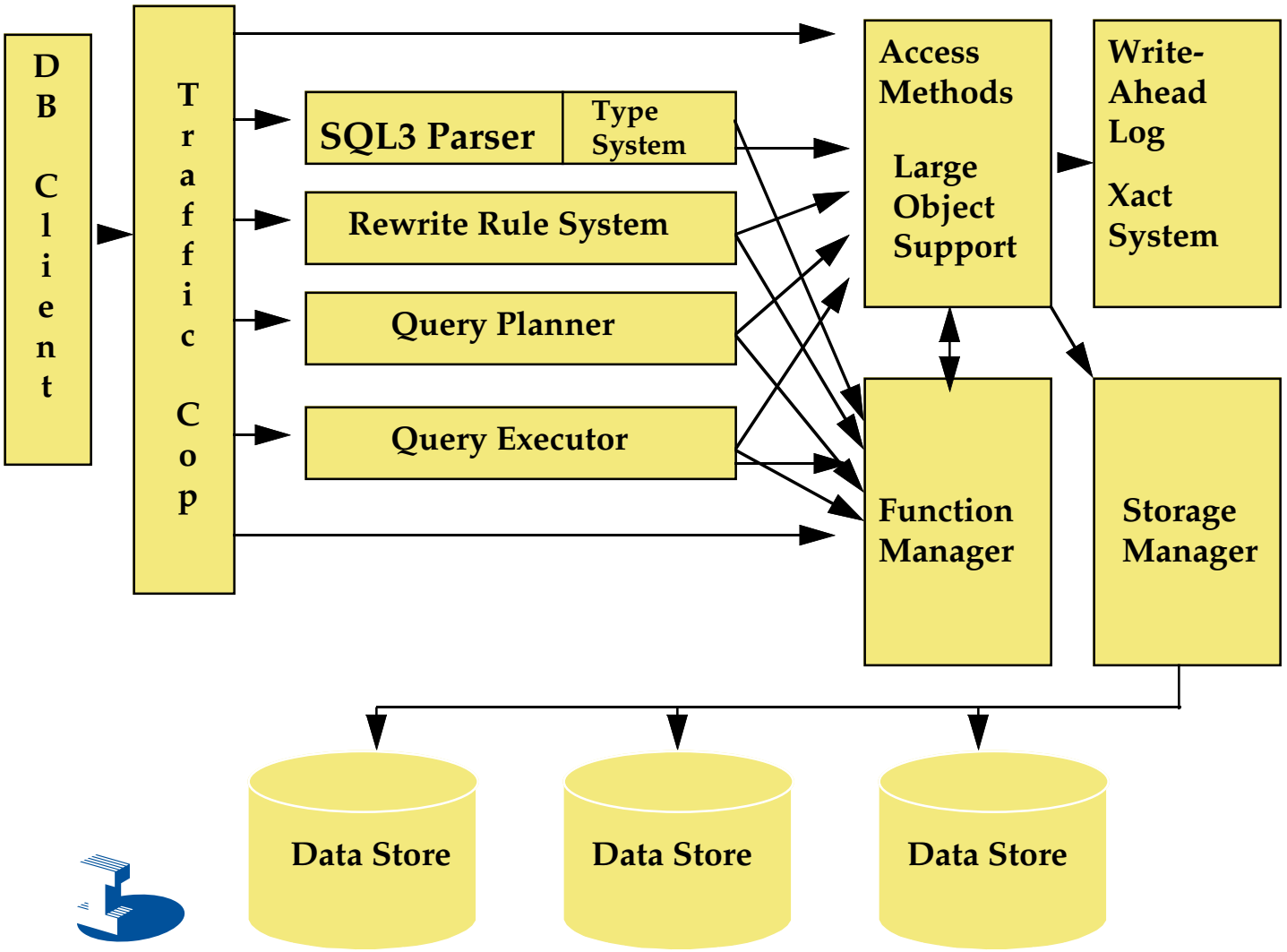
Presentation to Silicon Graphics March 3, 1994

ILLUSTRA

Purpose

- Explanation of Illustra Architecture
- What Makes it the DBMS for MM Data
- Not a Complete Tutorial on all Illustra
- Features

Architecture Overview



Objects in Illustra

- Objects are stored as records
- Tables are containers for objects
- Addressable in a variety of ways
  - By OID
  - By Content

- SQL Parser
  - Includes SQL3 Constructs
- Type System

Types and Functions

- Type Creation
  - Base Types
  - Composite Types
  - Constructed Types
  - Large Objects
  - Casts
- Functions
  - SQL Functions
  - C Functions - dynamic loading - client execution
  - Aggregates
  - Function resolution
  - Late binding

Types and Functions

- Table Driven
- All Types and Functions are “Equal”
- System Information is Cached
  - System Caches
  - Query Tree
  - Access Methods

Tables for Management of Types and Functions

FUNCTIONS

name, owner, language, trusted, varian, rettype,

argtypes, byte_pct, perbyte_cpu, percall_cpu, outin_ration, select_func, negator, commutator, binary, source

TYPES

name, length, class, byvalue, constructor, subtype, bound, parameter, input, output, columns, statcollect, opclass

Tables for Management of Types and Functions (cont)

CASTS

argument, result, function, isimplicit

OPERATORS

nargs, name, function

AGGREGATES

name, init, iter, final, returntype, argtypes

Type Creation

- Base Type

  create type point (internallength = 16,
  input = point_in, output = point_out, ...);

- Composite Type

  create type address (street char(20), City
  char(20), state char(2));

  create table mytab of new type mytype (c1
  text, c2 integer, c3 address, c4 point)

Type Creation (cont)

- Inheritance

```
create type positive (input = pos_input)
under integer

create type foreign (country char(10))
under address;
```

Type Creation (cont)

- Constructed Types

```
create table test(a arrayof(integer), s
    setof(interval hour to minute), f float[10]);

create distinct type darray arrayof(double
    precision);
```

- Large Objects

  Internal
  External

```
create type image under large_object;
create type document under external_file;
```

  Procedural interface: open/read/write/seek

Functions

- C-functions

  Dynamically loaded into server

  or

  Run in client application - must register function

  create function myfunc(integer, mytype)
  returns mytype as external name '/path/file.so'

Functions

- SQL functions

  One or more SQL statements

  ```
  create function sqlfunc(character) returns
      setof(emp) as select * from emp where
  name = $1;
  ```

- User definable casts

  ```
  create cast (mytype as int using myfunc);
  ```

- Aggregates

Function Resolution

- SQL3 defines a “best fit” algorithm
- Only considers functions which have the best match for all parameters

  1) Exact Match
  2) Direct Inheritance
  3) System Cast
  4) User Cast

Late Binding

- Functions may be declared LATE
- Resolved over the table hierarchy only

Access Methods

Illustra Access Methods

Two kinds of access methods in Illustra:

- All user data, system catalogs stored in heap tables. This is the primary data store.
- All indices (btree, rtree) are secondary.

Defined access method interface (AMI) allows:

- Three-star wizards to define new indexed access methods
- Programmers to call access method interface routines without worrying about the storage structure supported by the AM.

Illustra Access Methods (cont)

Interface is supported at three levels:

1. For use internally by the Illustra server
2. For use from dynamically-loaded backend functions
3. For use directly from the client program, via LibMi

AMI Routines

These are the interfaces required to write a new indexed access method. Some have defaults that can be used if the AM requires no special processing. Every indexed AM supports these calls.

- create, destroy index
- open, close index
- insert, delete record
- begin a scan, end a scan, reposition a scan to the beginning (scan creation takes a qualification as an argument)
- mark the current position in a scan, restore a scan to the marked postion

Hooks for AM-Specific Support Routines

Access methods may require special-purpose support routines for each operator class:

- Btree requires a comparison function
- Rtree requires routines to compute k-dimensional rectangle intersection, union, and size

Hooks for AM-Specific Support Routines (cont)

Illustra provides the "amsupport" table in which support routines can be stored.

Amsupport is loaded into memory, attached to the indexed relation descriptor (like a file descriptor) when the relation is open.

Simple interface for calling a routine from the access method.

Operator Classes

An operator class binds a set of type-specific operators (eg, integer comparison, floating point comparison) to the strategies supported by an access method.

If the user defines a new type, he can define the operator class bindings for the type and use existing access methods on the new type.

Storage Manager

The Storage Manager Switch

Like the device switches used by most Unix kernel implementations: A table with thirty-one entries that supports all the required operations on any storage device we manage.

A large object or table can be located on any storage manager, but on only one. Striping, other fancy services implemented inside the storage manager.

The Storage Manager Switch (cont)

The storage manager interface routines handle:

- Startup, shutdown
- Operations on relations
- Operations on large objects
- Special processing required at transaction boundaries (eg, flush dirty data pages to stable storage)
- Two interfaces to support write-ahead logging

Operations on Large Objects

- open, close
- unlink (by name or by open large object descriptor)
- seek, tell
- read, write
- synchronous write
- stat
- map into virtual memory, unmap
- return number of 8K pages in a large object (support for recovery)

Direct Access Interface

- Provides a direct interface to the Access Method level of the system
- This is a procedural interface:
  - Open Table/Index
  - Begin Scan
  - Fetch
  - Insert
  - Update
  - Delete
  - Execute -- Can execute user defined function

Conclusion

- Extension to MM is natural
  - New storage manager, access methods not too hard
  - New types easy
- This is the DBMS for MM