When inserting rows into MS Access from c# using TransactionLevel.ReadUncommitted from a prepared statement, the ordering seems off

Asked 15 years, 11 months ago Modified 15 years, 11 months ago Viewed 3k times

0

I have some code in c# that is writing information (row by row) into an access database using prepared statements. I am using OleDb, and a TransactionLevel.ReadUncommitted because sometimes I need to look at the data before committing it.

The problem seems to be that on 1 out of 7 different tables, the ordering that I retrieve the records from the database isn't the same ordering I put them in as. This happens about 1 out of every 4 times I try to use it, so I can't really seem to debug it.

Edit The reason ordering is important here is that we are dumping the table to an excel sheet which matches up with data already existing on there. Order by might make some progress, but there is a column which is ordered by a movement (i.e. N-NW) that couldn't be ordered to match up with the data on the sheet.

I'm leaning toward it being a race condition of the prepared statement's insert command (i.e. there are too many for access to handle at once, so they get jumbled).

Does anybody have any thoughts on this? Below are a couple of snippets of what I'm doing: (Sorry about the length, I tried to cut out as much as I could but still get the main components out)

```
protected override void PopulateTmpTable()
  {
    OleDbCommand objComm = null;
    try
    {
      objComm = new OleDbCommand("", mDbconn);
      ...
      //Begin SQL Transaction
      mTr =
mDbconn.BeginTransaction(System.Data.IsolationLevel.ReadUncommitted);
      objComm.Transaction = mTr;
      //Start Populating Temp Table
      for (int i = 1; i <= mNrows; i++)
      {
        ...
        ProcessNode(objComm, node, approaches);
        ProcessNodeSummary(objComm, node);
      }
      ProcessSummary(objComm);
    }
    catch (Exception e) { }
    finally
    {
      if (mTr != null) mTr.Commit();
      if (objComm != null) objComm.Dispose();
    }
  }  //End Method PopulateTmpTable

  private void ProcessNode(OleDbCommand objComm, string node, List<string>
approaches)
  {
    try
    {
      ...
      OleDbCommand objComm2 = new OleDbCommand("", mDbconn, mTr);
      for (int k = 0; k < MaxLegs; k++)
      {
        ...
            total = ProcessIterations(objComm, node, turning[m], m);
          }
          objComm.ExecuteNonQuery();
        }  //End if
      }  //End for
    }
    catch { }
  }  //End Method ProcessNode

private List<double> ProcessIterations(OleDbCommand objComm, string node,
string turn, int m)
  {
    try
    {
      OleDbCommand objComm2 = new OleDbCommand("", mDbconn, mTr);
      OleDbDataReader objRead;
      objComm.Parameters["parameter"].Value = //Stuff (x2)
        for (int j = 0; j < mIterations; j++)
        {
          ...
          objComm2.CommandText = "SELECT ROUND(AVG(Delay),1), COUNT(VehNo) FROM
[TABLE] WHERE NodeNo=" + node + " AND Movement='" + turn + "' AND Iteration=" +
mIterationValue[j] + mFilter[1];
          objRead = objComm2.ExecuteReader();
          objRead.Read();
          try
          {
              objComm.Parameters["more parameters"].Value = objRead[0];
              ...
          }
          catch { }
          objRead.Close();
        }//End for
        ...
        objComm.ExecuteNonQuery();
        objComm2.CommandText = "UPDATE " + mTmptable + " SET ave=" + avg +
",minimum=" + mMini[m] + ",maximum=" + mMaxi[m] + ",dev=" + stDev + " WHERE
node='" + node + "' AND movement = '" + temp + "';";
        objComm2.ExecuteNonQuery();
      }
    }
    catch{}
    return mTotal;
  }  //End Function ProcessIterations
```

c# ms-access transactions oledb

Share
Improve this question
Follow

edited Jan 19, 2009 at 17:30

asked Jan 17, 2009 at 0:49
Fry
4,266 ● 9 ● 40 ● 51

Can you clarify what you mean by "the ordering that I retrieve from the database isn't the same ordering I put them in as". Is this referring to the order of processing the tables, or the order of the records? If the latter, it's something you shouldn't worry about as that's what ORDER BY is for. – David-W-Fenton Jan 17, 2009 at 19:58

2 Answers

Sorted by: Highest score (default)

0

In Jet, the clustered index (physical ordering) of a table is determined by the PRIMARY KEY or, in absence of a PK, another key chosen (randomly?) by the engine. However, physical ordering only occurs when the database file (.mdb, .mde, .accdb) is compacted. Rows inserted subsequent to compacting (or before the first compact) are inserted in date/time order. Now, the smallest time granule supported by Jet's DATETIME data type is one second, though under the covers the value is stored as a double float, therefore I'm wondering whether the granularity is not fine enough for your purposes. But why rely on such poorly documented features of an elderly engine...?

If you know ther order in which you insert the rows then *model* this in the database e.g. use an INTEGER column to store a sequence number for which your app is in charge of supplying the values. And put a UNIQUE constraint on the column too, just to be sure. Then simply ORDER BY your sequence column in queries.

Share Improve this answer Follow

answered Jan 20, 2009 at 9:12
onedaywhen
56.9k ● 12 ● 102 ● 142

The table is created dynamically, so the number of fields is not known until runtime. So I was using SELECT *, but now I build the SELECT query at the same time as I build the table. In essence, this is exactly what I ended up doing, but this knowledge is helpful - Thanks! – Fry Jan 20, 2009 at 17:36

0

In your PopulateTmpTable() method, you are commiting the transaction even if an error occured. Is that what you intended?

In ProcessNode() you pass in a command object, create a new one and then use the one passed in?

Share Improve this answer Follow

answered Jan 17, 2009 at 0:56

Mitch Wheat

300k ●44 ●477 ●550

Well, most of the code in Process node was removed for brevity, the second command basically queries the source table and retrieves data to populate the temp table. Yes, the transaction is committed bcuz errors are thrown up and the table would be immediately removed with a msg to the user – Fry Jan 17, 2009 at 1:01

I removed the error handling code too as I didn't think it was of much importance – Fry Jan 17, 2009 at 1:02