Supporting Finite Element Analysis with a Relational Database Backend

Part I: There is Life beyond Files

Cong Wu

Introduction

- First of three-part,how RDBMS simplify FEA.
- Currently file-based FEA data management practice lacks the scalability and functionality
- The paper is about how to make the transition from a file-based to a database-centric environment in support of FEA

Finite Element Analysis (FEA)

- Model generation
- Meshing
- Attribute assignment
- Solution
- Post Processing
- Visualization Products

Limitations

- lack of a unified data modeling capability
- absence of data integrity enforcement and data coherence
- limited metadata management
- lack of query interfaces
- poor scalability
- no support for automatic parallelism
- poor application integration
- lack of support for Internet-scale distributed applications

FEA Data and Dataflow

- *Model creation*
- *meshing*
- *Attribute definition* and *assignment*
- *Solution*
- *Post-processing*

FEA Data and Dataflow

- *unstructured meshes*
- Structured or meshes allow the data to be stored in a dense array
- unstructured meshes require database representation where each voxel or vertex is an independently addressed item

large scale FEA

- We need an intelligent and scalable way to move data between the different levels of the memory hierarchy
- Post-processing typically starts from results in permanent storage

Basic facts about Relational Databases

- Schema is a logical and abstract description of the data
- A view is a mapping from the schema to a particular data layout as seen by one or more applications.
- This *data independence* was a major reason commercial applications adopted database technology
- Unlike formatted files, database metadata is unified with the data in a database

Microsoft SQL Server 2000

- T-SQL can be used as a programming language for stored procedures and other user-defined functions
- The standard advice is to just use T-SQL to encapsulate data in the database External DLLs can be accessed from SQL Server
- The general advice is to use SQL to retrieve exactly the subset of data that you need
- SQL Server can be accessed from programs running on Windows using the traditional ODBC/JDBC libraries

Mapping an FEA Data Model onto a Relational Model

- Frequently Asked Queries (FAQ).
- The first step:mesh representation
- a table of 3D vertices

```
CREATE TABLE Vertices (
       VertexID int PRIMARY KEY CLUSTERED,
       x float NOT NULL, y float NOT NULL, z float NOT NULL)
```

Tetrahedra Table

- Four vertices

```
CREATE TABLE Tetrahedra (
       ElemID int NOT NULL PRIMARY KEY,
       v0     int NOT NULL REFERENCES Vertices,
       v1     int NOT NULL REFERENCES Vertices,
       v2     int NOT NULL REFERENCES Vertices,
       v3     int NOT NULL REFERENCES Vertices,
       CONSTRAINT Tetrahedra_UNQ UNIQUE(v0, v1, v2, v3),
       CONSTRAINT Tetrahedra_CHK01 CHECK(v0 != v1),
       -- Add the 5 other pair-wise comparisons
       )
```

Awkward queries

- Find all tetrahedra that share a given vertex

```
SELECT ElemID
FROM Tetrahedra
WHERE @VertexID in (v0, v1, v2, v3)
```

Normalized version

```
CREATE TABLE TetrahedronVertices (
       ElemID   int     NOT NULL,
       Rank     tinyint NOT NULL CHECK(Rank < 4),
       VertexID int     NOT NULL REFERENCES Vertices,
       CONSTRAINT PK_TetrahedronVertices PRIMARY KEY(ElemID, Rank),
       CONSTRAINT UNQ_TetrahedronVertices UNIQUE(ElemID, VertexID)
       )
```

Straightforward query

```
SELECT ElemID
FROM TetrahedronVertices
WHERE VertexID = @VertexID
```

Normalized view

```
CREATE VIEW TetrahedronVertices AS
            SELECT ElemID, 0 AS Rank, v0 AS VertexID FROM Tetrahedra
      UNION ALL SELECT ElemID, 1 AS Rank, v1 AS VertexID FROM Tetrahedra
      UNION ALL SELECT ElemID, 2 AS Rank, v2 AS VertexID FROM Tetrahedra
      UNION ALL SELECT ElemID, 3 AS Rank, v3 AS VertexID FROM Tetrahedra
```

FEA Data I/O

- *impedance mismatch (*programming language datatypes and procedural interfaces) and (tables and non-procedural set-oriented access）
- massive data load and dump operations demand the bulk copy client utility，T-SQL BULK INSERT command, or Data Transformation Services (DTS)

I/O

- The most I/O intensive parts of FEA are:
  - The initial loading of the mesh into the database.
  - The attribute and input data generation for the equation solver.
  - The dumping of analysis results to the database.

Indexes

- It is common to implement them as B-trees
- For a *clustered* index, the leaves of the index tree are the rows of the table.
- For a *non-clustered* index, the leaves are pointers to actual rows.

FEA Attributes and Metadata

- dense tabular data 99%, 1% of mostly attributes and metadata
- Showed us it is important, but do not showed us how to do it.

Final

- how our FEA pipeline uses SQL Server as a common data store for both the simulation data and the metadata.
- It uses SQL Server as a data-shell, as the communication medium and glue among the six solution phases

The end

Thank you!

Cong Wu
2010.04.01