# چالش‌ها و راهکارهای کنترل همروندی در پایگاه داده‌ی گرافی Neo4j

محمدحسین پوینده پور شهرضایی[1]، سروش قدیسه[2]

## 1- مقدمه

**پایگاه داده‌های گرافی مانند** Neo4j **به دلیل ساختار منعطف و قابلیت نمایش روابط پیچیده، در بسیاری از کاربردها مانند تحلیل شبکه‌های اجتماعی و سیستم‌های پیشنهاددهنده استفاده می‌شوند. اما مدیریت همزمانی** (Concurrency Control) **در این سیستم‌ها چالش‌های خاص خود را دارد، زیرا گره‌ها و یال‌ها دارای ارتباطات پیچیده‌ای هستند. کنترل همروندی در پایگاه داده‌های گرافی به ویژه در محیط‌های توزیع‌شده، یکی از مهم‌ترین دغدغه‌های طراحی سیستم‌های پایگاه داده است.**

**در این مقاله، مکانیزم‌ها و پروتکل‌های کنترل همروندی در** Neo4j **بررسی می‌شوند و با توجه به پایگاه داده‌ی پیاده‌سازی‌شده (شامل فیلم‌ها، بازیگران و روابط بین آن‌ها) تحلیل می‌شود که چگونه** Neo4j **چالش‌های مربوط به همروندی را مدیریت می‌کند.**

## 2- معماری ومدل داده در پایگاه داده‌ی پیاده‌سازی‌شده

**پایگاه داده‌ی گرافی مورد استفاده، شامل نودهای زیر است:**

- **Movie** (فیلم‌ها)
- **Person** (کارگردان‌ها و بازیگران)
- **Category** (ژانرها)
- **Year** (سال انتشار)
- **Country** (کشور تولید)
- **Type** (سریال یا فیلم)

**روابط مهم در این گراف شامل موارد زیر هستند:**

- directed (کارگردانی‌شده توسط)
- acted_in (بازی شده در)
- where (محل تولید)
- created_on (انتشار در سال مشخص)
- in_category (ژانر مرتبط با فیلم)

**با این ساختار، همزمانی به هنگام اجرای تراکنش‌هایی مانند افزودن، حذف یا به‌روزرسانی اطلاعات فیلم‌ها و روابط بین آن‌ها بسیار مهم است.**

## 3- مکانیزم‌های کنترل همروندی در Neo4j

### 3-1- کنترل نسخه‌های چندگانه (MVCC)

Neo4j برای خواندن داده‌ها از MVCC[1] استفاده می‌کند.
این روش باعث می‌شود که **تراکنش‌های خواندن بدون نیاز به قفل شدن داده‌ها اجرا شوند**، زیرا نسخه‌ای از داده در زمان شروع تراکنش ذخیره می‌شود و تا پایان خوانده می‌شود.
MVCC باعث کاهش **بن‌بست (Deadlock)** و افزایش کارایی می‌شود[1] .

### 3-2- املا قفل‌های نوشتاری[2]

**برای تراکنش‌های** نوشتنی **(مانند به‌روزرسانی فیلم‌ها یا افزودن بازیگران جدید)، از** Neo4j قفل‌های نوشتاری سطح گره[3]و یال [4] **استفاده می‌کند.**
**این قفل‌ها تضمین می‌کنند که دو تراکنش به‌طور همزمان یک گره یا رابطه را تغییر ندهند.**
**اگر یک تراکنش بخواهد روی یک نود یا یال قفل بگیرد، تا زمان آزاد شدن قفل، تراکنش دیگر باید منتظر بماند یا به Rollback منجر شود**[2] .

### 3-3- ایزوله‌سازی تراکنش‌ها

**Neo4j** تراکنش‌ها را با سطح ایزوله‌سازی **Read Committed** اجرا می‌کند، یعنی یک تراکنش فقط می‌تواند تغییرات تاییدشده[5] را ببیند. این ویژگی باعث کاهش تناقض داده‌ها بین تراکنش‌های در حال اجرا می‌شود**[3] .**

---

[1] **Multiversion Concurrency Control**
[2] **Write Locks**
[3] **Node Lock**
[4] **Relationship Lock**
[5] **Committed**

### 3-4- کنترل خوش‌بینانه (Optimistic Concurrency Control)

در بسیاری از سناریوها، به‌جای قفل‌گذاری سخت‌گیرانه، Neo4j از **کنترل خوش‌بینانه** استفاده می‌کند.
این روش به‌خصوص زمانی مفید است که احتمال رخداد تضاد تراکنشی کم باشد. در این روش، تراکنش‌ها بدون قفل اجرا می‌شوند و در پایان بررسی می‌شود که آیا داده‌ها از زمان آغاز تراکنش تغییر کرده‌اند یا نه. اگر تغییر رخ داده باشد، تراکنش مجدداً اجرا می‌شود . [4]

## 4- چالش‌های کنترل همروندی در پایگاه داده‌ی پیاده‌سازی‌شده

### 1-4- تداخل در به‌روزرسانی‌های همزمان

مثال: دو تراکنش همزمان سعی در تغییر ژانر یک فیلم دارند. در این صورت یکی از تراکنش‌ها باید منتظر بماند یا لغو شود[5] .

### 2-4- مشکلات خواندن کثیف(Dirty Reads)

با توجه به سطح ایزوله‌ی Read Committed، در Neo4j امکان خواندن داده‌هایی که هنوز تایید نشده‌اند وجود ندارد. این موضوع جلوی خواندن داده‌های نامعتبر را می‌گیرد[6] .

### 3-4- بن‌بست(Deadlocks)

اگر دو تراکنش همزمان قفل‌های متقابل روی نودهای مختلف بگیرند، ممکن است بن‌بست رخ دهد Neo4j .به‌طور خودکار یکی از تراکنش‌ها را لغو کرده و دیگری را ادامه می‌دهد[7] .

## 5- راهکارهای بهینه‌سازی کنترل همروندی در Neo4j

- **افزایش سطح ایزوله‌سازی تراکنش‌ها** برای کاهش احتمال تداخل داده‌ها[7].
- **استفاده از الگوریتم‌های زمان‌بندی تراکنش** برای اولویت‌بندی پردازش‌ها و جلوگیری از بن‌بست‌ها.[8]
- **بهینه‌سازی گره‌ها و روابط پرتکرار** برای کاهش تأخیرهای تراکنشی در به‌روزرسانی‌های مکرر.[5]
- **استفاده از استراتژی‌های خوش‌بینانه و ترکیب آن با قفل‌گذاری محدود** برای دستیابی به تعادل بین کارایی و سازگاری داده‌ها[6].

## 6- نتیجه‌گیری

مدیریت همزمانی در **Neo4j**با ترکیب **MVCC، قفل‌گذاری، ایزوله‌سازی Read Committed و کنترل خوش‌بینانه** انجام می‌شود. این روش‌ها باعث **افزایش کارایی و کاهش تضادهای تراکنشی** در پایگاه داده‌های گرافی می‌شوند. در پایگاه داده‌ی پیاده‌سازی‌شده‌ی مربوط به فیلم‌ها و سریال‌ها، اجرای بهینه‌ی این روش‌ها می‌تواند باعث جلوگیری از **بن‌بست‌ها، تضادهای نوشتاری و کاهش تأخیرهای تراکنشی** شود.

## مراجع


[1] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, vol. 13, no. 6, pp. 377-387, 1970.
[2] P. A. Bernstein, V. Hadzilacos, and N. Goodman, "Concurrency Control and Recovery in Database Systems," Addison-Wesley, 1987.
[3] M. Stonebraker, "The Case for Shared Nothing," IEEE Database Engineering Bulletin, vol. 9, no. 1, pp. 4-9, 1986.
[4] C. Curino, E. Jones, Y. Zhang, and S. Madden, "Schism: a Workload-Driven Approach to Database Replication and Partitioning," Proceedings of the VLDB Endowment, vol. 3, no. 1-2, pp. 48-57, 2010.
[5] T. Özsu and P. Valduriez, "Principles of Distributed Database Systems," Springer, 2011.
[6] J. Gray and A. Reuter, "Transaction Processing: Concepts and Techniques," Morgan Kaufmann, 1993.
[7] H. Garcia-Molina, J. D. Ullman, and J. Widom, "Database Systems: The Complete Book," Pearson, 2008.
[8] D. Kossmann, "The State of the Art in Distributed Query Processing," ACM Computing Surveys, vol. 32, no. 4, pp. 422-469, 2000.