1. COMPILER PASSES

There are currently five subtasks or phases in the compilation of an LDM specification. The subtasks are organized in a sequence of five passes.

Pass 1: (initial parse / semantic check) The LDM source is parsed, and an initial check is made for semantic correctness. The internal data structures produced by this pass are described in the next section.

Pass 2: (query optimization) Query optimization translates queries expressed with high-level non-procedural constructs of an access specification language (ASL) into low-level procedural constructs of the same language. There are three steps: normalization of query bodies, join order selection and conjunct order selection.

Pass 3: (remaining physical design) There are four issues in object representation: index selection, object store management, object identification, and property representation. At present, the user must specify a selection of indices and store managers. This pass accomplishes the remaining physical design. First, the method of object identification is determined for each class. There are five possibilities, with the first reserved for built-in system classes. The remaining four depend on the store type declared for a class, and on use of the identity assignment operation. Property representation mainly involves record type generation. Record fields are generated for user-defined properties and indices.

Pass 4: (transaction compilation) Transaction bodies are compiled into procedures that explicitly encode operations for index and store maintenance.

Pass 5: (PDM source generation) A PDM source file is generated for the original LDM specification. PDM code for queries is in the form of a low-level access strategy language. For transactions, the PDM code is roughly at the level of C or Pascal, with the addition of generic index update and store operations. For object representation, PDM code is roughly at the level of Pascal record type declarations.

2. INTERNAL DATA STRUCTURES

Executing "PassOne" of the LDM compiler results in an assignment to the property list of atoms (that correspond to names of classes, properties, queries, transactions, indices and stores) parse information of input source. A number of global variables are also assigned to initial values. A list of these global variables together with an indication of the form of their values is given in Table 1 below.

Global Variable	**Value**
Schema	<SchemaName>
Classes	(<ClassName>...)
Properties	(<PropName>...)
Queries	(<QueryName>...)
Transactions	(<TransName>...)
QueryOrTransName	<QueryOrTransName>
Indices	(<IndexName>...)
Stores	(<StoreName>...)

Table 1. global variables

The form of the property list entries together with their indicators is given in Table 2. The following BNF-like grammar rules indicate the parse formats mentioned in the above tables. Note that "..." indicates one or more occurrences of the preceeding construct.

	Variable	Value
Classes	Class?	t
	SupClasses	(<ClassName>...)
	SupClasses*	(<ClassName>...)
	SubClasses	(<ClassName>...)
	SubClasses+	(<ClassName>...)
	ClassProps	(<PropName>...)
	ClassConstraints	(<ClassConstraint>...)
	ClassReference	<ClassEntRepSpec>
	ClassExtension	<ClassName>
	ClassFields	(<PropName>...)
	ClassIndices	(<IndexName>...)
	ClassDistIndices	(<IndexName>...)
	ClassStore	<StoreName>
	RCntEst	<Real>
Properties	Prop?	t
	Updated?	t
	PropType	<ClassName>
	PropConstraint	<PropConstraint>
Queries	Query?	t
	QueryBody	<Query>
Transactions	Trans?	t
	TransBody	<Transaction>
Indices	Index?	<IndexDesc>
	IndexClass	<ClassName>
	IndexType	<IndexType>
	IndexSearchConds	(<SearchCond>...)
	Distributed?	t
	DistPF	<PathFunction>
	StaticIndex?	t
	IndexSize	<Integer>
Stores	Store?	t
	StoreClasses	(<ClassName>...)
	StaticStore?	t
	StoreSize	<Integer>

Table 2. property list forms

<ClassConstraint>
    (Pfd <PathFunction> (<PathFunction>...))
    (Cover (<ClassName>...))

<PropConstraint>
    (Range <Integer> <Integer>)
    (Maxlen <Integer>)

<Query>
(AllQuery <QueryName> (<Var>...) <AccessSpec>)
(OneQuery <QueryName> (<Var>...) <AccessSpec>)

<Transaction>
(ExprTrans <TransName> (<Var>...) <Action> <Expr>)
(StmtTrans <TransName> (<Var>...) <Action>)

<Var>
(QVar <VarName> <PropName>)
(PVar <VarName> <PropName>)
(LVar <VarName> <PropName>)
(EVar <VarName> <PropName>)

<Constant>
(Constant "<STRING>" <BuiltInClass>))
Null

<BuiltInClass>
Integer
String
Real
DoubleReal

<Term>
<Constant>
<Var>
(UnMinusOp <Term>)
(ModOp <Term> <Term>)
(TimesOp <Term> <Term>)
(DivOp <Term> <Term>)
(AddOp <Term> <Term>)
(SubOp <Term> <Term>)
(Apply <Var> <PathFunction>)
(As <Term> <ClassName>)

<PathFunction>
(<PropName>...)

<SimpPF>
(<PropName>)

<Action>
(Block (<Var>...) <Stmt>...)

<Stmt>
(Assign <LeftTerm> <Term>)
(Insert (<Var>...) <Stmt>...)
(Delete (<Var>...))
(Add <IndexName> <Term>)
(Sub <IndexName> <Term>)
(Cre <IndexName> <Term>)
(Des <IndexName> <Term>)
(Alloc <StoreName> <Term>)
(Free <StoreName> <Term>)
(IndirectAlloc <StoreName> <Term>)
(IndirectFree <StoreName> <Term>)
(AllocId <Term>)
(FreeId <Term>)
(AssignId <Term>)
(If <Pred> <Stmt>)
(If <Pred> <Stmt> <Stmt))

<AccessSpec>
(Find <FindInfo> <FindType> <FindEntry>...)

<FindInfo>
(<CostInfo> <CondInfo> <BoundVars> <ProjectionInfo>)

<CostInfo>
<Real>

<CondInfo>
<GlobalCondGraph> <LocalCondGraph>

<ProjectionInfo>
<NotFDEVars> <NotFDSortedEVars>

<BoundVars>
(<Var>...)

<NotFDEVars>
(<Var>...)

<NotFDSortedEVars>
(<Var>...)

<FindType>
<PredFindType>
(All <Var>...)

<PredFindType>
(One)

<FindEntry>
    (AndHeap <Pred>...)
    (ScanHeap <Var>...)
    <Pred>
    (Cut <Var>)
    (Project <Var>...)
    (Scan <Var> <ScanSpec>)
    (AltFind (<AccessSpec>...) <SubGoalSpec>)

<SubGoalSpec>
    <AccessSpec> <Integer> <AccessSpec> <AccessSpec>...

<OrderDir>
    Asc
    Desc

<Pred>
    (EQ <Expr> <Expr>)
    (LT <Expr> <Expr>)
    (GT <Expr> <Expr>)
    (LE <Expr> <Expr>)
    (GE <Expr> <Expr>)
    (Not <Pred>)
    (Find <FindInfo> <PredFindType> <ScanEntry>...)
    (In <Term> <ClassName>)
    (Is <Term> <ClassName>)

<IndexDesc>
    (<SearchCond>...)

<SearchCond>
    (PFCond <PathFunction> <Dir>)
    (SCCond <ClassName>)

<Dir>
    <OrderDir>
    NoOrder

<StoreType>
    (Dynamic)
    (Static <Integer>)

<ScanSpec>
    (Log)
    (Iter <IndexName> [<SelectCond> ...])
    (SCIter <IndexName> [<SelectCond> ...])
    (Lookup <IndexName> [<SelectCond> ...])
    (SCLookup <IndexName> [<SelectCond> ...])
    (Substitute <Expr>)
    (CondSubstitute <Expr>)

<SelectCond>
(QualSC <ClassName>)
(QualPF <PathFunction> <Expr>)

<ClassReference>
System
Pointer
IndPointer
Offset
IndOffset

3. PDM FILE FORMAT

The PDM source file contains the following information.

- specification of record types for object representation (including index fields)
- specification of query access strategies
- compiled transaction code, indicating index and store maintenance requirements
- index declarations
- store declarations