C0306 - Isolation

Added by PAVAN KUMAR, last edited by PAVAN KUMAR on Nov 14, 2011

1. Objective

Understanding the purpose of Isolation and how to implement different levels of Isolation.

2. Introduction

The ACID Model is the most important concept in the database theory. The acronym stands for Atomicity, Consistency, Isolation and Durability. Every Database must follow these ACID Properties to maintain reliability. (To know more about ACID). This Article focuses on Isolation Property.

2.1 What is Isolation?

isolation is a property that defines when the changes made by one transaction become visible to other concurrent transactions. this means when there are multiple transactions are executing at the same time on same data, isolation make sure that nothing goes wrong in business.

for example:

when you are booking movie tickets, you choose the seats and proceed to payment gateway. since payment gateway is slow it took one minute to complete the transaction. and in between no one can select the same seats. this is achieved using isolation.

3. Presentation

What are the cases may occur wrongly when multiple transactions executes at same time on same data?

- Dirty read
- Unrepeatable reads
- Phantom reads

3.1 Dirty read

A dirty read occurs when a transaction reads data that has been written by another transaction but has not been yet been committed. this happens when there is a complete lack of synchronization on the data. We return to our banking system for an example of a dirty read:

- A client's bank account has $1000.

- Transaction 1 deposits $500 into the account, but does not yet commit the operation.
- Transaction 2 is posting a check for $1500, reads the account and sees the balance is $1500. It then processes the check against the account and commits.
- Transaction 1, which is still active, rolls back its operation. The balance is restored to the $1000 that it was at before transaction one started.
- The $1500 check has still been cleared and the bank just lost the money!

3.2 Unrepeatable reads

An unrepeatable read occurs when a transaction reads data from a database, but gets a different result if it tries to read the same data again within the same transaction. This typically happens when another transaction writes over some of the data that was read in by the first transaction.

Consider an order entry system where a customer's invoice is being reviewed:

- Clerk 1 is reading an invoice.
- Clerk 2 makes changes to the invoice's line items updating the unit price.
- Clerk 1 fulfills the invoice at the original price.
- The company has charged the client the wrong amount for the order!

The second clerk should not be allowed to modify the order while the first clerk is working with it.

3.3 Phantom reads

A phantom read occurs when a transaction executes multiple reads against a set of data and, in between two of the read operations, another transaction slips in and inserts additional data. This differs from the unrepeatable read problem in that here data is being inserted into our data set, rather than that data merely being updated.

Returning again to our example of an order-entry system, let's say we were fulfilling an order:

- The shipping department reviews the order, packages the items, and sends them on their way.
- Meanwhile, here comes Clerk 2 again who adds an additional item to the order.
- If the shipping department were to review the order again, they would see that another item has magically appeared. The client does not get their correct order and is very upset!

3.4 To fix the above three problems Dirty read, Unrepeatable read, and Phantom read, Isolation uses **Locking** and **Serialization** mechanism:

3.4.1 Locking controls access to a given set of data. The two primary types of locks are **read locks** and **write locks**.

- **Read locks** are non-exclusive locks – they will allow multiple transactions to read data simultaneously.
- **Write locks** are exclusive locks – they will only allow a single transaction to update a set of data.

3.4.2 Serialization guarantees that concurrently executing transactions will behave as if they were executing sequentially, not concurrently. Of course, the transactions will be executing concurrently, but they will appear to be executing in series. The result of serialization is the appearance that multiple transactions are working with data one at a time, in order.

3.5 what are the different isolation levels?

Isolation levels vary from very relaxed to very strict. As might be expected, the stricter the level of concurrency control, the greater the impact on performance. As with so many other issues in designing concurrent systems, special care must be taken when determining which isolation level to use. J2EE provides support for four types of isolation levels, as defined in the java.sql.Connection interface:

- **TRANSACTION_READ_UNCOMMITTED**
- **TRANSACTION_READ_COMMITTED**
- **TRANSACTION_REPEATABLE_READ**
- **TRANSACTION_SERIALIZABLE**

In the following table, an "X" marks each phenomenon that can occur

Isolation Level	Dirty Read	Unrepeatable reads	Phantom read
TRANSACTION_READ_UNCOMMITTED	X	X	X
TRANSACTION_READ_COMMITTED	-	X	X
TRANSACTION_REPEATABLE_READ	-	-	X
TRANSACTION_SERIALIZABLE	-	-	-

the performance of the system decreases because more restrictive isolation levels prevent transactions from accessing the same data. If isolation levels are very restrictive, like *Serializable*, then all transactions, even simple reads, must wait in line to execute. This can result in a system that is very slow.You must also balance the performance needs of your system against consistency.

Simulating the above three problems and fixing by setting proper isolation levels

By Default, Oracle uses isolation level TRANSACTION_READ_COMMITTED, and it allows to change the isolation levels to only TRANSACTION_READ_COMMITTED, TRANSACTION_SERIALIZABLE.

So We use open source database MySql for simulating the Dirty Read, UnRepeatable Reads, Phantom reads.

MySql allows to change the isolation levels to TRANSACTION_READ_UNCOMMITTED, TRANSACTION_READ_COMMITTED, TRANSACTION_REPEATABLE_READ, TRANSACTION_SERIALIZABLE.

We can change the isolation level in java using Connection Interface as:

```
connection.setTransactionIsolation( Connection.TRANSACTION_READ_UNCOMMITTED );
```

Similarly In Hibernate, you can include the following property in configuration file:

```
<property name="hibernate.connection.isolation">2</property>
```

we have to execute parallely more than one transaction at the same time for simulation. so please execute first Transaction 1 and then immediatly Transaction 2.

4. Sample Exercises

Download the attachments of this page, open the java project in eclipse and try the following simulations:

4.1 Dirty Read Problem Simulation

Transaction 1 Decreases the cost of a product by 10% and not committed.

Transaction 2 Reads the updated cost of a product and performs billing.

here Transaction 2 reads the updated cost, which is not committed. and Transaction 1 is rollbacked.

Find the Attached Java Project to simulate this case:

Problem: DirtyReadProblemTransaction1.java , DirtyReadProblemTransaction2.java.

Execute first DirtyReadProblemTransaction1 and then parallely DirtyReadProblemTransaction2.

This problem can be avoided by setting the isolation level to: TRANSACTION_READ_COMMITTED.

Solution: DirtyReadSolutionTransaction1.java, DirtyReadSolutionTransaction2.java

4.2 Unrepeatable Reads Simulation

Transaction 1: Read the cost of a product

Transaction 2: Decreases the cost of same product by 10% and performs commit.

Transaction 1: Again, Reads the cost of a product.

here Transaction 1, gets different prices for each read in the same transaction.

Problem: RepeatableReadProblemTransaction1.java, RepeatableReadProblemTransaction2.java

This problem can be avoided by setting the isolation level to: TRANSACTION_REPEATABLE_READ.

Solution: RepeatableReadSolutionTransaction1.java, RepeatableReadSolutionTransaction2.java.

4.3 Phantom Read Simulation

Transaction 1: Reads the list of products which are having the cost more than Rs: 10000/-

Transaction 2: Inserts a new product with cost Rs: 15000/- and perform commits.

Transaction 1: Again reads the list of products which are having the cost more than Rs: 10000/-

here Transaction 1, second time read gets more records than in the first read. and these additional added records are called as Phantom rows.

Problem: PhantomReadsProblemTransaction1.java, PhantomReadsProblemTransaction2.java

This problem can be avoided by setting the isolation level to: TRANSACTION_SERIALIZABLE.

Solution: PhantomReadsSolutionTransaction1.java, PhantomReadsSolutionTransaction2.java

5. Assignments / Assessment Criteria

- Download the attached Isolation Java Project, and try with MySql DB (since oracle does not allow to simulate dirty reads,..) (Installed MySql DB in my machine and connection url is pointing to this database, so you can simply download and try these cases).
- Identify the difference between Unrepeatable reads and Phantom reads.

6. Closure Criteria

Now we know the problems when concurrent transaction uses same data at same time, and by setting isolation levels we can avoid these problems. we need to understand the actual situation of the business and decide which level restriction can be used since the stricter the level of concurrency control, the greater the impact on performance.

7. References

Isolation and Database Locking

Basics of Transactions

Transaction Isolation

Isolation Configuration in Hibernate

Name	Size	Creator	Creation Date	Comment
DB Creation.txt	0.7 kB	PAVAN KUMAR	Sep 27, 2011 08:14	Product table creation script
mysql-connector-java-commercial-5.1...	758 kB	PAVAN KUMAR	Sep 26, 2011 13:45	My SQL driver jar
Isolation.zip	750 kB	PAVAN KUMAR	Sep 29, 2011 16:38	Isolation Java Project

Labels

acid isolation knopz java

Child Pages (1)

ACID Properties

Comments (2) Show Comments | Add Comment