Item 72
git_comments:

1. * * Reflection-based checker that exceptions thrown by JDBC interfaces' * implementation methods for unsupported-operation cases are SQLExceptions * (not UnsupportedOperationExceptions). * * @param <INTF> JDBC interface type
2. class NoNonSqlExceptionsChecker<INTF>
3. * * Gets minimal value suitable for use as actual parameter value for given * formal parameter type.
4. See if method throws exception:
5. Uncomment to suppress calling DatabaseMetaData.getColumns(...), which sometimes takes about 2 minutes, and other DatabaseMetaData methods that query, collectively taking a while too: else if (DatabaseMetaData.class == jdbcIntf && "getColumns".equals(method.getName())) { logger.debug("Skipping (because really slow): " + methodLabel); } else if (DatabaseMetaData.class == jdbcIntf && ResultSet.class == method.getReturnType()) { logger.debug("Skipping (because a bit slow): " + methodLabel); }
6. class PlainStatementChecker
7. No CallableStatement.
8. * * Assembles (minimal) arguments array for given method.
9. Not executed; for "final".
10. If here, method didn't throw--check if it's an expected non-throwing method (e.g., an isClosed). (If not, report error.)
11. * * Test that non-SQLException exceptions used by Drill's current version of * Avatica to indicate unsupported features are wrapped in or mapped to * SQLException exceptions. * * * As of 2015-08-24, Drill's version of Avatica used non-SQLException exception * class to report that methods/features were not implemented. * * <pre> * 5 UnsupportedOperationException in ArrayImpl * 29 UnsupportedOperationException in AvaticaConnection * 10 Helper.todo() (RuntimeException) in AvaticaDatabaseMetaData * 21 UnsupportedOperationException in AvaticaStatement * 4 UnsupportedOperationException in AvaticaPreparedStatement * 103 UnsupportedOperationException in AvaticaResultSet * </pre>
12. * * Hook/factory method to allow context to provide fresh object for each * method. Needed for Statement and PrepareStatement, whose execute... * methods can close the statement (at least given our minimal dummy * argument values).
13. * * Assembles method signature text for given method.
14. Self-check that member variables are set:
15. ms
16. * * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
17. Expected.
18. (Note: Can't use JdbcTest's connect(...) for this test class.)
19. (No ResultSetMetaData.isClosed() or DatabaseMetaData.isClosed():)
20. Known good-enough case--these methods throw NullPointerException because of the way we call them (with null) and the way Avatica code implements them.
21. Good case--almost any exception should be SQLException or subclass (but make sure not accidentally closed).
22. * * Tests one method. * (Disturbs members set by makeArgsAndLabel, but those shouldn't be used * except by this method.)
23. * "None" value for rowLastColumnOffset.
24. * * Resets last-column-referenced information for {@link #wasNull}. * Must be called whenever row is advanced (when {@link ResultSet#next()} * is called).
25. Update lastColumnIndexedInRow after indexing accessors to not touch lastColumnIndexedInRow in case of out-of-bounds exception.

26. * * @param accessorOffset 0-based index of accessor array (not 1-based SQL * column index/ordinal value)
27. (Not -1, since -1 can result from 0 (bad 1-based index) minus 1 (offset from 1-based to 0-based indexing.)
28. * Zero-based offset of last column referenced in current row. * For {@link #wasNull()}.
29. TODO(DRILL-xxxx): Eliminate this test-specific hack from production code. If we're not going to have tests themselves explicitly handle making names unique, then at least move this logic into a test base class, and have it go through DrillConnection.getClient().

git_commits:

1. **summary:** DRILL-2769: Fix most non-SQLException not-supported-yet exceptions.
   message: DRILL-2769: Fix most non-SQLException not-supported-yet exceptions. Core: Added (auto-scanning) unit test. [Drill2769UnsupportedReportsUseSqlExceptionTest] Added translation of lots of UnsupportedOperationExceptions (and some RuntimeExceptions) from Avatica code to SQLFeatureNotSupportedExceptions (tons of method overrides). Also: Added explicit bounds checks in ResultSetMetaData methods and checking of last-accessed column in DrillAccessorList.wasNull() (to fix other RuntimeExceptions to SQLExceptions). Added resetting of last-accessed column to fix latent bug in DrillAccessorList. Hygiene: - Renamed some zero-based index/ordinal-position parameters to "...Offset". - Renamed some one-based index/ordinal-position parameters to "...Number". - Renamed DrillAccessorList lastColumn to rowLastColumnOffset; declared explicit logical null value for rowLastColumnOffset.

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** many(?) JDBC methods throw non-SQLException exceptions (e.g., UnsupportedOperationException, RuntimeException)
   description: It seems that many JDBC methods throw exceptions of type {{UnsupportedOperationException}} or {{RuntimeException}} to indicate that they are not applicable (e.g., Drill's implementation of {{Connection.commit()}}, since Drill isn't transactional) or not implemented yet (and some throw other {{RuntimeException}}s to indicate other problems). However, these methods should be throwing exceptions of type {{SQLException}} (or subclasses thereof). The JDBC pattern is to throw {{SQLException}}s, not {{RuntimeException}}s, so JDBC client code is not likely to handle {{RuntimeException}}s well. (For example, it is suspected that {{Connection.commit()}}'s throwing of {{UnsupportedOperationException}} is causing a hang in the JDBC client Spotfire.) JDBC does provide a {{SQLFeatureNotSupportedException}}. However, it is specified to be for when "the JDBC driver does does not support an optional JDBC feature." It's not clear how risky it would be to use this exception when Drill does not support a _non_-optional JDBC feature. (Possibly, some methods that can't really do what JDBC specifies might need to just return silently without throwing any exception.)

jira_issues_comments:

1. Some Connection method cases: - isValid - abort - getClientInfo, setClientInfo - commit, rollback; setSavepoint, releaseSavepoint - prepareStatement, prepareCall - getTypeMap, setTypeMap - nativeSql - createArrayOf, createBlob, createClob, createNClob, createSQLXML, createStruct
2. Note: Reportedly, newer versions of the Avatica framework now use a SQLException rather than UnsupportedOperationException.
3. Notes on UnsupportedOperationExceptions (and related RuntimeExceptions) in current Avatica classes: - 5 UnsupportedOperationException in ArrayImpl - 29 UnsupportedOperationException in AvaticaConnection - 10 Helper.todo() (RuntimeException) in AvaticaDatabaseMetaData - 21

UnsupportedOperationException in AvaticaStatement - 4 UnsupportedOperationException in AvaticaPreparedStatement - 103 UnsupportedOperationException in AvaticaResultSet (no UnsupportedOperationException or Helper.todo() in: AvaticaFactory, AvaticaJdbc40Factory, AvaticaJdbc41Factory, AvaticaParameter, AvaticaPrepareResult, AvaticaResultSetMetaData, BuiltInConnectionProperty, ByteString, Casing, ColumnMetaData, ConnectionConfig, ConnectionConfigImpl, ConnectionProperty, ConnectStringParser, Cursor, DriverVersion, Handler, HandlerImpl, Helper, InternalProperty, Meta, Quoting, UnregisteredDriver) A few of the "throw new UnsupportedOperationException()" are not directly in JDBC-defined methods (although most are).
4. Pull request: https://github.com/apache/drill/pull/171 (see DRILL-2489).
5. Fixed by e4f257b1f0200d3f1e977d37b61bdc53fa60533a