Concurrency in databases

Alexander Shelemin

Senior developer at Veeam

Veeam In Numbers

R&D Offices :

Prague
Saint - Petersburg

82%
of Fortune 500

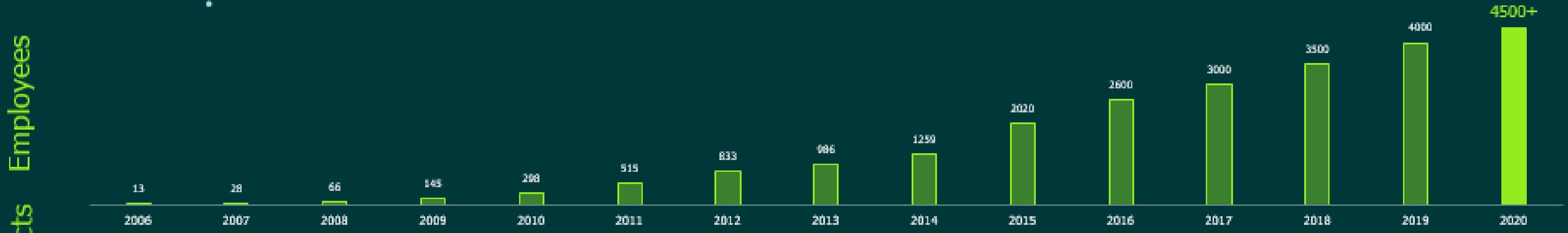
35+
Countries where Veeam has offices

400,000+
Customers worldwide

4500+
Employees worldwide

В 3.5 раза
Industry Average for Customer Satisfaction

$1+ млрд.
in revenue in 2019



Products

2006	Veeam FastSCP; Veeam Monitor for VMware
2007	Veeam Reporter
2008	Veeam Management Pack; Veeam Backup & Replication
2010	Veeam ONE
2016	Veeam Agent for Windows
2017	Veeam Agent for Linux; VB for Microsoft Office 365; VA Console
2019	Orchestrator; VA for Nutanix; Veeam Agent for Unix

Concurrency in databases

Concurrency: 2 or more clients executing code at the same time.

Databases: the talk is relevant for popular RDBMS (SQL Server, Postgres, MySQL, Oracle)

Agenda

- Transactions and atomicity
- Locking and deadlocks
- Isolation levels
- Let’s race!

Transactions

$$\begin{tikzcd}
 & \text{consistency} \arrow[d] & \\
\text{atomicity} \arrow[r] & \text{ACID} & \text{durability} \arrow[l] \\
 & \text{isolation} \arrow[u] &
\end{tikzcd}$$

Transactions: atomicity

atomicity in java, c#, c++: ‘no race conditions’

atomicity in databases: ‘all or nothing’

Don’t mix these!

Transactions: atomicity

Advice that doesn’t really work:

“Just wrap it all in a transaction!”

“Just rewrite everything as a single SQL query”

Locks, blocking, deadlocks

Locks:

- type
  - shared
  - exclusive
- scope
  - table
  - page
  - row
- duration

Locks, blocking, deadlocks

Blocking: progress can be made.

Deadlocks: no progress can be made. DB will abort one of the transactions.

- usually OK to retry

Transactions: isolation

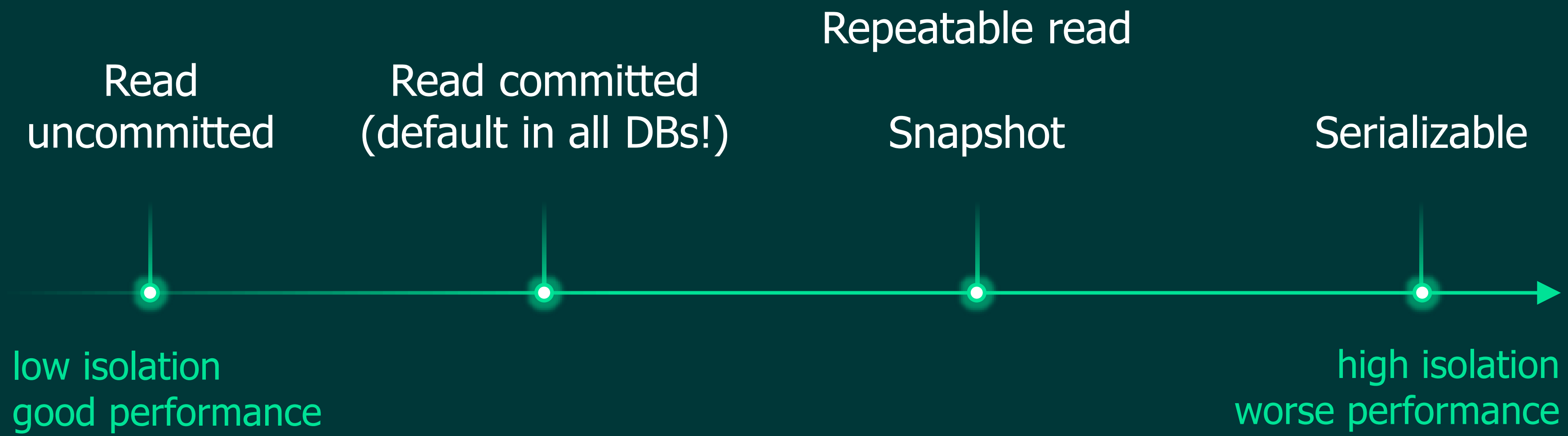
Expectation:

each transaction is running as if alone.

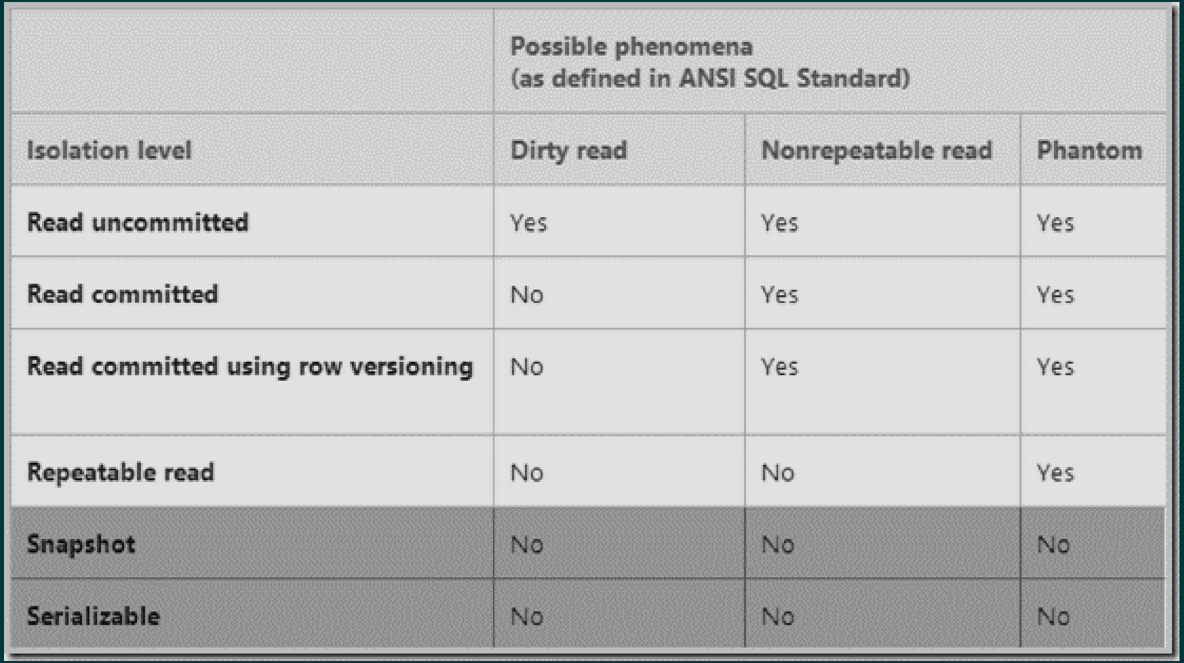
Reality:

performance sucks, need some tradeoffs.

Transactions: isolation levels



Isolation levels - anomalies



	Possible phenomena (as defined in ANSI SQL Standard)		
Isolation level	Dirty read	Nonrepeatable read	Phantom
Read uncommitted	Yes	Yes	Yes
Read committed	No	Yes	Yes
Read committed using row versioning	No	Yes	Yes
Repeatable read	No	No	Yes
Snapshot	No	No	No
Serializable	No	No	No

Картинка взята из статьи https://sqlperformance.com/2014/06/sql-performance/the-snapshot-isolation-level

Transactions: isolation levels

- Part of SQL language standard (SQL-92)
- Can be set on a transaction level.
- Some systems allow changing the default on DB level, some don’t

Isolation levels: implementation

SQL standard doesn’t specify implementation details, so different vendors do things differently.

Two popular approaches:

Locking (read/write locks, 2PL).

- Amount and duration of locks depend on isolation level
- Usually means writers block readers.

MVCC.

- Readers select data from a snapshot.
- Data may be stale.

Isolation levels: read uncommitted

- Allows dirty reads (i.e. reads of data from transactions that are not yet committed)
- ‘NOLOCK’ hint in SQL Server – not needed with Read Committed Snapshot.
- In Postgres and Oracle it maps to read committed – no way to read dirty data.
- Just don’t use it!

Isolation levels: read committed

Default in most popular DBMS

Allows nonrepeatable (fuzzy) reads

- You do the same SELECT twice in the same transaction.
- Rows can be UPDATED or DELETED by other transactions between your SELECTs.

```
-- Session 1                            -- Session 2
BEGIN TRAN
SELECT num from T1; -- 1
                                        UPDATE T1 set num = 123;
SELECT num from T1; -- 123
COMMIT;
```

Isolation levels: read committed

Default in most popular DBMS (Postgres, SQL Server, Oracle, MySQL)

2 different implementations in SQL Server:

- Read Committed – uses locking
- Read Committed Snapshot – uses MVCC.
    - Use Read Committed Snapshot as default!

Isolation levels: repeatable read

Data that you read cannot change until commit.

But, allows ‘phantom reads’

- You do the same SELECT with twice in the same transaction.
- Rows you saw once will stay the same.
- BUT new rows can be INSERTED by other transactions between your SELECTS.

Isolation levels: repeatable read

```
-- Session 1
BEGIN TRAN
SELECT count(*) from T1 where id > 100; -- 666

SELECT count(*) from T1 where id > 100; -- 667
COMMIT;
```

```
-- Session 2


 INSERT INTO T1(id) values (100500);
```

Isolation levels: snapshot

Provides point-in-time snapshot for the whole database for the duration of transaction.

- Very convenient for reports or batch selects.
- Data can be out of date.
- Allows “write skew” anomaly.
- Read committed snapshot works per statement!

Isolation levels: serializable

Doesn’t allow any anomalies.

Causes a lot of deadlocks/conflicts to guarantee that.

Use sparingly in highly concurrent systems.

Beyond isolation levels

"SELECT FOR UPDATE" (aka UPDLOCK in SQL Server)

- Useful for concurrent select/update queries.

You can lock the whole table – doesn't scale, but in rare cases could be a good solution – think batch inserts.

ON CONFLICT (Postgres, MySQL)

Application locks (called advisory locks in Postgres)

Let’s UPSERT some data!

Demo

Conclusions

- Don’t panic, but understand that race condition can happen.
- Look for cases with highly concurrent data modifications and pick isolation level and/or locks accordingly.
- Constraints are important! UNIQUE will help you find a problem in testing, not in production when you have duplicate rows.
- Stress testing combined with constraints is powerful.

References

- Martin Kleppmann’s talk on transactions https://www.youtube.com/watch?v=5ZjhNTM8XU8, als o his book is superb ("Designing data intensive applications").
- Critique of isolation levels paper overview: https://blog.acolyer.org/2016/02/24/a-critique-of-ansi-sql-isolation-levels/
- https://github.com/ept/hermitage a tool to research transactional anomalies and guarantees.
- https://sqlperformance.com/2014/07/t-sql-queries/isolation-levels in-depth explanation of SQL Server isolation levels behavior.
- http://www.bailis.org/blog/understanding-weak-isolation-is-a-serious-problem/ - interesting thoughts on prevalence of weak isolation in production systems.

Senior C# Developer (с возможностью переезда в Прагу) 2

Full Stack C# Developer

C# Developer (с возможностью релокации в Прагу)

Thank you

VEEAM