says Seashols. (We wouldn’t be surprised to see the whole product line called Object-Star, in keeping with relational tradition dating back to IBM’s R*, the prototype/predecessor to DB2.) Object-Sciences will focus on a high-performance portable engine, leaving front-ends and configuration details to OEMs and VARs. The system is designed to work in almost any environment, with resellers adding to the hooks Object-Sciences will provide. This is the initial strategy taken by most of the relational database companies, although they are now turning to tools and applications as the market becomes more cutthroat and their products become commodities.

The product is written in C and C++ and comes in five modules: Object-Base, the database model, and Object-Store, a storage system based on the ideas in WISS; plus a C++ pre-processor and database schema generator; a C++ interface; and Object-Navigator, a browsing tool. Object-Base uses Object-Store as its storage system by default, but it can also load instance data from an SQL database, providing easy integration with some existing data. You can also use Object-Store alone as the back-end to a single application that has large amounts of complex data but doesn’t need the concurrency, transaction management and manipulation capabilities of a database. Object-Navigator is a graphical front-end with a browser that enables users to navigate through a network of objects and visualize a schema for design, debugging or demonstration.
In short, it uses an extended C++ that can build a database directly from C++ source code. Concurrency works at the object level, but objects must be explicitly checked in and out, rather than used seamlessly.

The first version will run on Suns, with other UNIX workstations to follow, as well as VAXen with Ultrix and VMS, and eventually OS/2. Object-Base allows multiple inheritance, nested, distributed transactions and queries, collections, versions, and public and private databases.

Ontologic/Ontos -- Second time around (1988/89)

Ontologic is by far the oldest pure play in the object-oriented database business. It began life in 1982 as a hardware company (see Release 1.0, 87-8), Mosaic Technologies, and floundered. In 1985 it decided to focus on its object-oriented database project, headed by Tom Atwood. (Atwood left in 1987 and founded Object Design, above, early in 1988.) Ontologic first shipped Vbase in 1988, but it never quite hit the mark. Problems included the product's nonstandard languages, Type Definition Language and C Object Processor, a single-user-only implementation, and a lack of tools aggravated by the nonstandard language which meant Ontologic had to do everything itself. "Ontologic felt they got bitten by making changes to a standard language," says one customer.

Now the company has regrouped and is about to launch a C++-based rewrite of the product, called Ontos. Although outsiders have been skeptical of the company's chances, it has gained substantial credibility (and additional funding of $1 million from Kleiner Perkins and The Phoenix Partners) based on a contract from Index Technology, which is building a 1991 version of its Excelerator CASE tool suite around Ontos. "We were all dressed up with no one to go to the party with" except Ontologic, says Jerry Katzky, vp software development at Index. That mutual dependence virtually ensures Ontologic's near-term survival, although an important part of Index's decision was based on the availability of a C++ version and the portability it promises in extremis. With a shrunken staff of 20 (from 43) and revenues of about $2 million in 1989, about half from Index and half from 25 other customers, Ontologic is now stabilized and plans to ship the first release of Ontos in September.

Index has two beta versions of Ontos in-house -- one for UNIX on Suns, and one for OS/2. The company is pretty happy with the product -- and knows that Ontologic will listen hard to its suggestions. Index makes a point of Excelerator's openness, allowing other vendors to hook into its tools and data dictionary. With Ontos underneath, most of them will still want to go through the Excelerator interface, with Ontos invisible underneath. They can create objects with the Index tools, but they won't ever have to know how they're stored.

Other customers will get Ontos for Suns and Apollos in October, and for OS/2 in December. About 25 Vbase customers who were paying maintenance will get it free; 15 new customers will pay $15,000 each. Aside from its C++ sup-

----------

7 ...or no wardrobe, to use the metaphor more precisely.

Release 1.0

22 September 1989
port, Ontos will support multiple users and will even run on multiple, but homogeneous, servers. It supports collections and pessimistic or optimistic concurrency control, and version management. A graphical browser will be part of the February release. A Mac client for Sun or Apollo servers will be available in December.

Servio Logic/GemStone -- Persistent company (1988)

GemStone is one of the few systems actively on the market, and has evolved considerably on its path to its current version, Release 1.5. First sold commercially in early 1988, the system should generate five times the revenues this year as last year, although both figures are undisclosed, the company says. The number of units isn’t up as much, since customers are now buying higher-end systems (and five or six customers have even bought two or more). Units are never comparable, but we figure that Servio Logic is at least a leader in systems sold, with between 50 and 70, to Symbolics’ 60 and Ontologic’s 40.

However, none of the customers is yet deploying the system to do production work, admits Servio Logic data systems general manager Mike Connell. They are still working on research or development, even one unidentified company that has bought five copies for four different projects. As noted, Connell has seen a shift from internal CASE efforts at telecommunications and industrial customers to government and aerospace companies this year -- a shift he attributes in part to traditional dbms vendors who now promise objects too. The government and aerospace customers know precisely what they need, which is not tables of objects and links, and they need it now, not in a couple of years. Customers include NOAA (the National Oceanic and Atmospheric Administration), which will use it to manipulate maps, satellite scans and other information, and the Air Force’s Advanced Launch System group.

Performance has improved by a factor of 8 to 10 since initial launch in Release 1.5, and the range of platforms has broadened to include Suns, RTs and VAXen (ULtrix and VMS, DECstation coming). Clients include all of the above; and front-ends run on PCs and Macs. GemStone uses a different model from the traditional client-server architecture, offering distributed processing at user workstations, although the data must sit on a single server. Instead of potentially creating a processing bottleneck on the server, each local unit (Gem) directly reads the central database (Stone) for data which it then manipulates locally. This has the potential for creating data transfer bottlenecks, but Servio’s tests have shown that this approach generally produces better performance than central processing, although not as good as fully distributed data and processing (which is much harder to accomplish). Stone manages locks and allocates object IDs, but the Gems do the rest of the work. This is akin to a relational database architecture where databases might download two or more tables and then do a join or some other query locally.

The biggest knock against GemStone has been its proprietary DML/DDL, Opal, and its Smalltalk flavor. There is a remote procedure call interface from C and C++ (or any other language that can make a foreign C function call), but the company is working on tighter integration by the first quarter of 1990, so that you can define your objects in C++ and the tool will automatically

Release 1.0 22 September 1989
instantiate them in GemStone. Servio also has hooks for access to SQL databases, and to Neuron Data's Nexpert Object.

As an indication of its flavor, GemStone's documentation uses pictures of compound documents where most of the other vendors would use exploded engine diagrams or VLSI layouts (Objectivity). Concurrency is optimistic by default; but a user can optionally lock any or all objects pessimistically ("hybrid," if other objects in a single same transaction are left unlocked). This operates either at the object level or at a collection of objects predefined in the database or by an application. Prices range from $12,000 to $92,000.

Symbolics/Statice -- The power of LISP (1988)

Statice was one of the first object-oriented databases to be commercially available, since last year -- but not very... For practical purposes, its market was limited to Symbolics users, and most of the development team left during Symbolics' period of perestroika. Symbolics management had more pressing matters than Statice, so they kept the product on the market without a great deal of resources behind it. Regardless, more than 40 customers have bought some 60 units and have liked it enough to make enhancement suggestions (version management, multiple servers) that Symbolics is working to implement. Now that Symbolics is profitable again, it has decided to sink a little more investment into Statice, an appealing entrant in a hot market, and will appoint a product manager.

The product is LISP-based and limited to a single server, but it is the epitome of object-oriented flavor, with support for collections, multiple inheritance, dynamic schema modification, default optimistic concurrency control at the page level, with pessimistic optional. "We recommend to our users to keep transactions short," acknowledges Symbolics technical consultant Neal Feinberg. Statice costs $10,000 per seat or $50,000 per site. It integrates extremely well with the rest of Symbolics' Genera development environment -- but not so well with the rest of the world, C++ in particular. However, the world of Symbolics itself is growing apace, as Symbolics concentrates on selling add-in Ivory boards (for Macs, Suns etc.) and less on whole systems. In the long run Symbolics will probably shift its emphasis to software and port its products to standard environments. At that point, says vp technology Howie Shrobe, it might well be a fundamental part of Symbolics' strategy.

Customers include Alcoa, which is just starting to use Statice in a documentation system called EGADS (Electronic Guidance And Documentation System). It manages information about the equipment and processes in a cold-steel rolling mill: When an alarm is triggered, it helps the user at a Mac figure out what went wrong. Ultimately, says project manager Katherine Herring, the company hopes to hook it up to an expert system for diagnostics. The benefit, of course, is that Statice can serve as a back-end to a variety of applications. That same benefit is also key in the Houston Power & Light implementation, described above on page 4.
IBM'S NEW REPOSITORY: OBJECTS FOR DB2

One fine potential use of an object-oriented databases is as a repository for software development, which could store requirements, specifications, code elements, data definitions, documentation and other design artifacts, and all the relationships among them. It would also maintain versions and configurations, and mediate conflicts among users and applications.

However, given the long leadtimes for developing such a product, the aversion of the market to anything new, issues of scale, and IBM's own predilection for DB2, IBM is implementing such a product as an object-oriented application on top of its relational database, DB2. IBM has spent considerable effort to improve the performance of DB2, and the important thing is to get the system out there. IBM's eagerness to help train and field a force of 20,000 people to promote the use of CASE tools matters far more than the precise performance of the repository they will use.

The business CASE

There are many reasons not to adopt CASE, starting with inertia. But market fragmentation is another big factor: There didn't seem to be any truly safe choice. IBM's Repository should be a real boost to the CASE market, precisely because it is a market (COBOL-, mainframe-, dp-oriented) that still looks to IBM to set the standards in a world that elsewhere pays increasing lip service to open systems. IBM, once again, has a chance of establishing a solid platform on which other vendors -- Bachman, Index and KnowledgeWare for starters -- can innovate. The mere presence of the Repository makes all the third-party choices look safer.

How much standardization will the Repository bring to the party? The Repository is more than just a passive storage mechanism; it's also an extendible specification for how design elements must be represented. It provides formats and standard data requirements (call them classes) for a variety of software design elements. Third-party vendors will have to adopt these standards. The result is that it will be easier for users to apply different tools to the same design elements, reducing dependence on any one vendor (except IBM) and increasing the range of choices. Index's Ontos-based repository is an alternative to the IBM repository, but without IBM's clout. Index will appropriately position it as a waystation, where you can store your designs in a more flexible format between the workstation and the mainframe. But some customers may find the Index product sufficient by itself.

You have to bend your mind a little to keep this straight: The elements the Repository holds may be treated as objects by the CASE tools, to a greater or lesser extent, but they aren't (generally) "object-oriented" code themselves: They're code routines, diagrams, data definitions, comments and other inventory used by the software development process. The end result is whatever the tools build, and for now, in the CASE world, that's mostly host-based COBOL applications.

In the long run, there's nothing that limits the Repository (due 1990) to holding COBOL-oriented objects; it could just as easily be extended to hold C++ types or other new-age stuff. Just like the other third-party vendors supporting the Repository, object-oriented programming environments would have to create the proper function calls or language extensions to automatically define objects according to Repository protocols.

Release 1.0

22 September 1989
RESOURCES & PHONE NUMBERS

Francois Bancilhon, Altair, 011 (331) 396 35511
Doug Barry, Artemis, (612) 851-3155
Mitch Perlitch, Digital Equipment, (508) 467-7475
Dan Fishman, Hewlett-Packard Labs, (415) 857-1501
Bob Perreault, Hewlett-Packard, (408) 447-5134
Jerry Katzky, Terry Landers, Index Technology, (617) 494-8200
Craig Damon, Juniper Software, (508) 250-1097
Won Kim, Steve Maysonave, MCC, (512) 343-0978
Tom Atwood, Dan Weinreb, Object Design, (617) 270-9797
Bob Field, Drew Wade, Objectivity, (415) 688-8000
Kee Ong, Mike Seashols, Object-Sciences, (415) 325-2300
Si Lyle, Ontologic, (617) 272-7110
Jacob Stein, Mike Connell, Servio Logic, (503) 629-8383
Jackie Kovo, John Watkins, Hovie Shrobe, Symbolics, (617) 221-1000
Burt Rubenstein, Index, coordinator of Workshop on object-oriented Design, (617) 494-8200

For further reading:


"The Object-Oriented Database System Manifesto," by Malcolm Atkinson, François Bancilhon, David DeWitt, Klaus Dittrich, David Maier and Stanley Zdonik, a draft circulating in the usual channels, July 1989; to be presented in Kyoto December 4 (see calendar).

COMING SOON

- Lotus and Sybase.
- CompuServe, Prodigy, MCI Mail, USENET, Internet, et al. The only way we know to get around to getting online is to commit ourselves to writing about them...
- Network navigation.
- Transaction processing.
- And much more... (If you know of any good examples of the categories listed above, please let us know.)

Forum invitations will be mailed early in October.

Release 1.0 22 September 1989
**RELEASe 1.0 CALENDAR**

October 1-4  
*ADAPSO Management Conference - Orlando. Mingle with your peers (and Disneyworld's nearby just in case). With Alan Kay (Apple) and Robert Weissman (D&B), among others. Contact: Sheila Wakefield, (703) 522-5055.

October 1-4  
**Alex. Brown Computer Services Seminar - Baltimore. The tenth annual...** Contact: Rivka Hawk or Ellen Kempler, (301) 727-1700.

October 2-5  

October 2-6  
*OOPSLA - New Orleans. Sponsored by ACM/SIGPLAN. Come meet your fellow objects and share procedures. Send a message to Carole Mann, (407) 628-3602.

October 2-6  

October 2-6  
Interop 89 - San Jose. Interoperability made tangible, with tutorials, discussions, product demos and pitches, and speeches by Doug Engelbart and Vint Cerf of Corporation for National Research Initiatives. Sponsored by Advanced Computing Environments. Call Mark Belinsky, (415) 941-3399.

October 3-5  
PC Expo - Chicago. Sponsored by PC Expo. Keynote by Barry Kotar of Covia. Contact: Steven Faher, (800) 444-EXPO or (201) 569-8542.

October 3-5  
Open Systems Initiative - San Francisco. Sponsored by Open Systems Advisor; managed by ExpoConsul. Keynote by Geoff Morris, X/Open. Contact: Steven Faher, (800) 444-EXPO or (201) 569-8542.

October 5  
The R&D gold mine - Palo Alto. Sponsored by Regis McKenna. "Improve your effectiveness in using nonprofit research organizations as a source of new products and technologies." Contact: Elizabeth Batson at (415) 857-9388.

October 5-6  
Electronic messaging '89 - Chicago. Sponsored by Electronic Mail Association. With Warren Prince, Tymnet; Mike Zisman, Soft·Switch; others. EDI, X.400, and connected topics. Contact EMA at: telephone, (703) 522-7111; fax, (703) 528-4251; AT&T Mail, !EMA; Dialcom, 63:PRD003; EasyLink, 62886257; iNet, ema.association; CompuServe, 70007,2377; Envoy 100, EMA; GEnie, EMA; On Tyme, EMA.SUP; MCI Mail, EMA/2544290; Telemail, [ema/associates] mail/usa.

October 6  
Alliance Japan - San Francisco. Sponsored by Computerworld. "An action seminar for high-technology executives."

Release 1.0  
22 September 1989
With Mitchell Kapor and Vinod Khosla, among others. Contact: Jean-Pierre Nussbaumer at (415) 863-5074.

October 9-12
ShowCASE IV - St. Louis. "Integrating automated development environments." Tutorials and speeches, with Ed Yourdon, Vaughan Merlyn, Larry Constantine, Capers Jones, Sam Holcombe, among others. Sponsored by the CASE Studies Consortium. Call Donna Skaggs or Kimberly Yourick at (314) 889-4556.

October 10
*Massachusetts Computer Software Council fall meeting - Newton, MA. With a panel on "Object-oriented Everything" moderated by Esther Dyson and featuring Tom Atwood, Object Design; Tom Malone, MIT; and Burt Rubenstein, Index Technology. Call Joyce Plotkin, (617) 437-0600.

October 10-13

October 11-13
Strategic directions in computing research - Washington, DC. Sponsored by ACM. Call Donna Baglio, (212) 869-7440.

October 11-14

October 13-15
The Hackers Conference 5.0 - Saratoga, CA. Sponsored by itself, with help from friends. Call Glenn Tenney, (415) 574-0546.

October 14-21
International Computer Forum - Venice, Italy (the real one). Sponsored by Boston Computer Society. Seminars on desktop publishing, support strategies, computers in investing, etc. Call Beverly Kleiman, (617) 367-8080.

October 16-19
EDUCOM - Ann Arbor, MI. Keynotes: John Akers and Arno Penzias. Call Joan Davis, (609) 520-3340.

October 16-19

October 16-19
Scan-Tech 89 - San Jose. Sponsored by Automatic Identification Manufacturers. On beyond retail bar codes, including integration with EDI, tracking materials in offices, etc. Scan-Talk: Munster punster (and Wall Street Week host) Lou Rukeyser. Call Bill Hakanson, (412) 963-8588 or (800) 338-0206.

October 18
Rod Canion at New York PC User Group - New York City. Call David Hoffman, (212) 674-2632, or (212) 533-NYPC.

October 20-21
CPSR annual meeting - Washington, DC. Computer security and viruses, Federal support for R&D, and other timely

Release 1.0

22 September 1989


October 23-26 Hammer Forum - Boston. "Reengineering the corporation: Information technology and business process redesign." With Allen Loren, Apple; Paul Chapman, Rank Xerox; and Michael Hammer himself. Contact: Pam Davis, (617) 354-5555.

October 25-27 SGML '89 - Atlanta, GA. Chaired by SoftQuad's Yuri Rubinsky; sponsored by Graphic Communications Association. SGML will be to text what SQL is to data; find out about it early. Contact: Norman Scharpf, (703) 841-8160.

October 26 The R&D gold mine - Ann Arbor. Sponsored by Regis McKenna. "Improving your effectiveness in using nonprofit research organizations as a source of new products and technologies." Contact: Elizabeth Batson at (415) 857-9388.


October 29-November 1 Network systems forum - San Jose. Sponsored by 3Com. Call Cheryl Soderberg, (800) NET-3Com or (408) 562-6400.


November 1-3 *UNIX expo - New York City. Keynote by noted UNIX fan Ken Olsen of DEC; also speaking: Bill Joy and John White. Managed by National Expositions Co. Contact: Roger Halligan or Heidi Dethloff, (312) 332-4650 or (212) 391-9111.

November 5-10 Hypertext '89/SIGDOC 89 - Pittsburgh, PA. Much larger than the first, wonderful hypertext conference in late '87. Hypertext is the first three days; SIGDOC the last three. Sponsored by ACM. Contact: Elise Yoder, (412) 327-8181, or Nina Wishbow, (412) 323-2600. (How about a joint standards committee on the use of apostrophes?)


Release 1.0 22 September 1989
November 13-15  UIST - Williamsburg, VA. Symposium on user interface software and technology, sponsored by ACM SIGGRAPH and SIGCHI. Contact: John Sibert, (202) 994-4953.

November 13-15  16th annual computer security conference - Atlanta, GA. Yes, someone's been worried about this for a long time. Now that you are too, come hear FBI Director William Sessions, Cornell Provost Robert Barker, our own industry's Stewart Brand, Hank Jones, others. Sponsored by Computer Security Institute. Call John O'Mara, (508) 393-2600.

November 13-17  *Comdex - Las Vegas. Also including MACdex. Contact: Jane Wemyss at (617) 449-6600 or (800) 325-3330.

November 13-17  Supercomputing '89 - Reno, NV. Conveniently located near Comdex, if you tire of small computers and big hoopla. Contact: Ron Bailey, (415) 694-4500.


December 4-6  *First international conference on object-oriented and deductive databases - Kyoto. Sponsored by IEEE, MCC, many others. Contact: Professor Kiyoshi Agusa, 011 81 (75) 256-1677, or Won Kim at MCC, (512) 338-3439.


1990


January 28-31  **EDventure Holdings PC (Platforms for Computing) Forum - Tucson, AZ. Sponsored by us! New speakers will include Danny Hillis, Thinking Machines; Mike Slater, Microprocessor Report; Rod Canion is returning. Note that it's earlier this year. Contact: Daphne Kis, (212) 758-3434. Invitations will be mailed early in October.

Release 1.0 22 September 1989
February 6-9  *Software development '90 - Oakland, CA. Solid information from practitioners and luminaries: Larry Constantine, Ed Yourdon, Ken Ozz, Bill Gates, Philippe Kahn, et al. Spon- 
sored by Miller-Freeman. Call Cheryl How, (415) 995-2471.

February 20-22  Computer science conference - Washington, DC. "Coopera-
tion" among processing units, technologies, disciplines. 
Sponsored by ACM. Call Barbara Kyriakakis, (703) 323-2318.

Sponsored by Pacific Bell Directory. A multi-personality 
event with John Sculley, Alan Kay, Jaron Lanier, Herbie 
Hancock, Kenny Rankin, Ted Nelson. Contact: Mike Whit-
acre, (213) 832-9396.

March 5-9  *Seybold Seminars '90 - Boston. ...moves east. Call Kevin 
Howard, (213) 457-5850.

March 26-30  *International conference on extending database technology 
- Venice, Italy. Sponsored by IEEE Computer Society. Call 
Michael Brodie, (617) 466-2256.

April 1-4  *SIGCHI - Seattle. "Empowering people" with better inter-
faces to functional systems. With a panel on computer-
 aided conflict with Tom Malone and Esther Dyson, others. 
Sponsored by ACM. Call Toni MacHaffie, (503) 591-1981.

April 10-13  *Macworld - San Francisco. Later this year. Call Peggy 
Kilburn, (617) 326-9955.

April 23-26  *First international conference on systems integration 
- Morristown, NJ. Sponsored by ACM and IEEE groups. Call 
Peter Ng, (201) 596-3387.

April 25-27  *Conference on office automation systems - Cambridge, MA. 
Sponsored by ACM and IEEE groups. Call Joan Staunton, 
(212) 869-7440, or Robert Allen, (201) 829-4315.

April 27  *Computer Bowl - Boston. Sponsored by the Computer Museum. 
Call Kate Jose, (617) 426-2800.

May 1-3  *Second annual conference on innovative applications of 
artificial intelligence - Washington, DC. Sponsored by 
AAAI. Contact: Claudia Mazzetti, (415) 328-3123.

May 4-5  First conference on cyberspace - Austin, TX. Sponsored by 
some interested citizens, with Howard Rheingold and William 

Please let us know about any other events we should include. 
-- Denise DuBois

*The asterisks indicate events we plan to attend. Lack of an asterisk is no 
indication of lack of merit.
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 until December 31, 1989.* Payment must be enclosed. Multiple-copy rates on request. Satisfaction guaranteed or your money back.

<table>
<thead>
<tr>
<th>Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td></td>
</tr>
<tr>
<td>Company</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td></td>
</tr>
<tr>
<td>City</td>
<td>State               Zip</td>
</tr>
<tr>
<td>Telephone</td>
<td></td>
</tr>
<tr>
<td>How did you hear about Release 1.0?</td>
<td></td>
</tr>
</tbody>
</table>

*Effective January 1, 1990, the subscription rate will be $495 in the U.S. and Canada, and $575 overseas.

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.

Daphne Kis  
Associate Publisher

* Release 1.0  
22 September 1989