6
Referências

ABADI, D. J.; MARCUS, A.; MADDEN, S. R.; HOLLENBACH, K. **Scalabe semantic Web data management using vertical partitioning**. Proceedings of The 33rd International Conference on Very Large Databases (VLDB, 2007), Vienna, Austria, September 2007, pages 411-422.

BECKETT, D.; BROEKSTRA, J. **SPARQL Query Results XML Format**. W3C Recommendation 15 January 2008. Disponível em <http://www.w3.org/TR/rdf-sparql-XMLres/>. Acessado em 08 de dezembro de 2011.

BERNES-LEE, T. **Linked Data**. Design Issues about Web architecture, June 2006. Disponível em <http://www.w3.org/DesignIssues/LinkedData.html>. Acessado em 09 de dezembro de 2011.

BERNES-LEE, T.; HENDLER, J.; LASSILA, O. **The Semantic Web**. Scientific American, May 2001, p. 29-37.

BERNSTEIN, P. A.; HADZILACOS, V.; GOODMAN, N. **Concurrency Control and Recovery in Database System**. Addison-Wesley, 1987, ISBN: 0-201-10715-5.

BERNSTEIN, P. A.; NEWCOMER, E. **Principles of Transaction Processing.** Morgan Kaufmann, 2 edition, 2009, ISBN: 1-558-60623-8.

BOMFIM, M. H. S. **Um método e um ambiente para o desenvolvimento de aplicações na Web Semântica.** Dissertação de Mestrado, Departamento de Informática, PUC-Rio, 2011.

BRICKLEY, D.; GUHA, R. V. **RDF Vocabulary Description Language 1.0: RDF Schema**. W3C Recommendation 10 February 2004. Disponível em <http://www.w3.org/TR/rdf-schema/>. Acessado em 07 de dezembro de 2011.

BROCHELER, M.; PUGLIESE, A.; SUBRAHMANIAN, V. S. **Dogma: A disk-oriented graph matching algorithm for rdf databases**. Proceedings of International Semantic Web Conference, 2009, pages 97-113.

CACHERO, C.; GÓMEZ, J. **Advanced Conceptual Modeling of Web Applications: Embedding Operation Interfaces in Navigation Design**. Proceedings of 21th International Conference on Conceptual Modeling. El Escorial, Madrid, November 2002.

CLARK, K. G.; FEIGENBAUM, L.; TORRES, E. **SPARQL Protocol for RDF**. W3C Recommendation 15 January 2008. Disponível em <http://www.w3.org/TR/rdf-sparql-protocol/>. Acessado em 08 de dezembro de 2011.

DEAN, M.; SCHREIBER, G. **OWL Web Ontology Language Reference**. W3C Recommendation 10 February 2004. Disponível em <http://www.w3.org/TR/owl-ref/>. Acessado em 07 de dezembro de 2011.

DISTANTE, D.; ROSSI, G.; CANFORA, G.; TILLEY, S. **A comprehensive design model for integrating business process in Web applications**. Int. J. Web Engineering and Technology, Vol. 3, No. 1, 2007.

DISTANTE, D.; TILLEY, S. **Conceptual Modeling of Web Application Transactions: Towards a Revised and Extended Version of th UWA Transaction Design Model**. Proceedings of the 11th International Multi-Media Modeling Conference (MMM 2005), Melbourne, Australia, Los Alamitos, CA: IEEE Computer Society Press, January 2005.

FLANAGAN, D.; MASTUMOTO, Y. **The Ruby Programming Language**. O'Reilly, 2008, ISBN: 0-596-51617-7.

FOWLER, M.; RICE, D.; FOEMMEL, M.; HIEATT, E.; Mee, R.; STAFFORD, R. **Patterns of Enterprise Application Architecture**. Addison-Wesley, 2003, ISBN: 0-321-12742-0.

FOWLER, M; PARSONS, R. **Domain Specific Languages**. Addison-Wesley, 2011, ISBN: 0-321-71294-3.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley, 1994, ISBN: 0-201-63361-2.

GIOLDASIS, H.; CHRISTODOULAKIS, S. **UTML: Unified Transaction Modeling Language**. The 3rd International Conference on Web Information Systems Engineering (WISE 2002), Singapore, December 2002.

GRAY, J. N.; LORIE, R. A.; PUTZOLU, G. R.; TRAIGER, I. L. **Granularity of *Locks* and Degrees of Consistency in a Shared Data Base.** Proceedings of IFIP Working Conference on Modeling of Data Base Management Systems, Freudenstadt, Germany, January, 1976, pp. 695-723.

GRAY, J. **The Transaction Concept: Virtues and Limitations.** Proceedings of Seventh International Conference on Very Large Databases (VLDB), June 1981.

HARMELEN, v. F.; PATEL-SCHNEIDER, P. F.; Horrocks, I. **Reference Description of the DAML+OIL Ontology Markup Language**. March 2001. Disponível em <http://www.daml.org/2001/03/reference>. Acessado em 08 de dezembro de 2011.

HEE, K. V.; AALST, W. V. D. **Workflow Management: Models, Methods, and Systems**. MIT Press, 2004, ISBN: 0-262-72046-9.

HEFLIN, J. **OWL Web Ontology Language Uses Cases and Requirements.** W3C Recommendation 10 February 2004. Disponível em <http://www.w3.org/TR/webont-req/#onto-def>. Acessado em 07 de dezembro de 2011.

JACYNTHO, M. D. **Estado da Arte em Design Conceitual de Transação em Aplicações Web.** Monografia apresentada em estudo orientado durante o curso de doutorado, Departamento de Informática, PUC-Rio, Dezembro, 2007.

JACYNTHO, M. D.; SCHWABE, D. **Modelos e Meta-Modelos para Transações em Aplicações Web.** Anais do XVI Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia), Belo Horizonte - MG, Brasil, Outubro, 2010a.

JACYNTHO, M. D.; SCHWABE, D. **Models e Meta-Models for Transactions in Web Applications.** In 6th Model-Driven Web Engineering Workshop (MDWE 2010). Proceedings of the 10th international conference on Current trends in Web engineering (ICWE'10) Viena, Áustria - LNCS, vol. 6385. pp. 37-48, 2010b.

KOCH, N.; KRAUS, A. **The Expressive Power of UML-based Web Engineering**. Proceedings IWWOST 2002, CYTED, 105-119.

KOCH, N.; KRAUS, A.; CACHERO, C.; MELIÁ, S. **Modeling Web Business Processes with OO-H and UWE**. Proceedings of 3rd International Workshop on Web Oriented Software Technology (IWWOST 2003), Oviedo, Spain, July 2003.

KRASNER, G. E.; POPE, S. T. **A cookbook for using the model-view controller user interface paradigm in Smalltalk-80**. Journal of Object-Oriented Programming, Vol. 1, No. 3, pp. 26-49, August/September 1988.

LIMA, F. **Modelagem semântica de aplicações na WWW.** Tese de Doutorado, Departamento de Informática, PUC-Rio, 2003.

LOD Project. **Linking Open Data initiative**. Disponível em <http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>. Acessado em 04 de dezembro de 2011.

MANOLA, F.; MILLER, E. **RDF Primer**. W3C Recommendation 10 February 2004. Disponível em <http://www.w3.org/TR/rdf-primer/>. Acessado em 04 de dezembro de 2011.

NEUMANN, T.; WEIKUM, G. **x-RDF-3X: Fast Querying, High Update Rates, and Consistency for RDF Databases**. Proceedings of The 36th International Conference on Very Large Databases (VLDB, 2010), Singapore, September 2010, pages 256 – 263.

PAPAZOGLOU, M. P. **Web Services and Business Transactions.** Proceedings of World Wide Web: Internet and Web Information Systems, March 2003, 49-91.

PERROTA, P. **Metaprogramming Ruby**. Pragmatic Bookshelf, 2010, ISBN: 1-934356-47-6.

PRUD'HOMMEAUX, E.; SEABORNE, A. **SPARQL Query Language for RDF**. W3C Recommendation 15 January 2008. Disponível em < http://www.w3.org/TR/rdf-sparql-query/>. Acessado em 04 de dezembro de 2011.

RIES, D. R.; STONEBRAKER, M. R. ***Locking* Granularity Revisited**. Journal of ACM Transactions on Database Systems, vol. 4, no. 2, June, 1979, pp. 210-227.

ROSSI, G. **Um método orientado a objetos para projeto de aplicações hipermídia.** Tese de Doutorado, Departamento de Informática, PUC-Rio, 1996.

ROSSI, G.; SCHMID, H.; LYARDET, F. **Engineering Business Process in Web Applications: Modeling and Navigation Issues**. Proceedings of 3rd International Workshop on Web Oriented Software Technology (IWWOST 2003), Oviedo, Spain, July 2003.

SANTOS, D. L. **Um Modelo de Operações para aplicações na Web semântica.** Dissertação de Mestrado, Departamento de Informática, PUC-Rio, 2010.

SCHMIT, B.; DUSTDAR, S. **Model-driven Development of Web service Transactions.** International Journal of Enterprise Modeling and Information Systems Architecture, 1(1), October 2005.

STICKER, P. **CBD - Concise Bounded Description**. W3C Member Submission 3 June 2005. Disponível em < http://www.w3.org/Submission/CBD/ >. Acessado em 04 de dezembro de 2011.

STROKA, S. **Transaction Management and Historiography in Social Semantic Web Applications**. Diploma Thesis, Salzburg University of Applied Sciences, 2009.

TROYER, O. D.; CASTELEYN, S. **Modeling Complex Processes for Web Application using WSDM**. Proceedings of 3rd International Workshop on Web Oriented Software Technology (IWWOST 2003), Oviedo, Spain, July 2003.

UWA CONSORTIUM. **Ubiquitous Web applications**. Proceedings of The eBusiness and eWork Conference (e2002), Prague, Czech Republic, October 2002.

WEIKUM, G.; VOSSEN, G. **Transactional Information Systems. Theory, Algorithms, and the Practice of Concurrency Control and Recovery.** Morgan Kaufmann, 2002, ISBN: 1-55860-508-8.

WEISS, C.; KARRAS, P.; BERNSTEIN, A. **Hexastore: sextuple indexing for semantic Web data management**. Proceedings of The 34rd International Conference on Very Large Databases (VLDB, 2008), Auckland, New Zealand, August 2007, pages 1008- 1019.

YU. L. **A Developer´s Guide to the Semantic Web.** Springer, 2011, ISBN: 978-3-642-15969-5.

Apêndice

Este apêndice apresenta todos os detalhes da simulação realizada para avaliar o desempenho do modelo de bloqueio proposto, desde a estratégia e design da simulação até a implementação. Todos os resultados da simulação são mostrados graficamente. Por fim, também são listados todos os códigos fonte relativos à implementação deste trabalho. Todos eles, com exceção do vocabulário de aquisição de bloqueios, que foi escrito em OWL, são scripts Ruby.

A.1. Simulação - Estratégia e Resultados

O modelo de bloqueio proposto foi avaliado por meio de simulação, a fim de se obter evidências que:

- Os quatros grânulos propostos ("*Graph*", "*Property*", "*Resource*" e "*Property of Resource*") permitem obter um melhor desempenho, de acordo com os tamanhos das transações, se comparado apenas com o grânulo *Resource* (que é o que mais se aproxima de uma tupla no esquema relacional);
- O protocolo multigranular proposto, de fato, melhora o desempenho, com relação ao protocolo monogranular, quando temos transações de vários tamanhos;
- Os novos tipos de bloqueio de leitura e escrita, para inserção e remoção, compatíveis entre si, melhoram o nível de multiprogramação quando comparados com os tipos de bloqueio de leitura e escrita convencionais, incompatíveis.

A.1.1. Design da Simulação

A simulação[50] também foi desenvolvida na linguagem Ruby e o design de classes, com as principais propriedades e operações, é exposto na Figura 46.

[50] Veja Apêndice, seção A.5, para código fonte completo da simulação.

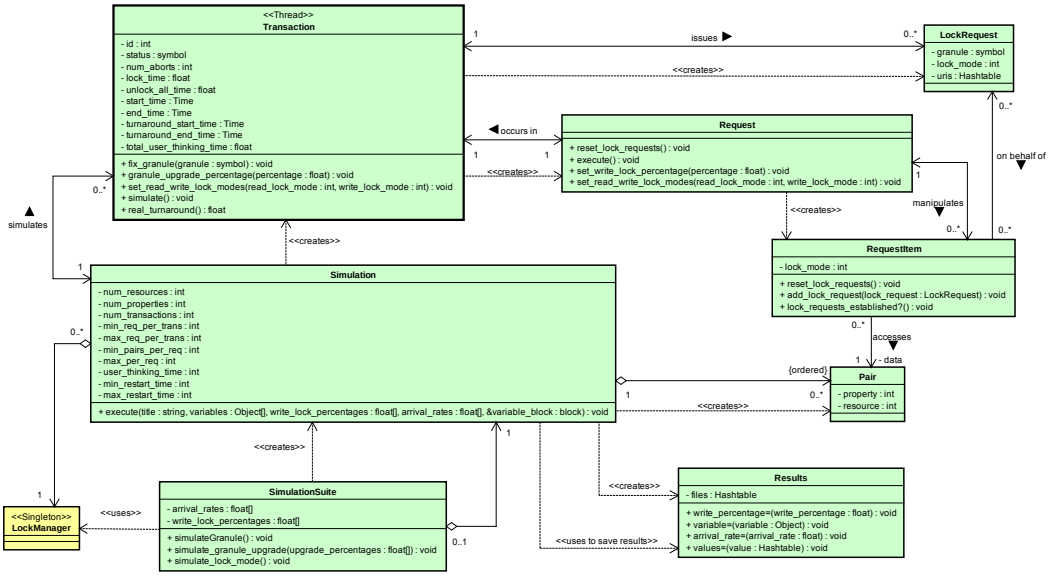

Figura 46 - Modelo de classes da simulação

Basicamente, temos um conjunto de simulações (`SimulationSuite`), onde cada simulação (`Simulation`) cria várias transações (`Transaction`), e cada transação é simulada no seu próprio *thread* de execução. Cada transação emite uma requisição (`Request`) composta de vários itens (`RequestItem`), e cada item é definido por um tipo de bloqueio (*`lock`*`_mode`) e um item de dado (`Pair`). Item de dado nada mais é do que um par (propriedade, recurso) que, a rigor, representa o par (URI da propriedade, URI do recurso). Para cada transação, de acordo com o tipo de simulação, são criadas requisições de bloqueio (`LockRequest`) que cubram todos os itens da transação. Para simular as URIs dos recursos e propriedades foram utilizados números inteiros sequenciais.

Ao final da simulação, o objeto `Simulation` usa o objeto `Results` para gerar um arquivo para cada valor aferido.

Antes de iniciar a simulação propriamente dita, é preciso preparar a composição das transações, ou seja, a carga (ou perfil de acesso). Para cada transação, é instanciado o objeto que representa sua requisição. Ao instanciar um objeto requisição, são instanciados os respectivos itens. Para cada item, é gerado, aleatoriamente e sem repetição, um dentre os possíveis pares (propriedade, recurso). Uma vez estabelecida a composição de cada transação, esta mesma estrutura, com os mesmos pares (propriedade, recurso) é utilizada em todas as simulações de uma *suite* de simulações (`SimulationSuite`). Ou

seja, dentro de uma mesma *suite*, todas as simulações são feitas com as mesmas transações.

Os parâmetros de entrada utilizados para cada simulação (atributos das classes `Simulation e SimulationSuite`) são:

- **num_resources**
  Número de recursos.

- **num_properties**
  Número de propriedades.

- **num_transactions**
  Número de transações. O número de transações é fixo. O que varia é a taxa de chegada das transações.

- **min_pairs_per_req**
  Número mínimo de pares (propriedade, recurso) por requisição.

- **max_pairs_per_req**
  Número máximo de pares (propriedade, recurso) por requisição.

- ***min_restart_time***
  Tempo mínimo a esperar para retentar, em caso de aborto, por não ter conseguido obter um bloqueio. Simulado em segundos.

- ***max_restart_time***
  Tempo máximo a esperar para retentar, em caso de aborto, por não ter conseguido obter um bloqueio. Simulado em segundos.

- ***arrival_rates***
  Vetor com as taxas λ de chegada das transações (transações por segundo). As transações chegam de acordo com um processo de *Poisson*[51] com taxa λ. A propósito, taxa infinita (λ = ∞) significa que todas as transações chegam ao mesmo tempo.

[51] *Poisson process* - http://en.wikipedia.org/wiki/Poisson_process

- ***write_lock_percentages***
  Vetor com as porcentagens, dentre todos os bloqueios, de bloqueios de escrita. Todas as transações, ao definir os tipos de bloqueio (*lock_mode*), respeitam esta porcentagem.

Os parâmetros de saída aferidos durante cada simulaçao são:

- **Número de bloqueios**
  Número médio de bloqueios, em cada um dos quatro grânulos, requisitados por cada transação.

- **Número de abortos**
  Número médio de abortos que cada transação sofreu. Ou seja, número de conflitos e, por conseguinte, de reinícios.

- ***Turnaround* (Tempo de Resposta Médio)**
  Este é um termo emprestado dos antigos sistema *batch*, nos quais significava o tempo que o usuário esperava pela saída. Neste trabalho, significa o tempo de resposta médio, em segundos, desde que o usuário iniciou uma transação até conseguir alcançar o objetivo desejado, contabilizando, deste modo, enventuais abortos e renícios.

Todas as simulações foram realizadas com um total de 1000 (mil) transações.

Com relação à política de bloqueio, foi utilizado o esquema *abortar-reiniciar* (em inglês, *no-wait*) em caso de não obtenção de um bloqueio, sem espera para reiniciar (min*_restart_time e max_restart_time,* ambos iguais a zero). Para garantir um escalonamento serializável, foi empregado o protocolo de bloqueio em duas fases estrito (*strict 2PL*) [Bernstein e Newcomer, 2009].

Quanto a granularidade da política de bloqueio, temos os seguintes cenários:

- Simulação **NO_MGL** (monogranular)

  Objetivo:

  Determinar qual dos quatro grânulos tem o melhor desempenho, quando usados isoladamente (protocolo monogranular).

  Descrição:

  Simular os grânulos em separado. Utilizando o protocolo monogranular, consiste em todas as transações requisitarem bloqueios no mesmo tipo de grânulo, para observar qual grânulo tem o o melhor desempenho separadamente. O *Lock Manager* opera no modo de operação monogranular, NO_MGL, onde não há propagação de bloqueios planejados (ou de intenção).

  Foram utilizadas as seguintes cargas (ou perfis de acesso), para cada transação:

  - Acesso a, exatamente, 0.1% do total de 30.000 (trinta mil) pares (propriedade, recurso);
  - Acesso a, exatamente, 1% do total de 30.000 (trinta mil) pares (propriedade, recurso);
  - Acesso a, exatamente, 10% do total de 30.000 (trinta mil) pares (propriedade, recurso).

  Para cada carga, foram criadas duas *suites* de simulações:

  - *Suite 1*: taxas λ de chegada das transações (t/s): [10, 100, ∞]; porcentagem de bloqueios de escrita: 80%.
  - *Suite 2*: taxas λ de chegada das transações (t/s): [10, 100, ∞]; porcentagem de bloqueios de escrita: 20%.

  Em cada *suite*, para cada taxa de chegada (t/s), simulou-se *a comparação entre os grânulos, separadamente*. Usando os novos tipos de bloqueio, de inserção e remoção, quatro simulações, onde em cada simulação, todas as transações bloqueiam exatamente o mesmo tipo de grânulo ("*Graph*", "*Property*", "*Resource*", "*Property of Resource*"), em separado.

- Simulaçao **MGL** (multigranular)

  Objetivo:

  Comparar o desempenho do protocolo multigranular proposto com o protocolo monogranular.

  Descrição:

  Simular o protocolo multigranular proposto. Primeiro os grânulos foram simulados separadamente de novo, usando o *Lock Manager* no modo NO_MGL (monogranular). Em seguida, as mesmas transações foram simuladas com o protocolo multigranular, ou seja, usando o *Lock Manager* no modo MGL. Na simulação multigranular, para decidir qual grânulo a transação deve bloquear é necessário decidir quando uma transação é considerada abrangente o suficiente para bloquear todo o grânulo. Isto se traduz em qual porcentagem mínima do total de pares (propriedade, recurso) "contidos" no grânulo, uma transação deve acessar para então bloquear todo o grânulo. Para este propósito, inspirado pelo experimento realizado em [Ries e Stonebraker, 1979], foram definidos alguns *threshold percentages* (tp) (percentuais de limiar) para se bloquear todo o grânulo. O algoritmo tenta bloquear do maior grânulo ("*Graph*") para o menor ("*Property of Resource*"), de acordo com estes *thresholds*.

  Foi utilizada a seguinte carga (ou perfil de acesso), para cada transação:

  - Mix de transações de vários tamanhos, geradas aleatoriamente. Em um total de 30.000 (trinta mil) pares (propriedade, recurso), para cada transação foi sorteado um valor entre 30 (trinta) e 3.000 (três mil) para determinar o número de pares a serem acessados.

Foram criadas duas *suites* de simulações:

- *Suite 1*: taxas λ de chegada das transações (t/s): [10, 100, ∞]; porcentagem de bloqueios de escrita: 80%.
- *Suite 2*: taxas λ de chegada das transações (t/s): [10, 100, ∞]; porcentagem de bloqueios de escrita: 20%.

Em cada *suite,* simulou-se:

- *Comparação entre os grânulos, separadamente*. Usando os novos tipos de bloqueio, de inserção e remoção, quatro simulações, onde em cada simulação, todas as transações bloqueiam exatamente o mesmo tipo de grânulo ("*Graph*", "*Property*", "*Resource*", "*Property of Resource*"), em separado.

- *Upgrade de Grânulo*. De acordo com certos *threshold percentages* (tp), são escolhidos os grânulos, do maior grânulo ("*Graph*") para o menor ("*Property of Resource*"), que a transação irá bloquear. Os seguinte *tps* foram utilizados: 0.5%, 1%, 5%, 10%, 15%, 20%, 25%, 30% e 50%.

Com relação aos novos tipos de bloqueio (*lock modes*) de leitura e escrita, para inserção e remoção, temos:

Objetivo:

Comparar o desempenho (nível de multiprogramação) dos novos tipo de bloqueio de leitura e escrita, para inserção e remoção, compatíveis entre si, com o dos tipos de bloqueio convencionais, de leitura e escrita, incompatíveis.

Descrição:

Comparação entre novos tipos de bloqueio e os tipos de bloqueio tradicionais, usando o protocolo monogranular (modo NO_MGL do *Lock Manager*) e como único grânulo, usado por todas as transações, o menor grânulo: "*Property of Resource*".

Foram utilizadas as seguintes cargas (ou perfis de acesso), para cada transação:

- Acesso a, exatamente, 0.1% do total de 30.000 (trinta mil) pares (propriedade, recurso);
- Acesso a, exatamente, 1% do total de 30.000 (trinta mil) pares (propriedade, recurso);
- Acesso a, exatamente, 10% do total de 30.000 (trinta mil) pares (propriedade, recurso).

Para cada carga, foram criadas duas *suites* de simulações:

- *Suite 1*: taxas λ de chegada das transações (t/s): [10, 100, ∞]; porcentagem de bloqueios de escrita: 80%.
- *Suite 2*: taxas λ de chegada das transações (t/s): [10, 100, ∞]; porcentagem de bloqueios de escrita: 20%.

Em cada suite, para cada taxa de chegada (t/s), simulou-se *a comparação entre novos tipos de bloqueio e os tipos de bloqueio tradicionais*. Duas simulações, onde todas as transações bloqueiam exatamente o mesmo tipo de grânulo ("*Property of Resource*"). Na primeira simulação, é usado o esquema tradicional (relacional) - bloqueios de leitura e escrita incompatíveis (riR e riW) e, na segunda, o novo esquema proposto - bloqueios de leitura e escrita compatíveis (rR e iW).

A.1.2. Ambiente da Simulação

O ambiente de simulação foi:

- **Máquina**
  - Processador: Intel Core i7 920 @ 2.67GHz;
  - Memória RAM: 6GB (DDR-3 1064 MHz);
  - Placa mãe: Intel DX58SO;
  - Tipo de sistema: 64-bit.

- **Sistema Operacional**
  - Microsoft Windows Vista Ultimate 64-bit - Service Pack 2.

- **Interpretador da Linguagem Ruby**
  - JRuby 1.6.5 (ruby-1.8.7-p330) 64-bit (2011-10-25 9dcd388).

- **Máquina Virtual Java**
  - Java HotSpot 64-Bit Server VM 1.7.0_01.

Foi escolhido o JRuby, como interpretador Ruby, porque a implementação C padrão do Ruby 1.8 usa apenas um único thread nativo e roda todos os threads Ruby dentro deste único thread nativo. Isto significa que no Ruby 1.8 threads são bastante leves, porém eles nunca rodam em paralelo, mesmo em CPUs multicore. Já o JRuby, no entanto, mapeia cada thread Ruby para um thread Java. Sendo assim, a implementação e comportamento dos threads dependem da máquina virtual java adjacente. As máquinas virtuais java modernas, como a utilizada neste trabalho, implementam seus threads como threads nativos e, desta forma, permitem processamento verdadeiramente paralelo em CPUs multicore.

A.1.3. Resultados da Simulação

Para cada um dos cenários apresentados, serão apresentados os resultados obtidos. Para cada parâmetro de saída será apresentado um gráfico para melhor visualizar os resultados.

Como vimos, as taxas λ nominais de chegada das transações (t/s) foram: 1 t/s; 10 t/s; 100 t/s e ∞ t/s. Taxa nominais porque, na verdade, as taxas reais podem variar um pouco, pois no comando `sleep` do Ruby, para esperar λ^{-1} segundos entre uma transação e outra, o thread não é bloqueado exatamente o tempo solicitado. O thread "*dorme*" por um tempo aproximado. As taxas λ reais também foram medidas e serão mostradas.

A.1.3.1.
Cenário 1: Grânulos em Separado (NO_MGL - monogranular); Acesso a 0.1% pares do total de 30.000; Suite 1 (80% bloqueios de escrita)

Parâmetros de entradas específicos:

- número de recursos: 300
- número de propriedades: 100
- min_pairs_per_req: 30
- max_pairs_per_req: 30
- *min_restart_time*: 0
- *max_restart_time*: 0

A seguir os valores observados.

Número de bloqueios:

Tabela 22 - Cenário 1 - número de bloqueios

grânulo	número de bloqueios
graph	1,000
property	25,979
resource	28,584
property of resource	30,000

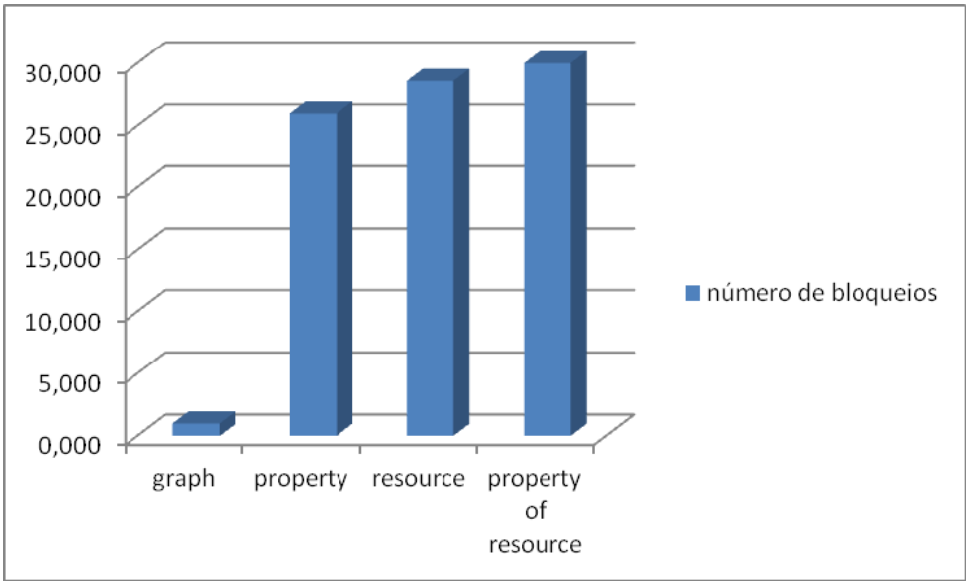

Figura 47 - Cenário 1 - número de bloqueios

Número de abortos:

Tabela 23 - Cenário 1 - número de abortos

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	0,000	0,547	315,602
property	0,000	1,357	92,934
resource	0,000	0,585	39,777
property of resource	0,000	0,013	4,781

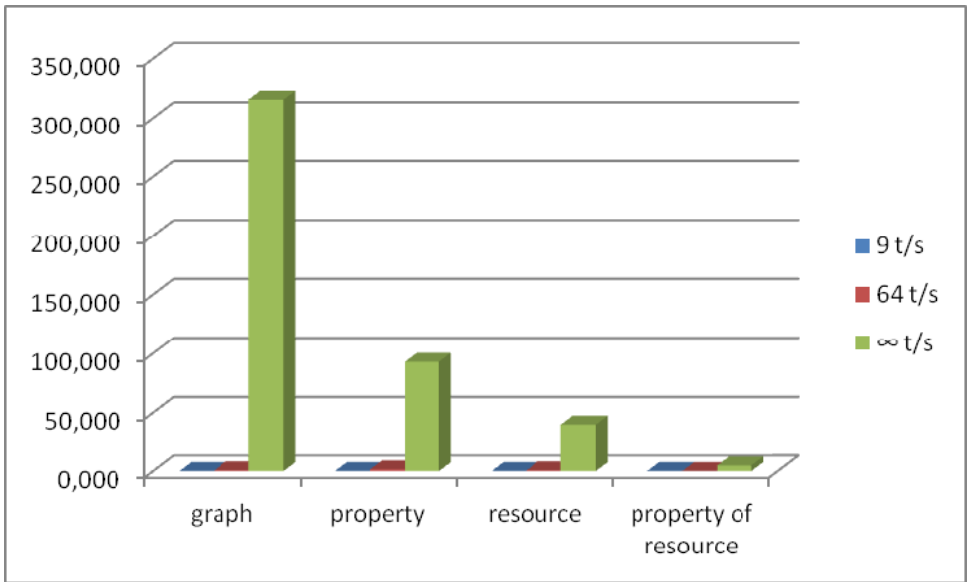

Figura 48 - Cenário 1 - número de abortos

***Turnaround* (seg):**

Tabela 24 - Cenário 1 - *turnaround* (seg)

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	0,00177	0,00036	5,89066
property	0,00040	0,00079	5,76884
resource	0,00041	0,00079	3,85834
property of resource	0,00055	0,00108	1,93812

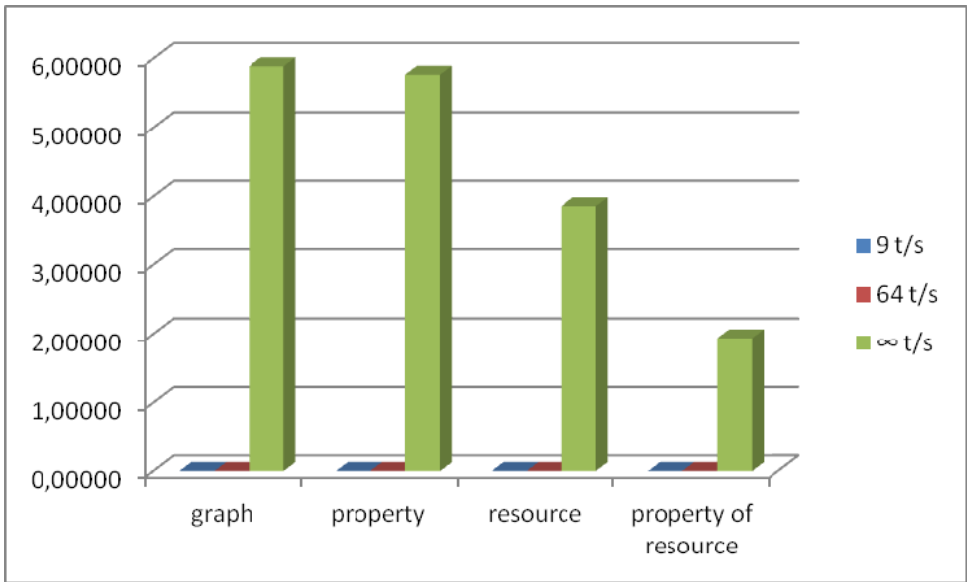

Figura 49 - Cenário 1 - *turnaround* (seg)

A.1.3.2. Cenário 2: Grânulos em Separado (NO_MGL - monogranular); Acesso a 0.1% pares do total de 30.000; Suite 2 (20% bloqueios de escrita)

Parâmetros de entrada específicos:

- número de recursos: 300
- número de propriedades: 100
- min_pairs_per_req: 30
- max_pairs_per_req: 30
- *min_restart_time*: 0
- *max_restart_time*: 0

A seguir, os valores observados.

Número de bloqueios:

Tabela 25 - Cenário 2 - número de bloqueios

grânulo	número de bloqueios
graph	1,000
property	26,040
resource	28,588
property of resource	30,000

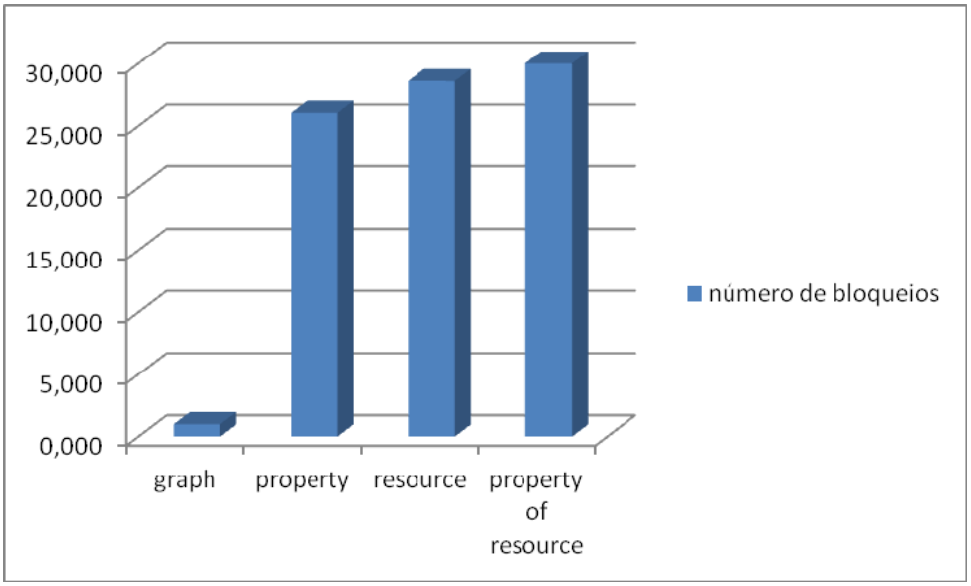

Figura 50 - Cenário 2 - número de bloqueios

Número de abortos:

Tabela 26 - Cenário 2 - número de abortos

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	0,000	3,280	350,721
property	0,000	0,436	40,698
resource	0,000	0,765	13,136
property of resource	0,000	0,000	4,464

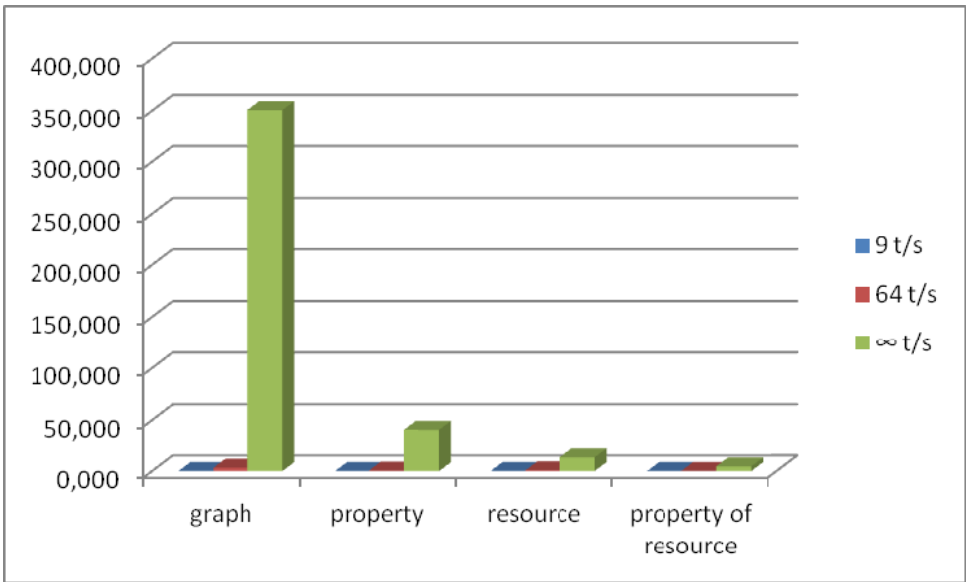

Figura 51 - Cenário 2 - número de abortos

***Turnaround* (seg):**

Tabela 27 - Cenário 2 - *turnaround* (seg)

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	0,00016	0,00045	6,55782
property	0,00036	0,00101	3,90785
resource	0,00045	0,00127	2,13097
property of resource	0,00034	0,00057	2,10405

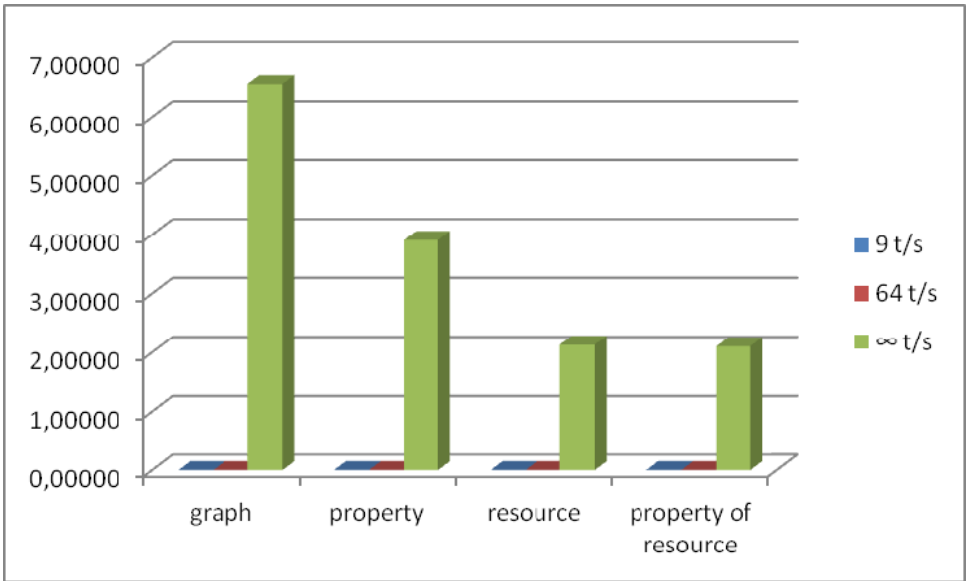

Figura 52 - Cenário 2 - *turnaround* (seg)

A.1.3.3. Cenário 3: Grânulos em Separado (NO_MGL - monogranular); Acesso a 1% pares do total de 30.000; Suite 1 (80% bloqueios de escrita)

Parâmetros de entradas específicos:

- número de recursos: 300
- número de propriedades: 100
- min_pairs_per_req: 300
- max_pairs_per_req: 300
- *min_restart_time*: 0
- *max_restart_time*: 0

A seguir os valores observados.

Número de bloqueios:

Tabela 28 - Cenário 3 - número de bloqueios

grânulo	número de bloqueios
graph	1,000
property	95,225
resource	190,447
property of resource	300,000

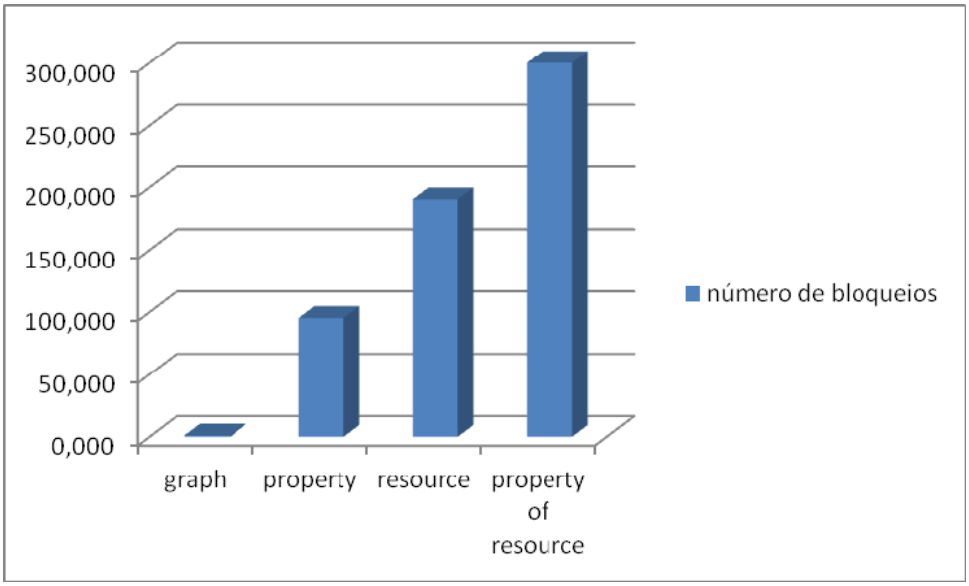

Figura 53 - Cenário 3 - número de bloqueios

Número de abortos:

Tabela 29 - Cenário 3 - número de abortos

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	8,968	2.609,449	2.620,611
property	14,189	2.343,980	2.363,574
resource	11,082	1.535,956	1.537,825
property of resource	3,199	36,105	35,214

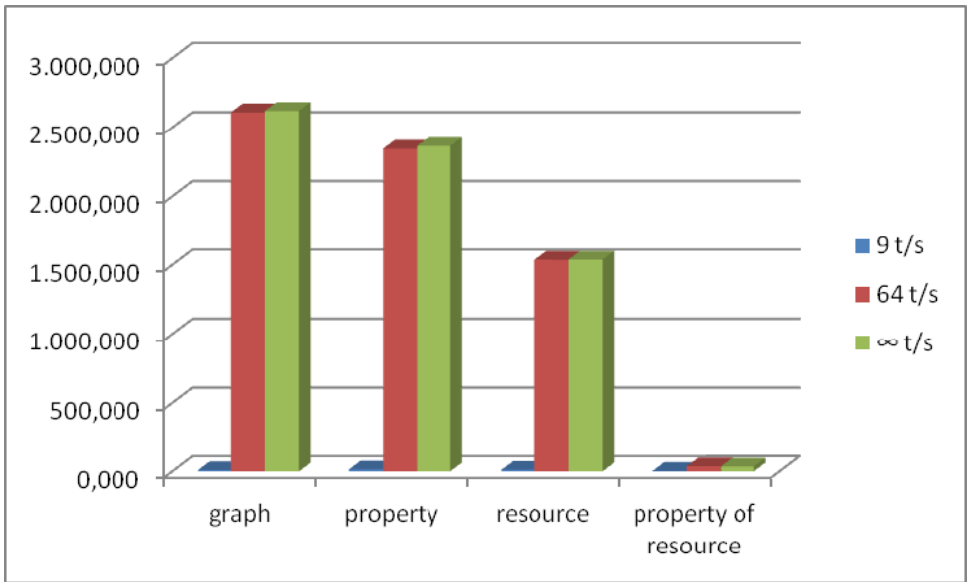

Figura 54 - Cenário 3 - número de abortos

***Turnaround* (seg):**

Tabela 30 - Cenário 3 - *turnaround* (seg)

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	0,06414	39,13390	47,76607
property	0,06423	39,79464	47,69819
resource	0,06594	35,99384	43,91385
property of resource	0,07030	21,23356	28,76855

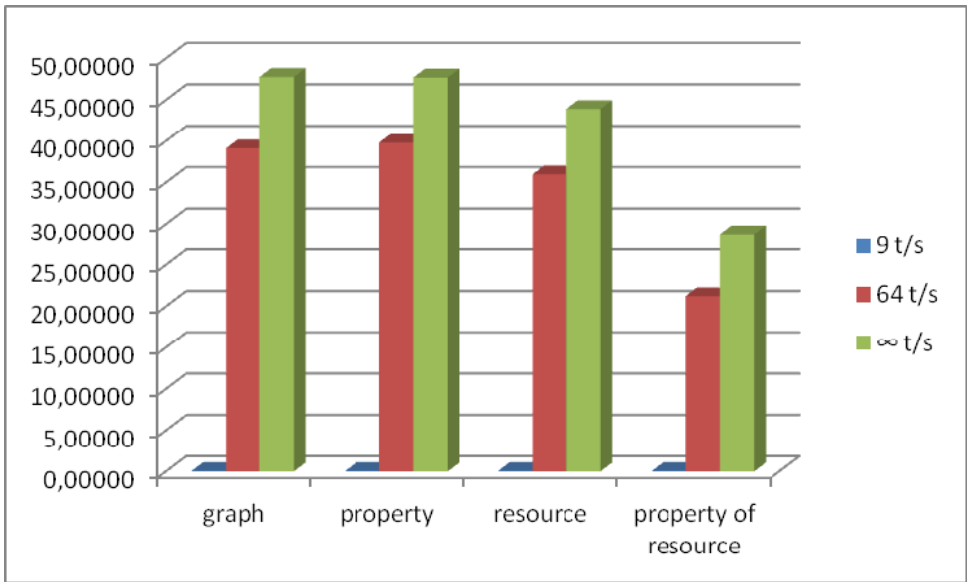

Figura 55 - Cenário 3 - *turnaround* (seg)

A.1.3.4. Cenário 4: Grânulos em Separado (NO_MGL - monogranular); Acesso a 1% pares do total de 30.000; Suite 2 (20% bloqueios de escrita)

Parâmetros de entradas específicos:

- número de recursos: 300
- número de propriedades: 100
- min_pairs_per_req: 300
- max_pairs_per_req: 300
- *min_restart_time*: 0
- *max_restart_time*: 0

A seguir os valores observados.

Número de bloqueios:

Tabela 31 - Cenário 4 - número de bloqueios

grânulo	número de bloqueios
graph	1,000
property	95,270
resource	190,398
property of resource	300,000

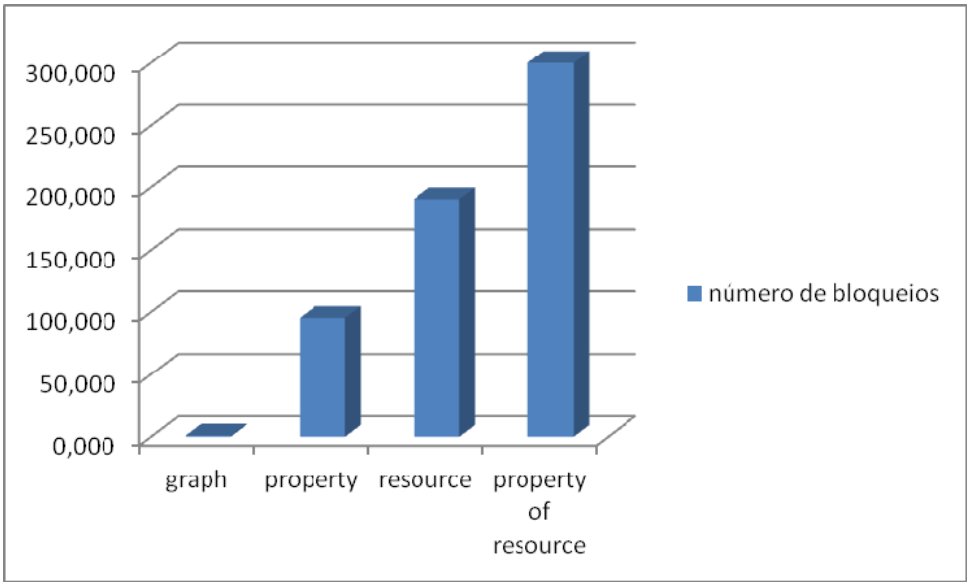

Figura 56 - Cenário 4 - número de bloqueios

Número de abortos:

Tabela 32 - Cenário 4 - número de abortos

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	12,643	2.691,941	2.737,042
property	0,128	1.801,948	1.824,026
resource	0,552	813,861	879,726
property of resource	0,338	11,051	13,500

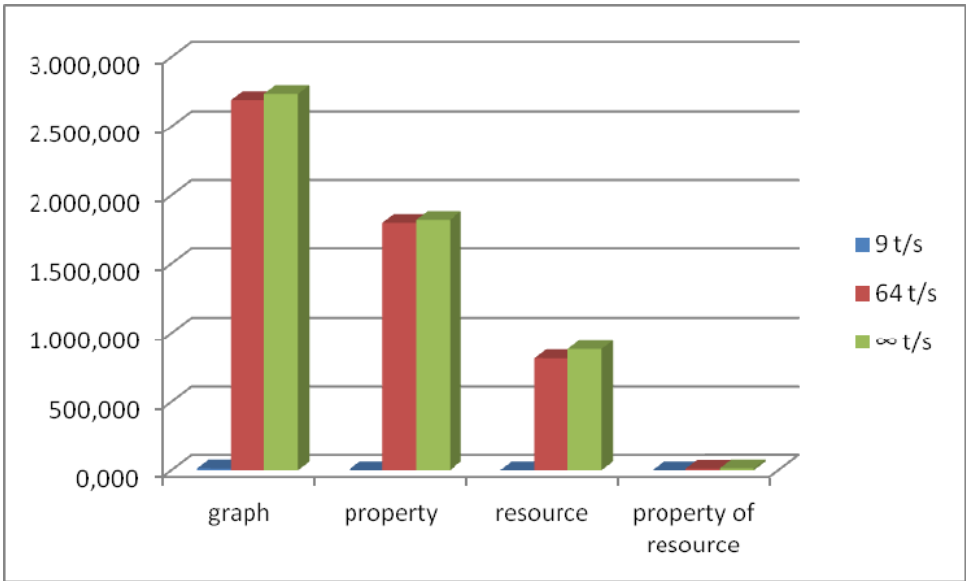

Figura 57 - Cenário 4 - número de abortos

***Turnaround* (seg):**

Tabela 33 - Cenário 4 - *turnaround* (seg)

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	0,066	40,749	49,769
property	0,065	36,822	44,846
resource	0,066	36,204	44,247
property of resource	0,084	11,603	18,575

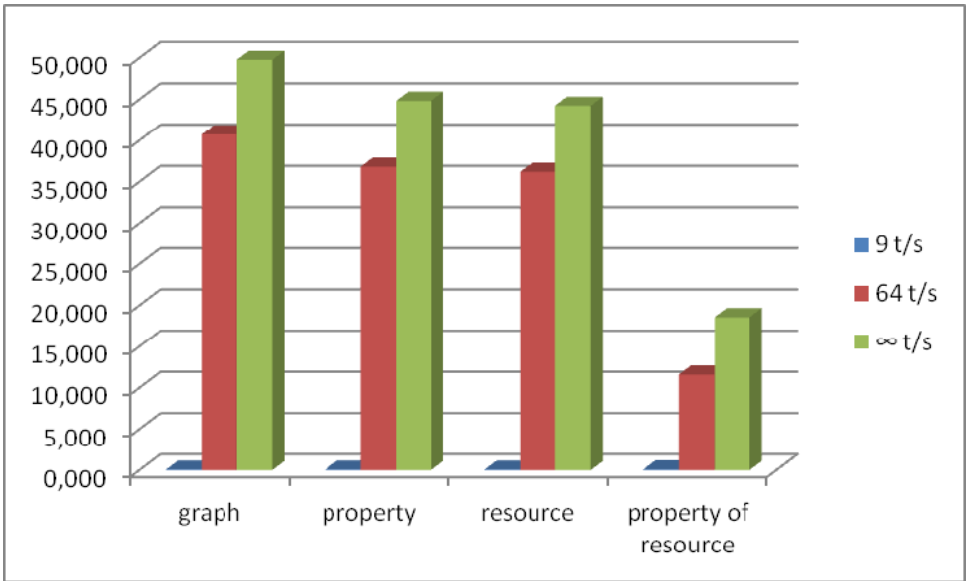

Figura 58 - Cenário 4 - *turnaround* (seg)

A.1.3.5. Cenário 5: Grânulos em Separado (NO_MGL - monogranular); Acesso a 10% pares do total de 30.000; Suite 1 (80% bloqueios de escrita)

Parâmetros de entradas específicos:

- número de recursos: 300
- número de propriedades: 100
- min_pairs_per_req: 3000
- max_pairs_per_req: 3000
- *min_restart_time*: 0
- *max_restart_time*: 0

A seguir os valores observados.

Número de bloqueios:

Tabela 34 - Cenário 5 - número de bloqueios

grânulo	número de bloqueios
graph	1,000
property	100,000
resource	299,991
property of resource	3.000,000

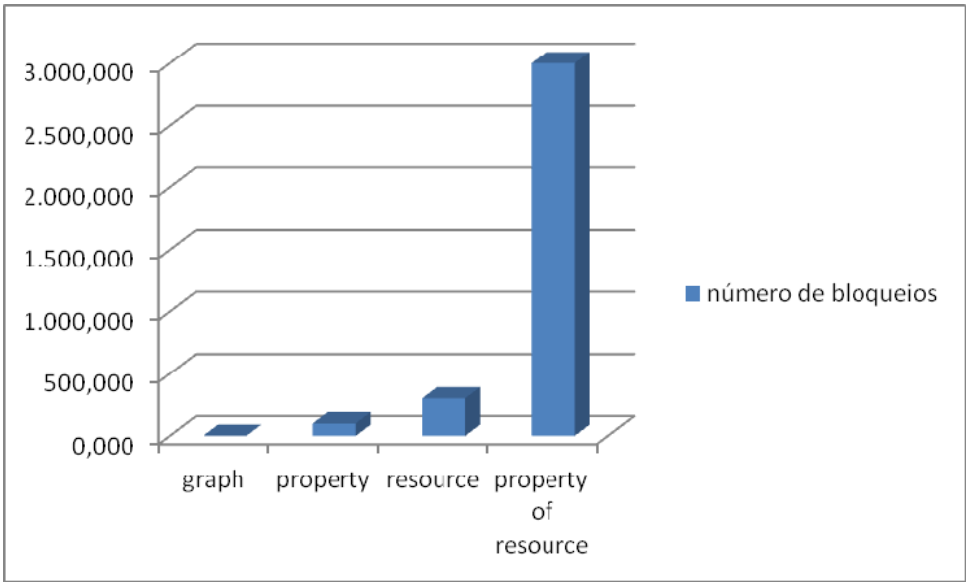

Figura 59 - Cenário 5 - número de bloqueios

Número de abortos:

Tabela 35 - Cenário 5 - número de abortos

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	25.321,671	25.788,277	26.031,912
property	23.226,871	23.202,492	23.098,477
resource	22.894,949	22.803,396	23.173,156
property of resource	3.126,661	3.145,798	3.184,074

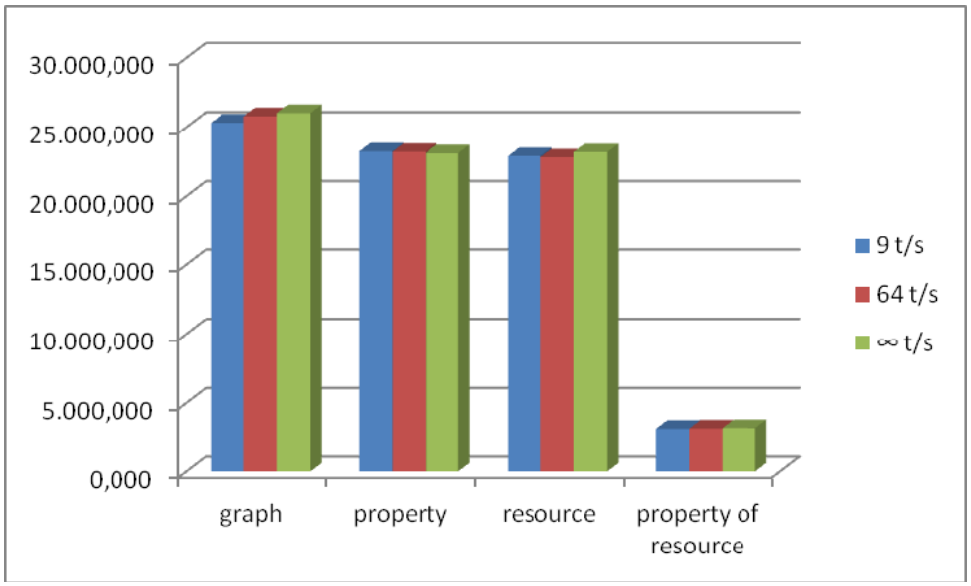

Figura 60 - Cenário 5 - número de abortos

***Turnaround* (seg):**

Tabela 36 - Cenário 5 - *turnaround* (seg)

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	415,008	470,009	483,826
property	406,129	451,879	461,469
resource	416,250	463,896	477,859
property of resource	438,402	488,584	503,732

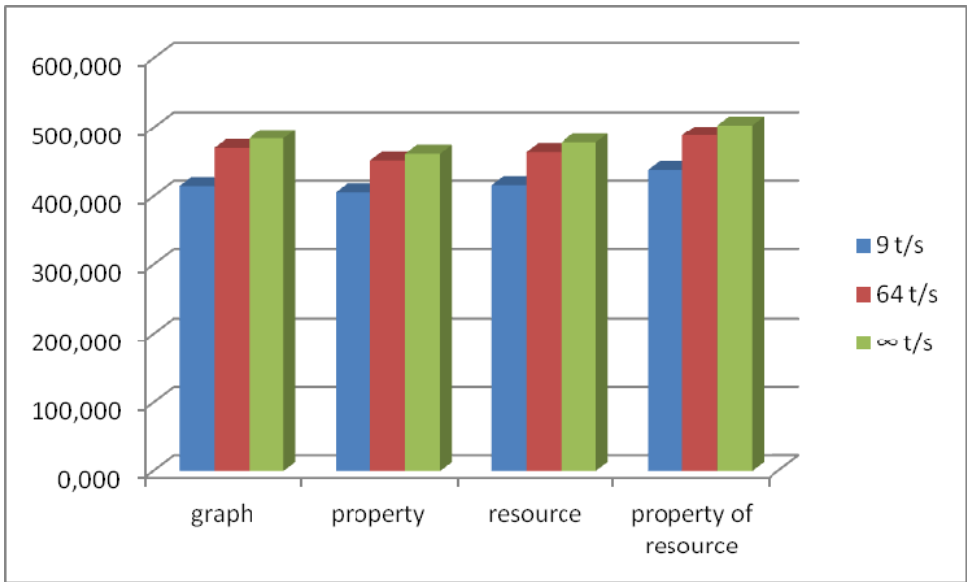

Figura 61 - Cenário 5 - *turnaround* (seg)

A.1.3.6. Cenário 6: Grânulos em Separado (NO_MGL - monogranular); Acesso a 10% pares do total de 30.000; Suite 2 (20% bloqueios de escrita)

Parâmetros de entradas específicos:

- número de recursos: 300
- número de propriedades: 100
- min_pairs_per_req: 3000
- max_pairs_per_req: 3000
- *min_restart_time*: 0
- *max_restart_time*: 0

A seguir os valores observados.

Número de bloqueios:

Tabela 37 - Cenário 6 - número de bloqueios

grânulo	número de bloqueios
graph	1,000
property	100,000
resource	299,993
property of resource	3.000,000

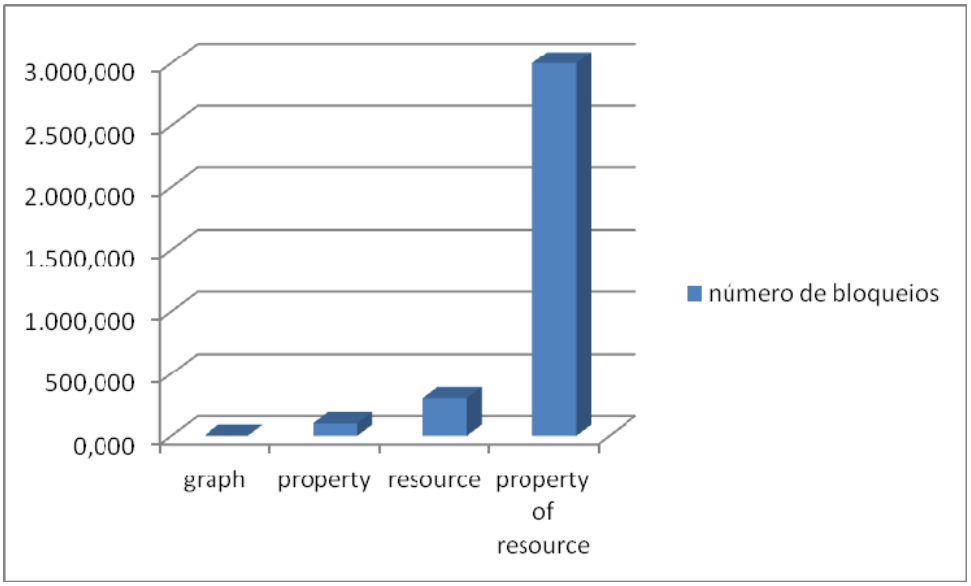

Figura 62 - Cenário 6 - número de bloqueios

Número de abortos:

Tabela 38 - Cenário 6 - número de abortos

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	31.037,390	31.568,414	31.756,039
property	28.863,269	28.546,797	29.068,542
resource	27.402,433	27.404,148	27.482,897
property of resource	1.466,712	1.458,063	1.429,412

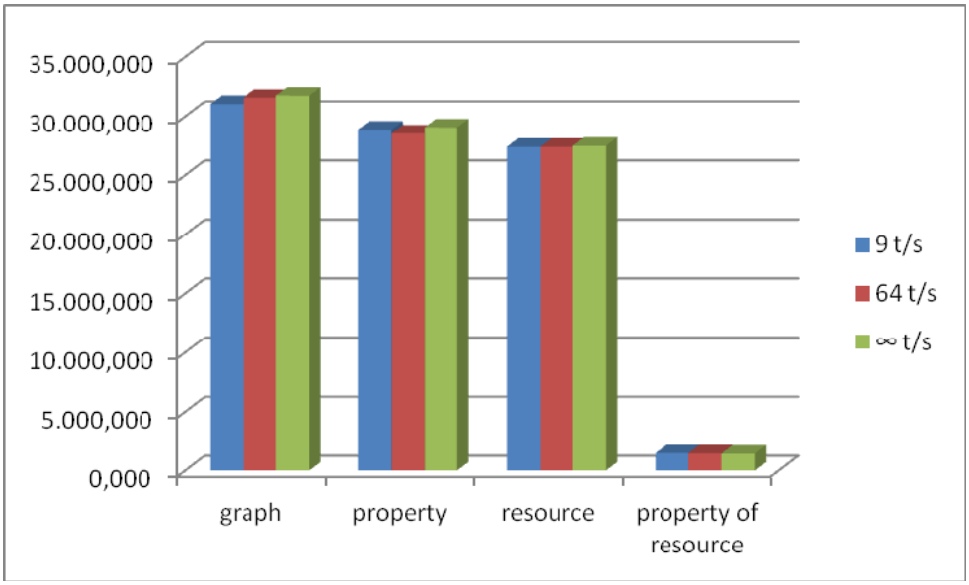

Figura 63 - Cenário 6 - número de abortos

***Turnaround* (seg):**

Tabela 39 - Cenário 6 - *turnaround* (seg)

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	524,049	580,330	591,029
property	513,183	556,781	573,895
resource	521,472	570,484	581,298
property of resource	531,156	582,844	596,157

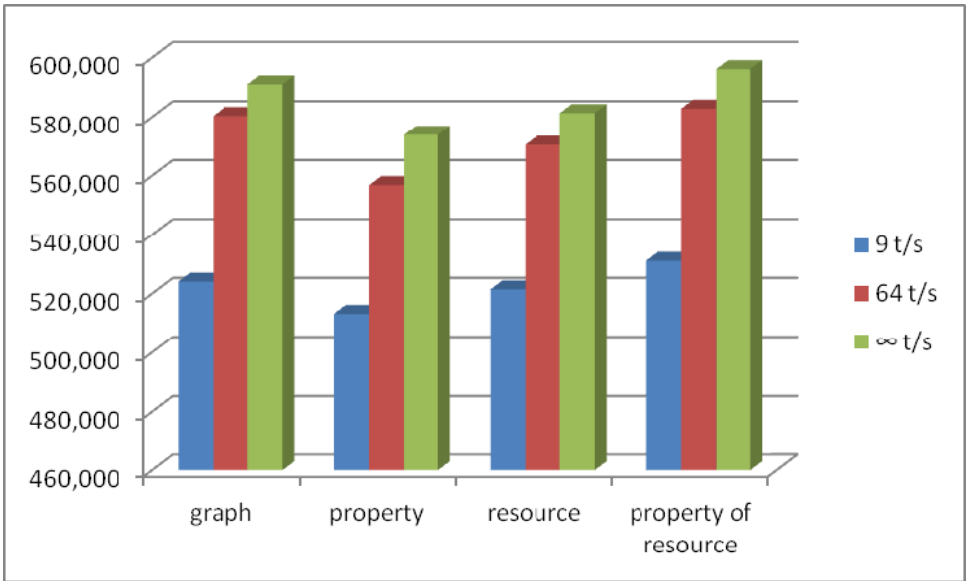

Figura 64 - Cenário 6 - *turnaround* (seg)

A.1.3.7.
Cenário 7: Comparação Multigranular (MGL) com Monogranular (NO_MGL); Mix de transações de diferentes tamanhos, desde pequenas (acesso a 30 do total de 30.000 pares) até grandes (acessos a 3.000 do total de 30.000 pares); Suite 1 (80% bloqueios de escrita)

Parâmetros de entradas específicos:

- número de recursos: 300
- número de propriedades: 100
- min_pairs_per_req: 30
- max_pairs_per_req: 3000
- *min_restart_time*: 0
- *max_restart_time*: 0

A seguir os valores observados.

Número de bloqueios:

Tabela 40 - Cenário 7 - número de bloqueios

grânulo	número de bloqueios
graph	1,000
property	97,442
resource	273,064
property of resource	1.507,708
tp 0,5%	5,453
tp 1%	19,709
tp 5%	346,340
tp 10%	1.364,422
tp 15%	1.502,020
tp 20%	1.507,609
tp 25%	1.507,708
tp 30%	1.507,708
tp 50%	1.507,708

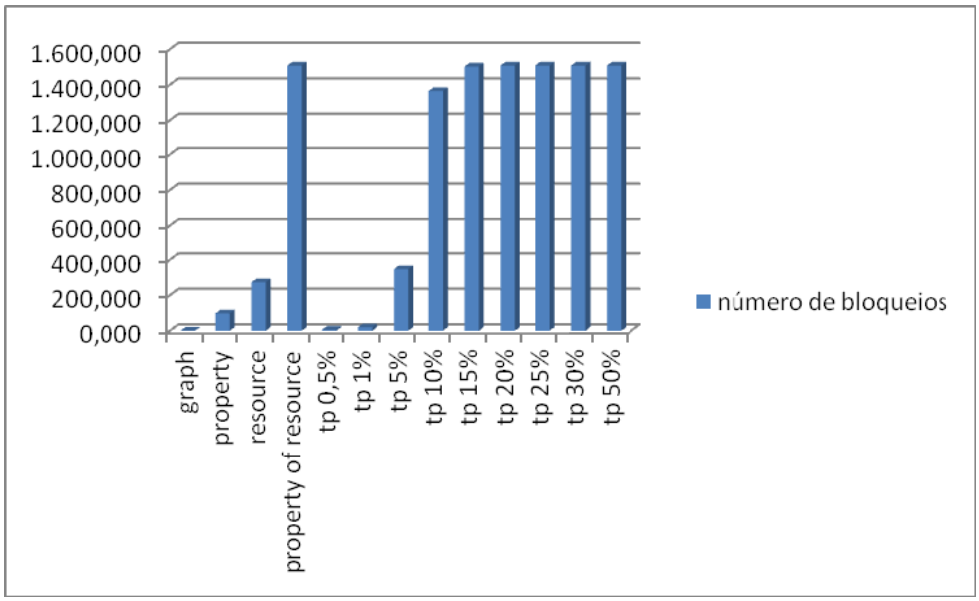

Figura 65 - Cenário 7 - número de bloqueios

Número de bloqueios por grânulo:

Tabela 41 - Cenário 7 - número de bloqueios por grânulo

grânulo \ grânulo	graph	property	resource	property of resource
graph	1,000	0,000	0,000	0,000
property	0,000	97,442	0,000	0,000
resource	0,000	0,000	273,064	0,000
property of resource	0,000	0,000	0,000	1.507,708
tp 0,5%	0,961	0,864	2,574	1,054
tp 1%	0,904	2,397	10,846	5,562
tp 5%	0,494	5,407	24,431	316,008
tp 10%	0,000	7,007	30,413	1.327,002
tp 15%	0,000	0,015	2,564	1.499,441
tp 20%	0,000	0,000	0,031	1.507,578
tp 25%	0,000	0,000	0,000	1.507,708
tp 30%	0,000	0,000	0,000	1.507,708
tp 50%	0,000	0,000	0,000	1.507,708

Número de abortos:

Tabela 42 - Cenário 7 - número de abortos

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	13.843,891	13.915,868	13.949,295
property	12.559,592	12.506,651	12.575,762
resource	12.540,781	12.460,156	12.454,584
property of resource	1.265,864	1.258,731	1.260,986
tp 0,5%	11.513,518	11.484,292	11.648,696
tp 1%	11.625,276	11.697,516	11.911,930
tp 5%	10.261,288	10.023,589	10.110,038
tp 10%	1.401,557	1.424,941	1.415,417
tp 15%	467,126	468,351	469,334
tp 20%	399,201	397,185	395,257
tp 25%	402,401	395,927	398,151
tp 30%	419,068	403,668	414,941
tp 50%	416,772	412,192	415,707

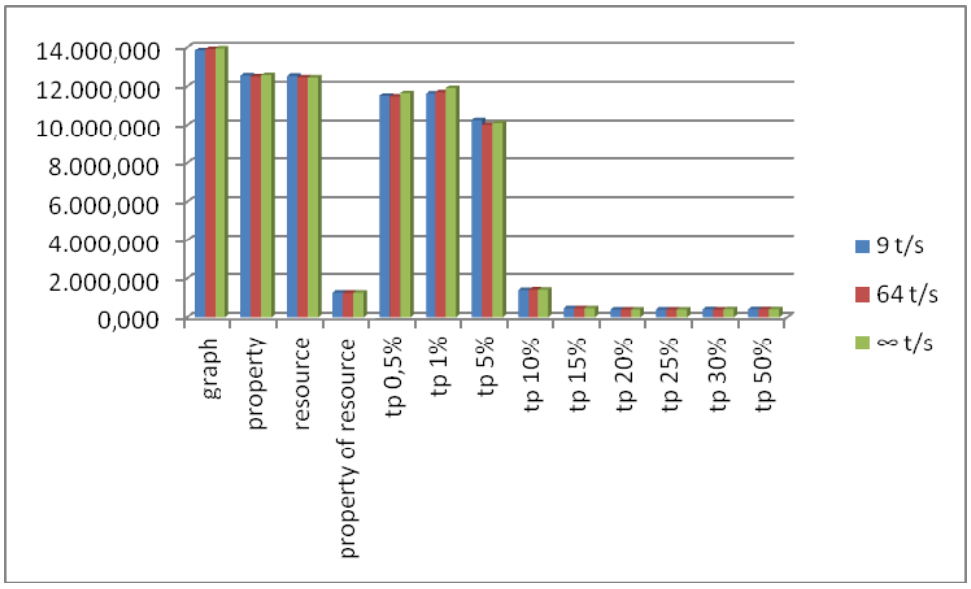

Figura 66 - Cenário 7 - número de abortos

***Turnaround* (seg):**

Tabela 43 - Cenário 7 - *turnaround* (seg)

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	213,731	263,138	280,221
property	209,948	259,411	264,391
resource	213,261	258,077	265,099
property of resource	188,341	231,705	243,619
tp 0,5%	169,717	220,051	227,311
tp 1%	166,662	213,342	221,447
tp 5%	133,722	178,911	188,609
tp 10%	192,848	233,167	245,632
tp 15%	193,418	238,294	252,476
tp 20%	198,777	237,325	251,655
tp 25%	203,754	246,178	254,185
tp 30%	204,263	253,846	259,075
tp 50%	204,315	254,531	258,807

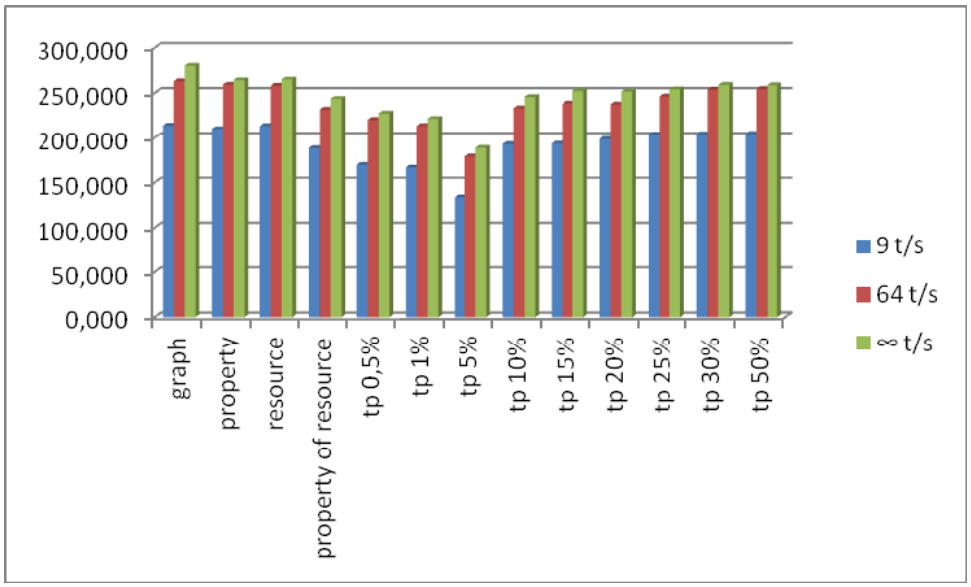

Figura 67 - Cenário 7 - *turnaround* (seg)

A.1.3.8.
Cenário 8: Comparação Multigranular (MGL) com Monogranular (NO_MGL); Mix de transações de diferentes tamanhos, desde pequenas (acesso a 30 do total de 30.000 pares) até grandes (acessos a 3.000 do total de 30.000 pares); Suite 2 (20% bloqueios de escrita)

Parâmetros de entradas específicos:

- número de recursos: 300
- número de propriedades: 100
- min_pairs_per_req: 30
- max_pairs_per_req: 3000
- *min_restart_time*: 0
- *max_restart_time*: 0

A seguir os valores observados.

Número de bloqueios:

Tabela 44 - Cenário 8 - número de bloqueios

grânulo	número de bloqueios
graph	1,000
property	97,644
resource	274,151
property of resource	1.513,928
tp 0,5%	3,669
tp 1%	9,480
tp 5%	274,553
tp 10%	1.215,797
tp 15%	1.485,296
tp 20%	1.513,298
tp 25%	1.513,928
tp 30%	1.513,928
tp 50%	1.513,928

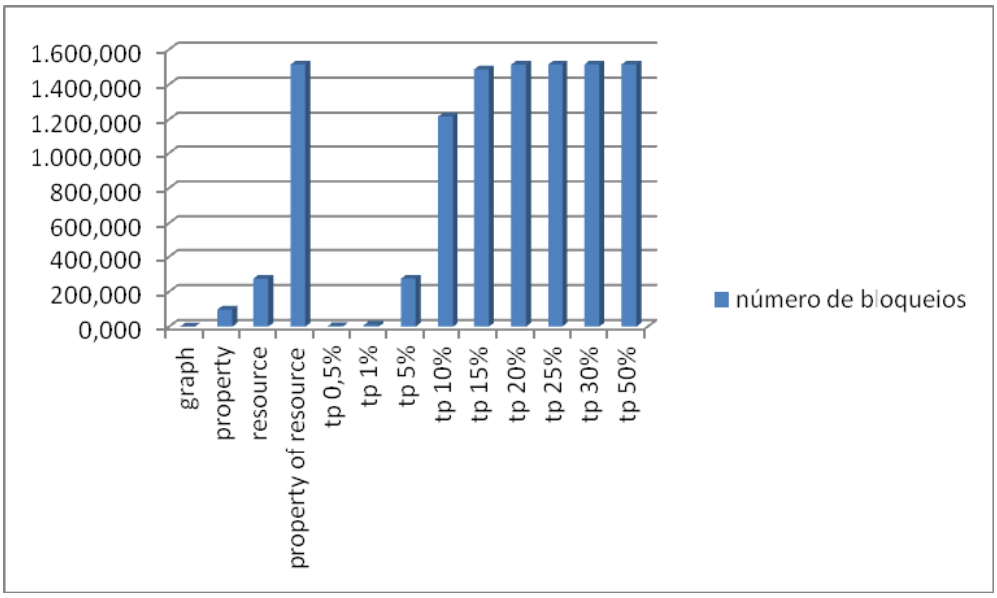

Figura 68 - Cenário 8 - número de bloqueios

Número de bloqueios por grânulo:

Tabela 45 - Cenário 8 - número de bloqueios por grânulo

grânulo \ grânulo	graph	property	resource	property of resource
graph	1,000	0,000	0,000	0,000
property	0,000	97,644	0,000	0,000
resource	0,000	0,000	274,151	0,000
property of resource	0,000	0,000	0,000	1.513,928
tp 0,5%	0,961	0,832	1,611	0,265
tp 1%	0,922	1,724	5,757	1,077
tp 5%	0,501	5,160	14,241	254,651
tp 10%	0,000	6,871	15,885	1.193,041
tp 15%	0,000	0,010	2,412	1.482,874
tp 20%	0,000	0,000	0,042	1.513,256
tp 25%	0,000	0,000	0,000	1.513,928
tp 30%	0,000	0,000	0,000	1.513,928
tp 50%	0,000	0,000	0,000	1.513,928

Número de abortos:

Tabela 46 - Cenário 8 - número de abortos

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	16.269,115	16.164,750	16.263,458
property	14.426,436	14.503,232	14.395,705
resource	13.119,081	13.130,047	13.155,648
property of resource	531,504	528,274	529,361
tp 0,5%	14.299,030	14.496,682	14.419,830
tp 1%	14.332,785	14.390,239	14.696,408
tp 5%	12.232,059	12.138,784	11.962,513
tp 10%	940,360	935,494	958,842
tp 15%	284,259	283,061	291,763
tp 20%	198,577	177,476	176,398
tp 25%	176,214	176,929	178,255
tp 30%	182,582	176,470	184,089
tp 50%	182,251	184,802	183,857

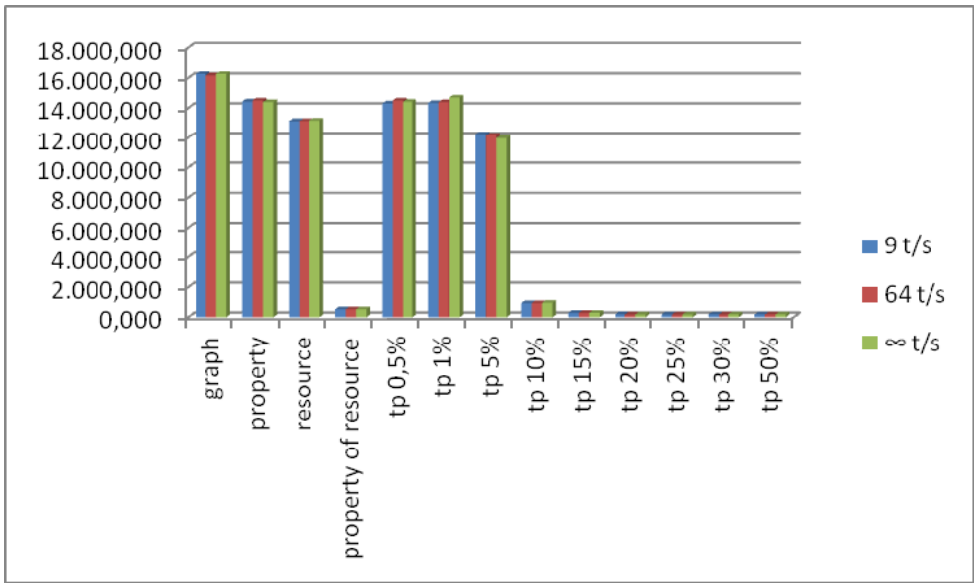

Figura 69 - Cenário 8 - número de abortos

***Turnaround* (seg):**

Tabela 47 - Cenário 8 - *turnaround* (seg)

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	269,399	321,629	323,283
property	258,963	308,493	314,091
resource	251,558	300,413	311,872
property of resource	217,115	255,060	265,526
tp 0,5%	228,620	278,640	280,130
tp 1%	217,019	269,362	276,495
tp 5%	164,613	208,683	216,569
tp 10%	209,283	257,349	266,314
tp 15%	220,079	261,712	269,571
tp 20%	222,799	261,011	273,522
tp 25%	219,233	262,053	273,575
tp 30%	222,905	275,319	279,879
tp 50%	227,616	273,113	283,041

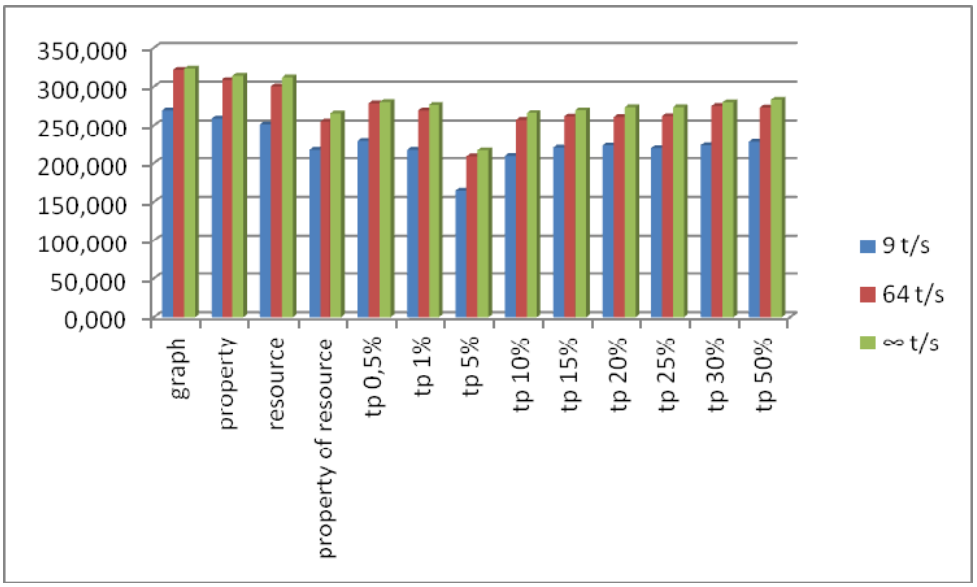

Figura 70 - Cenário 8 - *turnaround* (seg)

A.1.3.9.
Cenário 9: Comparação entre tipos de bloqueio, de leitura e escrita, novos (compatíveis) e convencionais (incompatíveis); Protocolo Monogranular (NO_MGL); Grânulo "Property of Resource"; Acesso a 0.1% pares do total de 30.000; Suite 1 (80% bloqueios de escrita)

Parâmetros de entradas específicos:

- número de recursos: 300
- número de propriedades: 100
- min_pairs_per_req: 30
- max_pairs_per_req: 30
- *min_restart_time*: 0
- *max_restart_time*: 0

A seguir os valores observados.

Número de abortos:

Tabela 48 - Cenário 9 - número de abortos

Tipos de Bloqueio \ (tran/s)	9 t/s	62 t/s	∞ t/s
incompatíveis (riR & riW)	0,000	0,038	5,006
compatíveis (rR & iW)	0,000	0,000	5,278

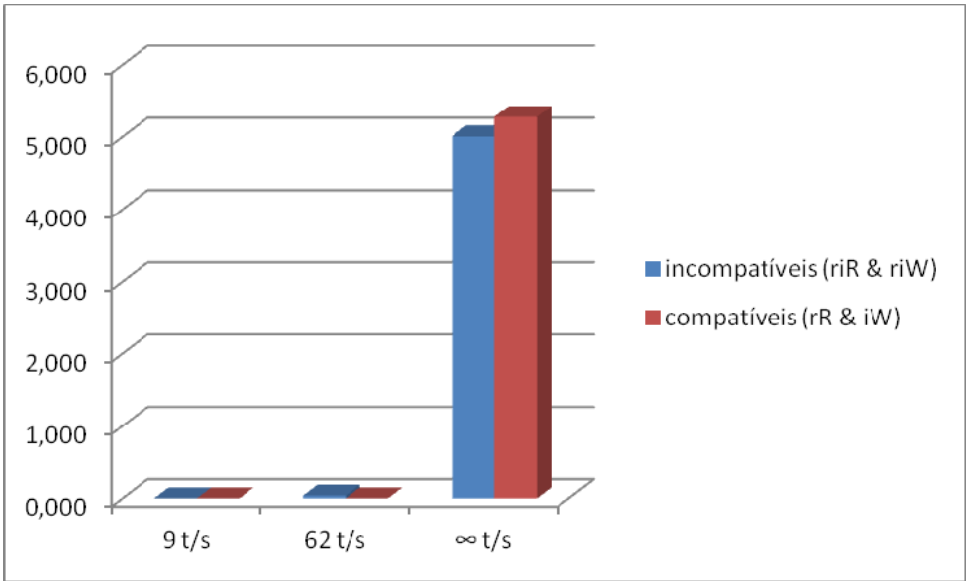

Figura 71 - Cenário 9 - número de abortos

***Turnaround* (seg):**

Tabela 49 - Cenário 9 - *turnaround* (seg)

Tipos de Bloqueio \ (tran/s)	9 t/s	62 t/s	∞ t/s
incompatíveis (riR & riW)	0,00066	0,00080	1,93568
compatíveis (rR & iW)	0,00049	0,00101	2,15079

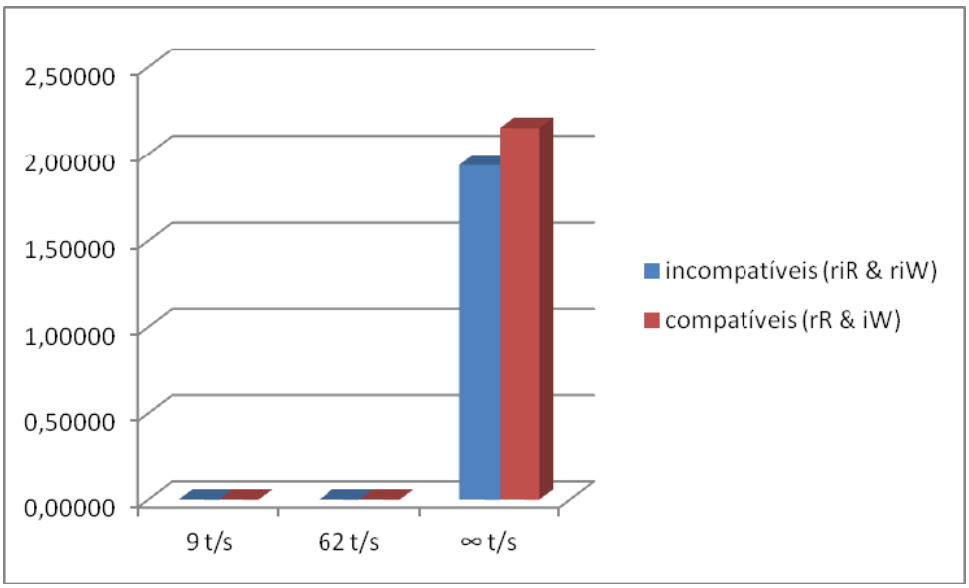

Figura 72 - Cenário 9 - *turnaround* (seg)

A.1.3.10.
Cenário 10: Comparação entre tipos de bloqueio, de leitura e escrita, novos (compatíveis) e convencionais (incompatíveis); Protocolo Monogranular (NO_MGL); Grânulo "Property of Resource"; Grânulo "Property of Resource"; Acesso a 0.1% pares do total de 30.000; Suite 2 (20% bloqueios de escrita)

Parâmetros de entradas específicos:

- número de recursos: 300
- número de propriedades: 100
- min_pairs_per_req: 30
- max_pairs_per_req: 30
- *min_restart_time*: 0
- *max_restart_time*: 0

A seguir os valores observados.

Número de abortos:

Tabela 50 - Cenário 10 - número de abortos

Tipos de Bloqueio \ (tran/s)	9 t/s	64 t/s	∞ t/s
incompatíveis (riR & riW)	0,000	0,000	4,552
compatíveis (rR & iW)	0,000	0,000	4,308

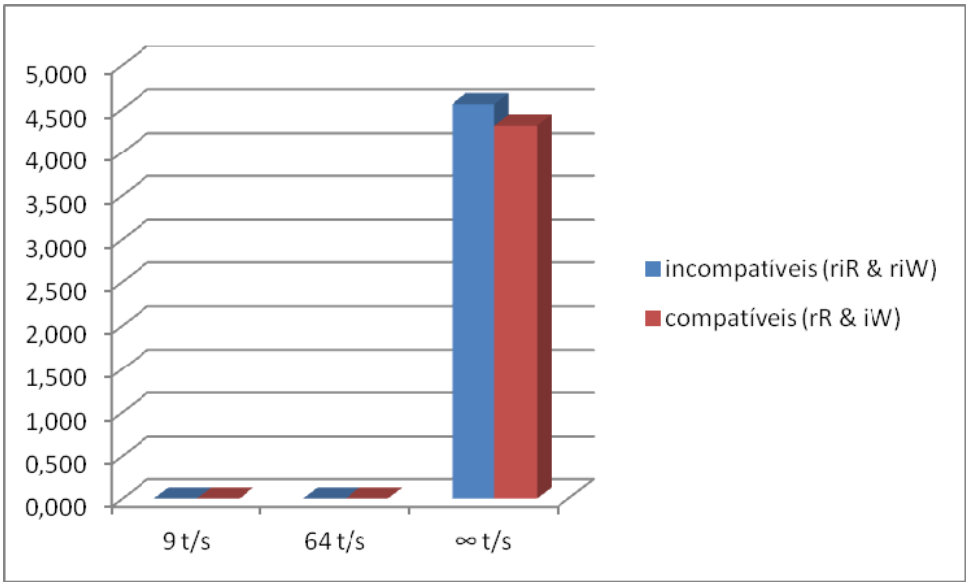

Figura 73 - Cenário 10 - número de abortos

***Turnaround* (seg):**

Tabela 51 - Cenário 10 - *turnaround* (seg)

Tipos de Bloqueio \ (tran/s)	9 t/s	64 t/s	∞ t/s
incompatíveis (riR & riW)	0,00102	0,00114	2,03213
compatíveis (rR & iW)	0,00033	0,00113	3,15627

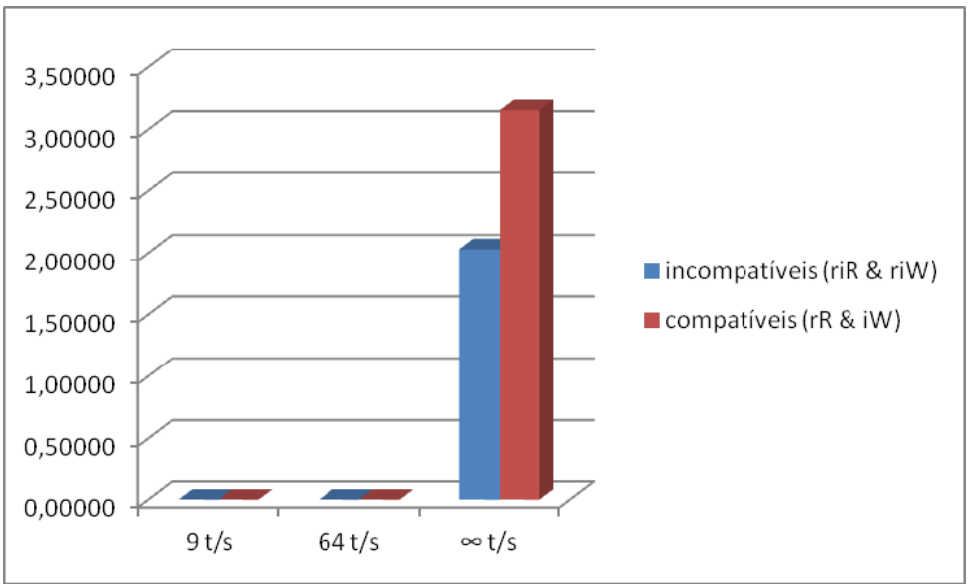

Figura 74 - Cenário 10 - turnaround (seg)

A.1.3.11.
Cenário 11: Comparação entre tipos de bloqueio, de leitura e escrita, novos (compatíveis) e convencionais (incompatíveis); Protocolo Monogranular (NO_MGL); Grânulo "Property of Resource"; Acesso a 1% pares do total de 30.000; Suite 1 (80% bloqueios de escrita)

Parâmetros de entradas específicos:

- número de recursos: 300
- número de propriedades: 100
- min_pairs_per_req: 300
- max_pairs_per_req: 300
- *min_restart_time*: 0
- *max_restart_time*: 0

A seguir os valores observados.

Número de abortos:

Tabela 52 - Cenário 11 - número de abortos

Tipos de Bloqueio \ (tran/s)	9 t/s	63 t/s	∞ t/s
incompatíveis (riR & riW)	0,576	38,378	38,893
compatíveis (rR & iW)	0,341	26,324	25,432

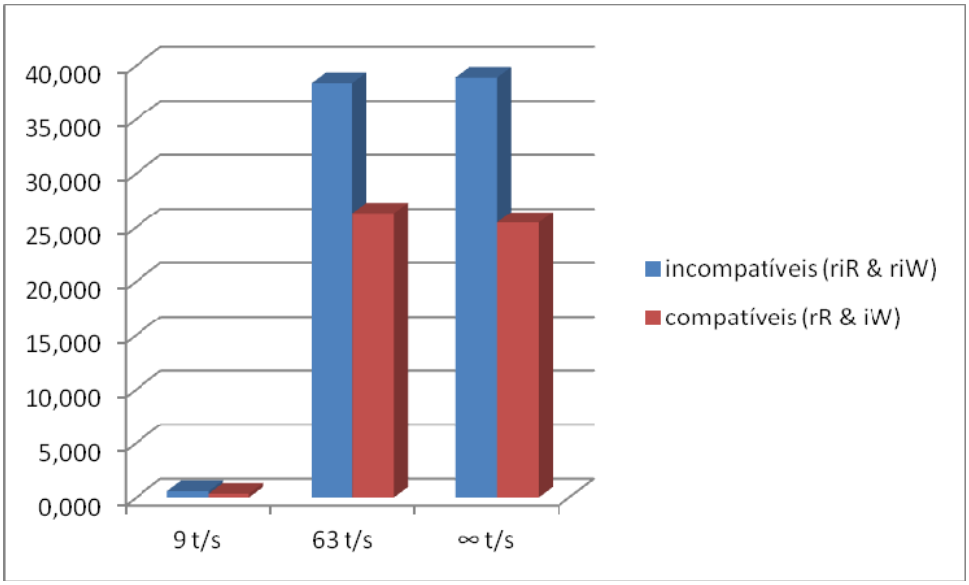

Figura 75 - Cenário 11 - número de abortos

***Turnaround* (seg):**

Tabela 53 - Cenário 11 - *turnaround* (seg)

Tipos de Bloqueio \ (tran/s)	9 t/s	63 t/s	∞ t/s
incompatíveis (riR & riW)	0,072	22,706	30,396
compatíveis (rR & iW)	0,073	17,426	25,246

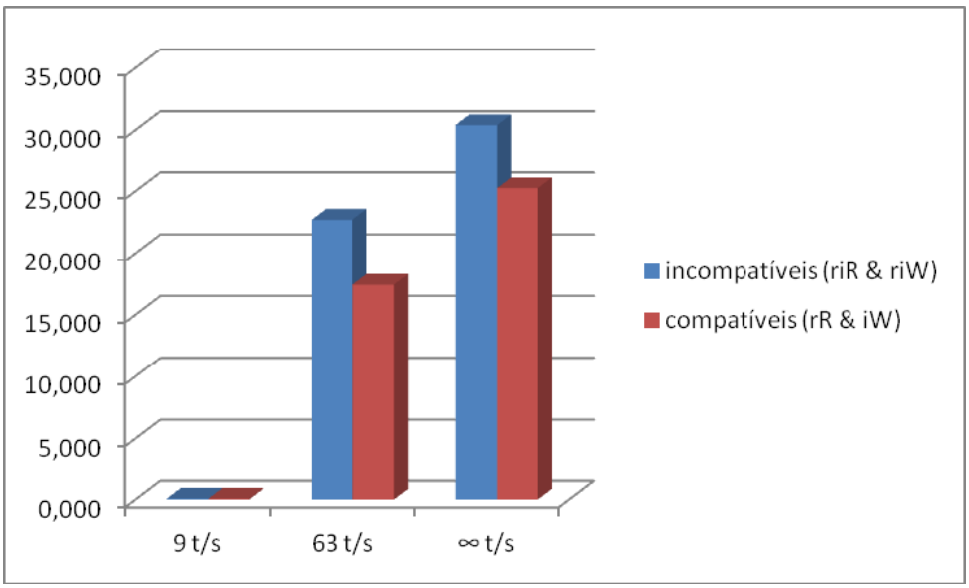

Figura 76 - Cenário 11 - *turnaround* (seg)

A.1.3.12.
Cenário 12: Comparação entre tipos de bloqueio, de leitura e escrita, novos (compatíveis) e convencionais (incompatíveis); Protocolo Monogranular (NO_MGL); Grânulo "Property of Resource"; Acesso a 1% pares do total de 30.000; Suite 2 (20% bloqueios de escrita)

Parâmetros de entradas específicos:

- número de recursos: 300
- número de propriedades: 100
- min_pairs_per_req: 300
- max_pairs_per_req: 300
- *min_restart_time*: 0
- *max_restart_time*: 0

A seguir os valores observados.

Número de abortos:

Tabela 54 - Cenário 12 - número de abortos

Tipos de Bloqueio \ (tran/s)	9 t/s	63 t/s	∞ t/s
incompatíveis (riR & riW)	0,019	14,252	13,748
compatíveis (rR & iW)	0,000	4,504	4,660

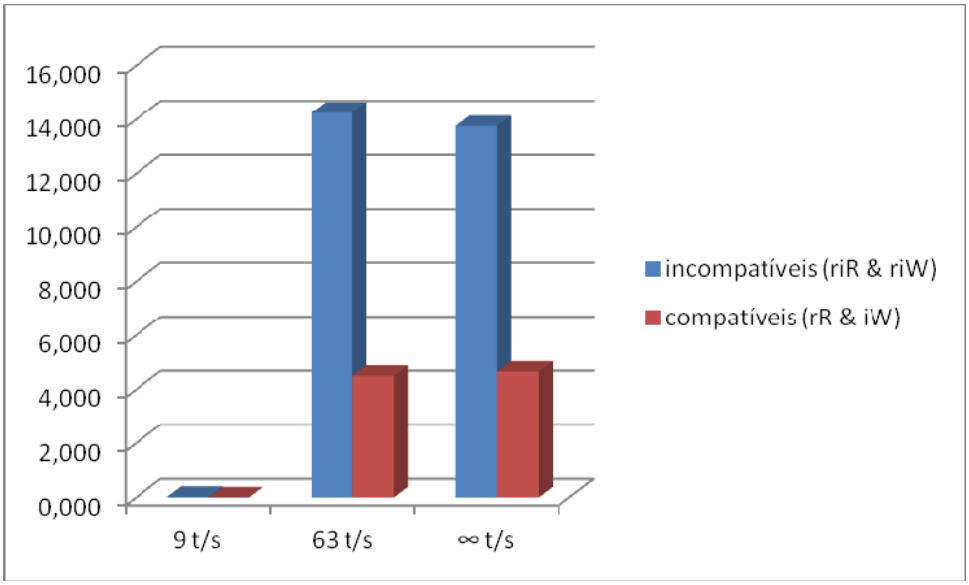

Figura 77 - Cenário 12 - número de abortos

***Turnaround* (seg):**

Tabela 55 - Cenário 12 - *turnaround* (seg)

Tipos de Bloqueio \ (tran/s)	9 t/s	63 t/s	∞ t/s
incompatíveis (riR & riW)	0,078	12,223	20,052
compatíveis (rR & iW)	0,078	9,118	17,004

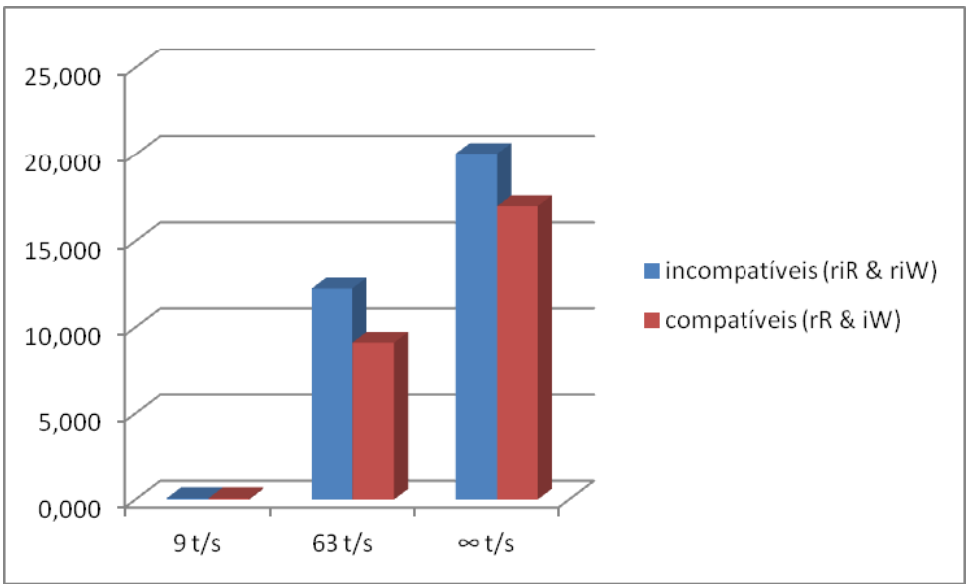

Figura 78 - Cenário 12 - turnaround (seg)

A.1.3.13.
Cenário 13: Comparação entre tipos de bloqueio, de leitura e escrita, novos (compatíveis) e convencionais (incompatíveis); Protocolo Monogranular (NO_MGL); Grânulo "Property of Resource"; Acesso a 10% pares do total de 30.000; Suite 1 (80% bloqueios de escrita)

Parâmetros de entradas específicos:

- número de recursos: 300
- número de propriedades: 100
- min_pairs_per_req: 3000
- max_pairs_per_req: 3000
- *min_restart_time*: 0
- *max_restart_time*: 0

A seguir os valores observados.

Número de abortos:

Tabela 56 - Cenário 13 - número de abortos

Tipos de Bloqueio \ (tran/s)	9 t/s	64 t/s	∞ t/s
incompatíveis (riR & riW)	3.392,136	3.447,401	3.472,456
compatíveis (rR & iW)	2.569,003	2.596,661	2.595,610

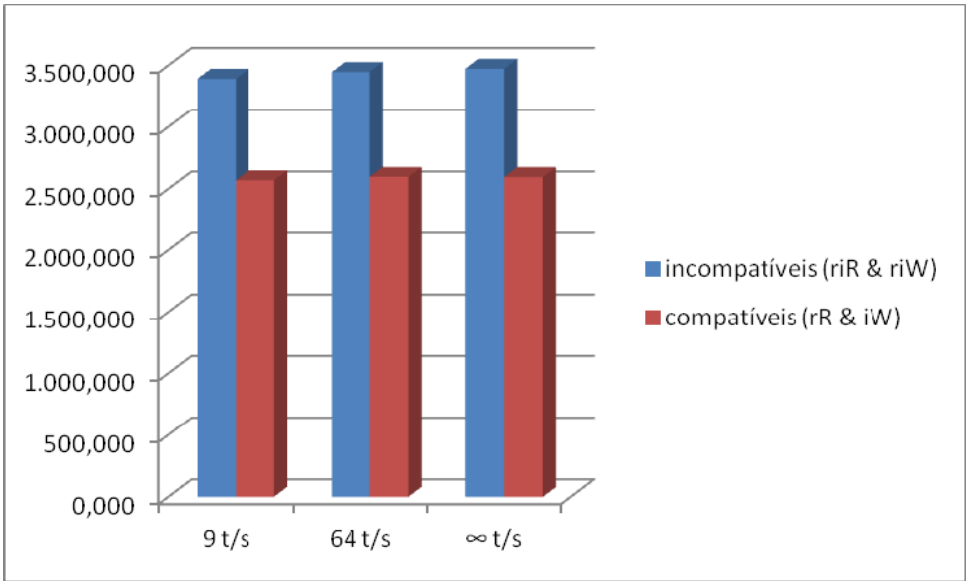

Figura 79 - Cenário 13 - número de abortos

***Turnaround* (seg):**

Tabela 57 - Cenário 13 - *turnaround* (seg)

Tipos de Bloqueio \ (tran/s)	9 t/s	64 t/s	∞ t/s
incompatíveis (riR & riW)	451,260	502,219	515,338
compatíveis (rR & iW)	464,911	513,973	523,861

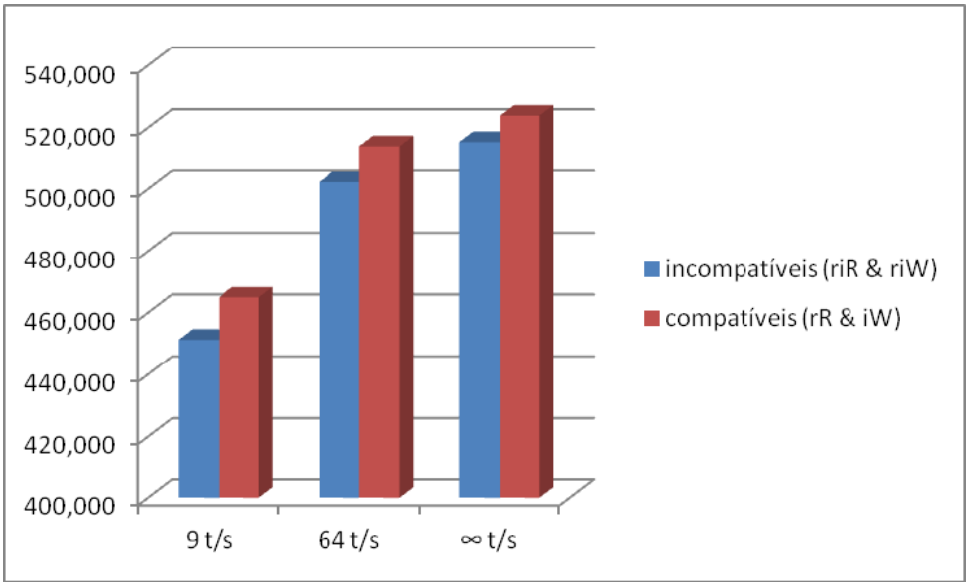

Figura 80 - Cenário 13 - *turnaround* (seg)

A.1.3.14.
Cenário 14: Comparação entre tipos de bloqueio, de leitura e escrita, novos (compatíveis) e convencionais (incompatíveis); Protocolo Monogranular (NO_MGL); Grânulo "Property of Resource"; Acesso a 10% pares do total de 30.000; Suite 2 (20% bloqueios de escrita)

Parâmetros de entradas específicos:

- número de recursos: 300
- número de propriedades: 100
- min_pairs_per_req: 3000
- max_pairs_per_req: 3000
- *min_restart_time*: 0
- *max_restart_time*: 0

A seguir os valores observados.

Número de abortos:

Tabela 58 - Cenário 14 - número de abortos

Tipos de Bloqueio \ (tran/s)	9 t/s	64 t/s	∞ t/s
incompatíveis (riR & riW)	1.793,379	1.817,480	1.788,752
compatíveis (rR & iW)	197,484	195,294	202,127

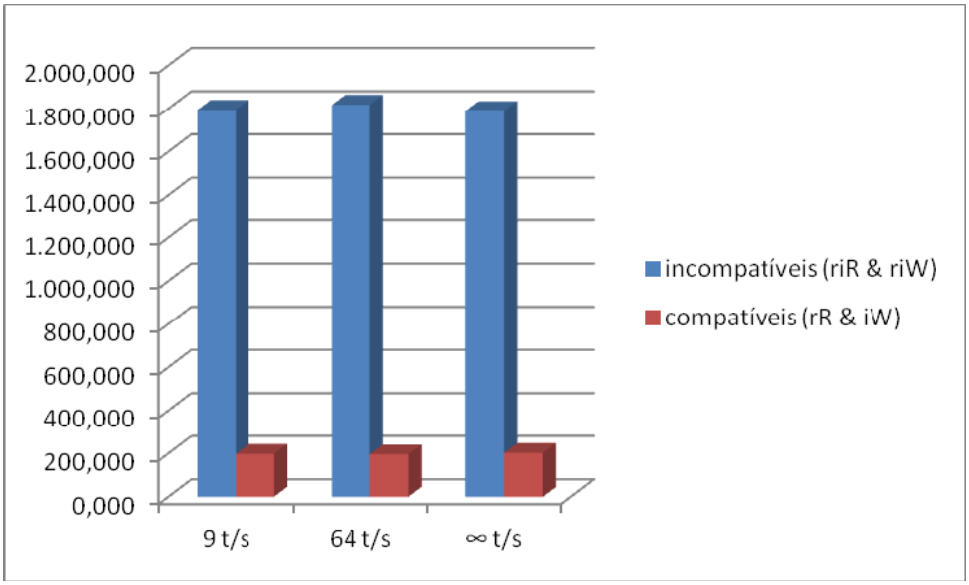

Figura 81 - Cenário 14 - número de abortos

***Turnaround* (seg):**

Tabela 59 - Cenário 14 - *turnaround* (seg)

Tipos de Bloqueio \ (tran/s)	9 t/s	64 t/s	∞ t/s
incompatíveis (riR & riW)	543,446	595,607	608,548
compatíveis (rR & iW)	542,684	598,770	609,738

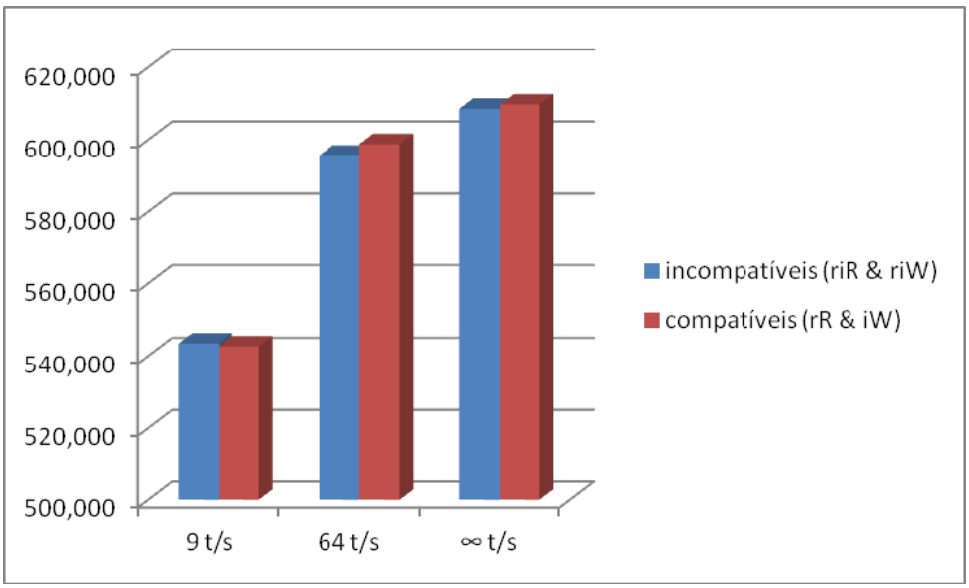

Figura 82 - Cenário 14 - *turnaround* (seg)

A.1.3.15.
Cenário 15 (extra): Grânulos em Separado (NO_MGL - monogranular); Acesso a 20% pares do total de 30.000; Suite 1 (80% bloqueios de escrita)

Parâmetros de entradas específicos:

- número de recursos: 300
- número de propriedades: 100
- min_pairs_per_req: 6000
- max_pairs_per_req: 6000
- *min_restart_time*: 0
- *max_restart_time*: 0

A seguir os valores observados.

Número de bloqueios:

Tabela 60 - Cenário 15 - número de bloqueios

grânulo	número de bloqueios
graph	1,000
property	100,000
resource	300,000
property of resource	6.000,000

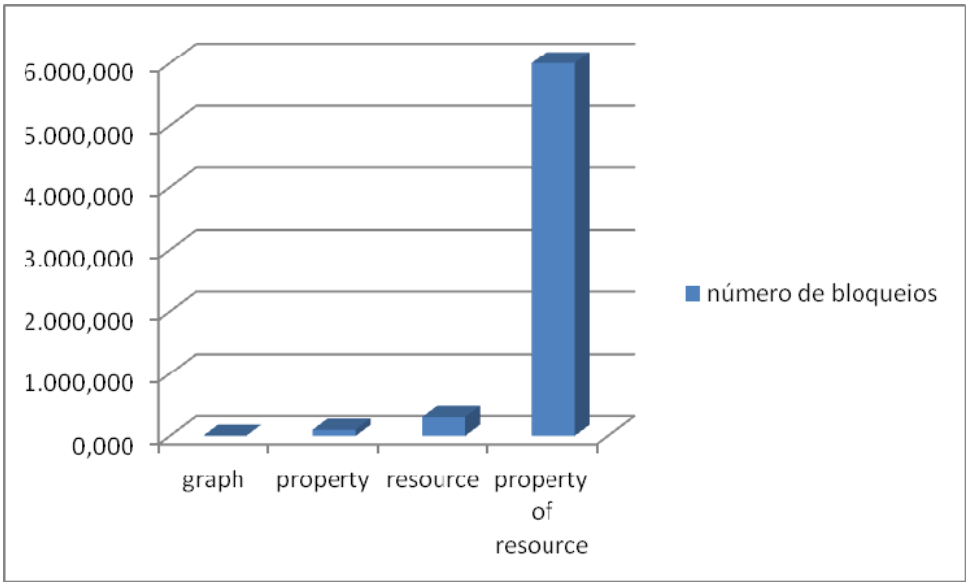

Figura 83 - Cenário 15 - número de bloqueios

Número de abortos:

Tabela 61 - Cenário 15 - número de abortos

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	78.618,275	79.440,182	80.812,965
property	76.904,695	77.494,613	78.485,516
resource	77.382,587	78.602,671	79.563,592
property of resource	17.576,869	17.798,101	17.947,627

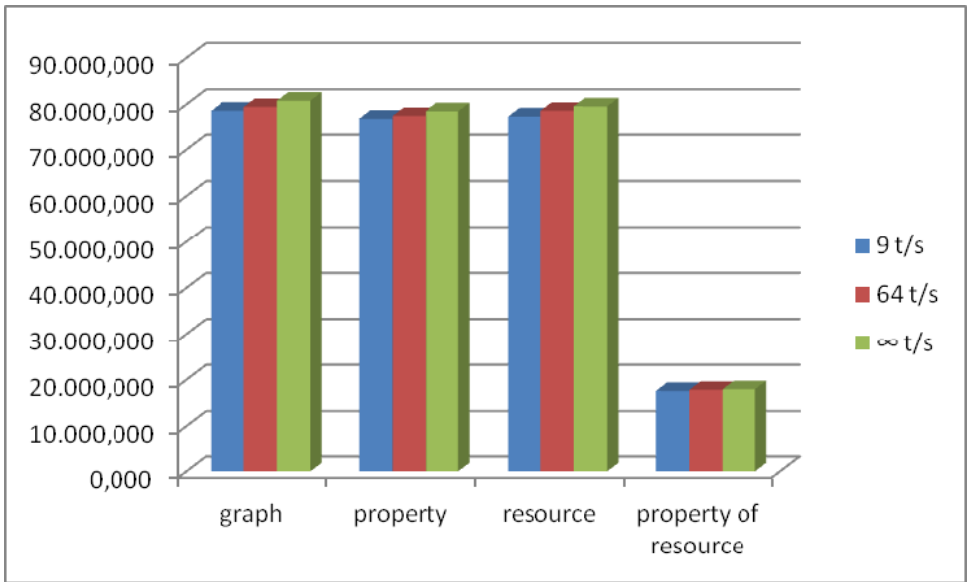

Figura 84 - Cenário 15 - número de abortos

***Turnaround* (seg):**

Tabela 62 - Cenário 15 - *turnaround* (seg)

grânulo \ (tran/s)	9 t/s	64 t/s	∞ t/s
graph	1.406,968	1.462,639	1.501,542
property	1.458,678	1.520,996	1.554,260
resource	1.519,511	1.591,760	1.626,311
property of resource	1.540,352	1.597,581	1.622,808

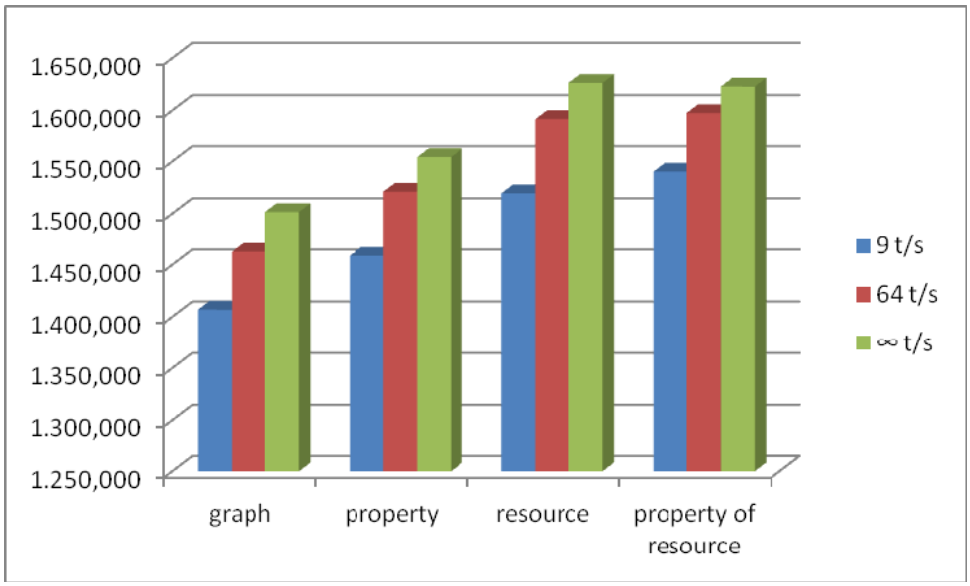

Figura 85 - Cenário 15 - *turnaround* (seg)

A.2.
Código Fonte do *Lock Manager*

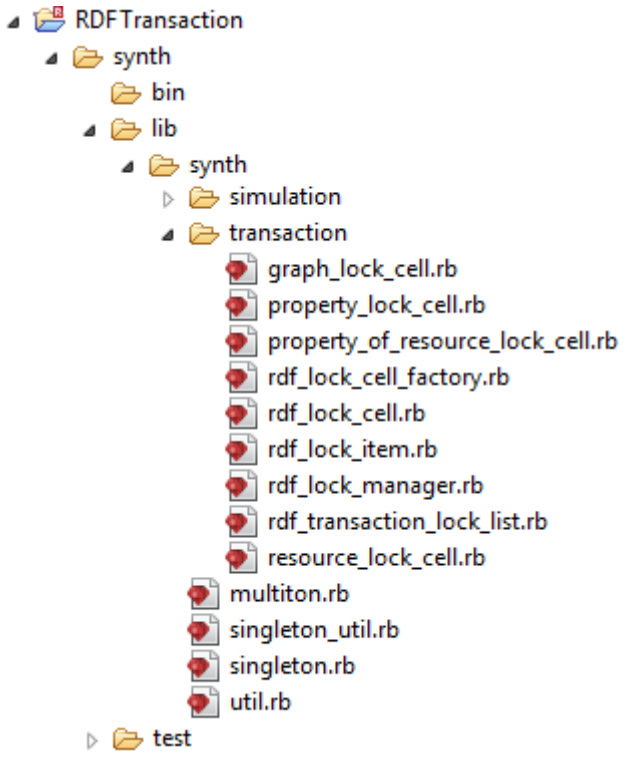

Figura 86 - Pacotes do *Lock Manager*

rdf_lock_manager.rb

```
require 'thread'
require 'synth/singleton'
require 'rdf_lock_cell_factory'
require 'rdf_transaction_lock_list'

module Synth
  module Transaction
    class RdfLockManager
      include Synth::Singleton

      #constantes dos granulos
      def self.graph();                :d_g; end
      def self.property();             :c_p; end
      def self.resource();             :b_r; end
      def self.property_of_resource(); :a_pr; end


      #instance methods apenas por conveniencia
      def graph();                self.class.graph; end
      def property();             self.class.property; end
      def resource();             self.class.resource; end
      def property_of_resource(); self.class.property_of_resource; end

      @granule_names = {graph => 'graph',
                        property => 'property',
                        resource => 'resource',
                        property_of_resource => 'property of resource'}
      @granule_names.freeze

      #constantes dos lock Modes (tipos de locks)
      #Lock primitivos
      def self.rR();       0; end
      def self.iR();       1; end
```

```
      def self.riR();      2; end
      def self.rW();       3; end
      def self.iW();       4; end
      def self.riW();      5; end
      def self.prR();      6; end
      def self.piR();      7; end
      def self.priR();     8; end
      def self.prW();      9; end
      def self.piW();     10; end
      def self.priW();    11; end
      #Locks compostos
      def self.rRpiR();   12; end
      def self.rRprW();   13; end
      def self.rRpiW();   14; end
      def self.rRpriW();  15; end
      def self.iRprR();   16; end
      def self.iRprW();   17; end
      def self.iRpiW();   18; end
      def self.iRpriW();  19; end
      def self.riRprW();  20; end
      def self.riRpiW();  21; end
      def self.riRpriW(); 22; end
      def self.rWpiW();   23; end
      def self.iWprW();   24; end

      #instance methods apenas por conveniencia
      #Lock primitivos
      def rR();       self.class.rR; end
      def iR();       self.class.iR; end
      def riR();      self.class.riR; end
      def rW();       self.class.rW; end
      def iW();       self.class.iW; end
      def riW();      self.class.riW; end
      def prR();      self.class.prR; end
      def piR();      self.class.piR; end
      def priR();     self.class.priR; end
      def prW();      self.class.prW; end
      def piW();      self.class.piW; end
      def priW();     self.class.priW; end
      #Locks compostos
      def rRpiR();    self.class.rRpiR; end
      def rRprW();    self.class.rRprW; end
      def rRpiW();    self.class.rRpiW; end
      def rRpriW();   self.class.rRpriW; end
      def iRprR();    self.class.iRprR; end
      def iRprW();    self.class.iRprW; end
      def iRpiW();    self.class.iRpiW; end
      def iRpriW();   self.class.iRpriW; end
      def riRprW();   self.class.riRprW; end
      def riRpiW();   self.class.riRpiW; end
      def riRpriW();  self.class.riRpriW; end
      def rWpiW();    self.class.rWpiW; end
      def iWprW();    self.class.iWprW; end

      @lock_mode_names = ['rR', 'iR', 'riR', 'rW', 'iW', 'riW', 'prR', 'piR', 'priR',
'prW', 'piW', 'priW', 'rRpiR', 'rRprW', 'rRpiW', 'rRpriW', 'iRprR', 'iRprW', 'iRpiW',
'iRpriW', 'riRprW', 'riRpiW', 'riRpriW', 'rWpiW', 'iWprW']
      @lock_mode_names.freeze

      @compatibility_matrix =
        [
          [true,  true,  true,  false, true,  false, true,  true,  true,  false,
true,  false, true,  false, true,  false, true,  false, true,  false, false,
true,  false, false, false],
          [true,  true,  true,  true,  false, false, true,  true,  true,  true,
false, false, true,  true,  false, false, true,  true,  false, false, true,
false, false, false, false],
          [true,  true,  true,  false, false, false, true,  true,  true,  false,
false, false, true,  false, false, false, true,  false, false, false, false,
false, false, false, false],
          [false, true,  false, false, false, false, false, true,  false, false,
false, false, false, false, false, false, false, false, false, false, false,
false, false, false, false],
          [true,  false, false, false, false, false, true,  false, false, false,
```

```
false,  false,  false,  false,  false,  false,  false,  false,  false,  false,  false,
false,  false,  false,  false],
          [false, false,  false,  false,  false,  false,  false,  false,  false,  false,  false,
false,  false,  false,  false,  false,  false,  false,  false,  false,  false,  false,
false,  false,  false,  false],
          [true,  true,   true,   false,  true,   false,  true,   true,   true,   true,
true,   true,   true,   true,   true,   true,   true,   true,   true,   true,   true,
true,   true,   false,  true ],
          [true,  true,   true,   true,   false,  false,  true,   true,   true,   true,
true,   true,   true,   true,   true,   true,   true,   true,   true,   true,   true,
true,   true,   true,   false],
          [true,  true,   true,   false,  false,  false,  true,   true,   true,   true,
true,   true,   true,   true,   true,   true,   true,   true,   true,   true,   true,
true,   true,   false,  false],
          [false, true,   false,  false,  false,  false,  true,   true,   true,   true,
true,   true,   false,  false,  false,  false,  true,   true,   true,   true,   false,
false,  false,  false,  false],
          [true,  false,  false,  false,  false,  false,  true,   true,   true,   true,
true,   true,   true,   true,   true,   true,   false,  false,  false,  false,  false,
false,  false,  false,  false],
          [false, false,  false,  false,  false,  false,  true,   true,   true,   true,
true,   true,   false,  false,  false,  false,  false,  false,  false,  false,  false,
false,  false,  false,  false],
          [true,  true,   true,   false,  false,  false,  true,   true,   true,   false,
true,   false,  true,   false,  true,   false,  true,   false,  true,   false,  false,
true,   false,  false,  false],
          [false, true,   false,  false,  false,  false,  true,   true,   true,   false,
true,   false,  false,  false,  false,  false,  true,   false,  true,   false,  false,
false,  false,  false,  false],
          [true,  false,  false,  false,  false,  false,  true,   true,   true,   false,
true,   false,  true,   false,  true,   false,  false,  false,  false,  false,  false,
false,  false,  false,  false],
          [false, false,  false,  false,  false,  false,  true,   true,   true,   false,
true,   false,  false,  false,  false,  false,  false,  false,  false,  false,  false,
false,  false,  false,  false],
          [true,  true,   true,   false,  false,  false,  true,   true,   true,   true,
false,  false,  true,   true,   false,  false,  true,   true,   false,  false,  true,
false,  false,  false,  false],
          [false, true,   false,  false,  false,  false,  true,   true,   true,   true,
false,  false,  false,  false,  false,  false,  true,   true,   false,  false,  false,
false,  false,  false,  false],
          [true,  false,  false,  false,  false,  false,  true,   true,   true,   true,
false,  false,  true,   true,   false,  false,  false,  false,  false,  false,  false,
false,  false,  false,  false],
          [false, false,  false,  false,  false,  false,  true,   true,   true,   true,
false,  false,  false,  false,  false,  false,  false,  false,  false,  false,  false,
false,  false,  false,  false],
          [false, true,   false,  false,  false,  false,  true,   true,   true,   false,
false,  false,  false,  false,  false,  false,  true,   false,  false,  false,  false,
false,  false,  false,  false],
          [true,  false,  false,  false,  false,  false,  true,   true,   true,   false,
false,  false,  true,   false,  false,  false,  false,  false,  false,  false,  false,
false,  false,  false,  false],
          [false, false,  false,  false,  false,  false,  true,   true,   true,   false,
false,  false,  false,  false,  false,  false,  false,  false,  false,  false,  false,
false,  false,  false,  false],
          [false, false,  false,  false,  false,  false,  false,  true,   false,  false,
false,  false,  false,  false,  false,  false,  false,  false,  false,  false,  false,
false,  false,  false,  false],
          [false, false,  false,  false,  false,  false,  true,   false,  false,  false,
false,  false,  false,  false,  false,  false,  false,  false,  false,  false,  false,
false,  false,  false,  false]
        ]
      @compatibility_matrix.freeze

      @conversion_matrix =
        [
          [rR,      riR,     riR,     rW,    iW,    riW, rR,      rRpiR,   rRpiR,   rRprW,
rRpiW,   rRpriW,  rRpiR,   rRprW,   rRpiW,   rRpriW,  riR,     riRprW,  riRpiW,  riRpriW,
riRprW,  riRpiW,  riRpriW, rWpiW, iWprW],
          [riR,     iR,      riR,     rW,    iW,    riW, iRprR,   iR,      iRprR,   iRprW,
iRpiW,   iRpriW,  riR,     riRprW,  riRpiW,  riRpriW, iRprR,   iRprW,   iRpiW,   iRpriW,
riRprW,  riRpiW,  riRpriW, rWpiW, iWprW],
          [riR,     riR,     riR,     rW,    iW,    riW, riR,     riR,     riR,
```

```
riRprW,  riRpiW,  riRpriW, riR,     riRprW,  riRpiW,  riRpriW, riR,     riRprW,  riRpiW,
riRpriW, riRprW,  riRpiW,  riRpriW, rWpiW, iWprW],
          [rW,      rW,      rW,      rW,    riW,   riW, rW,      rW,      rW,      rW,
rWpiW,   rWpiW,   rW,      rW,      rWpiW,   rWpiW,   rW,      rW,      rWpiW,   rWpiW,
rW,      rWpiW,   rWpiW,   rWpiW, riW  ],
          [iW,      iW,      iW,      riW,   iW,    riW, iW,      iW,      iW,      iWprW,
iW,      iWprW,   iW,      iWprW,   iW,      iWprW,   iW,      iWprW,   iW,      iWprW,
iWprW,   iW,      iWprW,   riW,   iWprW],
          [riW,     riW,     riW,     riW,   riW,   riW, riW,     riW,     riW,     riW,
riW,     riW,     riW,     riW,     riW,     riW,     riW,     riW,     riW,     riW,
riW,     riW,     riW,     riW,   riW  ],
          [rR,      iRprR,   riR,     rW,    iW,    riW, prR,     priR,    priR,    prW,
piW,     priW,    rRpiR,   rRprW,   rRpiW,   rRpriW,  iRprR,   iRprW,   iRpiW,   iRpriW,
riRprW,  riRpiW,  riRpriW, rWpiW, iWprW],
          [rRpiR,   iR,      riR,     rW,    iW,    riW, priR,    piR,     priR,    prW,
piW,     priW,    rRpiR,   rRprW,   rRpiW,   rRpriW,  iRprR,   iRprW,   iRpiW,   iRpriW,
riRprW,  riRpiW,  riRpriW, rWpiW, iWprW],
          [rRpiR,   iRprR,   riR,     rW,    iW,    riW, priR,    priR,    priR,    prW,
piW,     priW,    rRpiR,   rRprW,   rRpiW,   rRpriW,  iRprR,   iRprW,   iRpiW,   iRpriW,
riRprW,  riRpiW,  riRpriW, rWpiW, iWprW],
          [rRprW,   iRprW,   riRprW,  rW,    iWprW, riW, prW,     prW,     prW,     prW,
priW,    priW,    rRprW,   rRprW,   rRpriW,  rRpriW,  iRprW,   iRprW,   iRpriW,  iRpriW,
riRprW,  riRpriW, riRpriW, rWpiW, iWprW],
          [rRpiW,   iRpiW,   riRpiW,  rWpiW, iW,    riW, piW,     piW,     piW,     priW,
piW,     priW,    rRpiW,   rRpriW,  rRpiW,   rRpriW,  iRpiW,   iRpriW,  iRpiW,   iRpriW,
riRpriW, riRpiW,  riRpriW, rWpiW, iWprW],
          [rRpriW,  iRpriW,  riRpriW, rWpiW, iWprW, riW, priW,    priW,    priW,    priW,
priW,    priW,    rRpriW,  rRpriW,  rRpriW,  rRpriW,  iRpriW,  iRpriW,  iRpriW,  iRpriW,
riRpriW, riRpriW, riRpriW, rWpiW, iWprW],
          [rRpiR,   riR,     riR,     rW,    iW,    riW, rRpiR,   rRpiR,   rRpiR,   rRprW,
rRpiW,   rRpriW,  rRpiR,   rRprW,   rRpiW,   rRpriW,  riR,     riRprW,  riRpiW,  riRpriW,
riRprW,  riRpiW,  riRpriW, rWpiW, iWprW],
          [rRprW,   riRprW,  riRprW,  rW,    iWprW, riW, rRprW,   rRprW,   rRprW,   rRprW,
rRpriW,  rRpriW,  rRprW,   rRprW,   rRpriW,  rRpriW,  iRprW,  iRprW,  iRpriW, iRpriW,
riRprW,  riRpriW, riRpriW, rWpiW, iWprW],
          [rRpiW,   riRpiW,  riRpiW,  rWpiW, iW,    riW, rRpiW,   rRpiW,   rRpiW,
rRpriW,  rRpiW,   rRpriW,  rRpiW,   rRpriW,  rRpiW,   rRpriW,  riRpiW,  riRpriW, riRpiW,
riRpriW, riRpriW, riRpiW,  riRpriW, rWpiW, iWprW],
          [rRpriW,  riRpriW, riRpriW, rWpiW, iWprW, riW, rRpriW,  rRpriW,  rRpriW,
rRpriW,  rRpriW,  rRpriW,  rRpriW,  rRpriW,  rRpriW,  rRpriW,  riRpriW, riRpriW, riRpriW,
riRpriW, riRpriW, riRpriW, riRpriW, rWpiW, iWprW],
          [riR,     iRprR,   riR,     rW,    iW,    riW, iRprR,   iRprR,   iRprR,   iRprW,
iRpiW,   iRpriW,  riR,     riRprW,  riRpiW,  riRpriW, iRprR,   iRprW,   iRpiW,   iRpriW,
riRprW,  riRpiW,  riRpriW, rWpiW, iWprW],
          [riRprW,  iRprW,   riRprW,  rW,    iWprW, riW, iRprW,   iRprW,   iRprW,   iRprW,
iRpriW,  iRpriW,  riRprW,  riRprW,  riRpriW, riRpriW, iRprW,   iRprW,   iRpriW,  iRpriW,
riRprW,  riRpriW, riRpriW, rWpiW, iWprW],
          [riRpiW,  iRpiW,   riRpiW,  rWpiW, iW,    riW, iRpiW,   iRpiW,   iRpiW,
iRpriW,  iRpiW,   iRpriW,  riRpiW,  riRpriW, riRpiW,  riRpriW, iRpiW,   iRpriW,  iRpiW,
iRpriW,  riRpriW, riRpiW,  riRpriW, rWpiW, iWprW],
          [riRpriW, iRpriW,  riRpriW, rWpiW, iWprW, riW, iRpriW,  iRpriW,  iRpriW,
iRpriW,  iRpriW,  iRpriW,  riRpriW, riRpriW, riRpriW, riRpriW, iRpriW,  iRpriW,  iRpriW,
iRpriW,  riRpriW, riRpriW, riRpriW, rWpiW, iWprW],
          [riRprW,  riRprW,  riRprW,  rW,    iWprW, riW, riRprW,  riRprW,  riRprW,
riRprW,  riRpriW, riRpriW, riRprW,  riRprW,  riRpriW, riRpriW, riRprW,  riRprW,  riRpriW,
riRpriW, riRprW,  riRpriW, riRpriW, rWpiW, iWprW],
          [riRpiW,  riRpiW,  riRpiW,  rWpiW, iW,    riW, riRpiW,  riRpiW,  riRpiW,
riRpriW, riRpiW,  riRpriW, riRpiW,  riRpriW, riRpiW,  riRpriW, riRpiW,  riRpriW, riRpiW,
riRpriW, riRpriW, riRpiW,  riRpriW, rWpiW, iWprW],
          [riRpriW, riRpriW, riRpriW, rWpiW, iWprW, riW, riRpriW, riRpriW, riRpriW,
riRpriW, riRpriW, riRpriW, riRpriW, riRpriW, riRpriW, riRpriW, riRpriW, riRpriW, riRpriW,
riRpriW, riRpriW, riRpriW, riRpriW, rWpiW, iWprW],
          [rWpiW,   rWpiW,   rWpiW,   rWpiW, riW,   riW, rWpiW,   rWpiW,   rWpiW,   rWpiW,
rWpiW,   rWpiW,   rWpiW,   rWpiW,   rWpiW,   rWpiW,   rWpiW,   rWpiW,   rWpiW,   rWpiW,
rWpiW,   rWpiW,   rWpiW,   rWpiW, riW  ],
          [iWprW,   iWprW,   iWprW,   riW,   iWprW, riW, iWprW,   iWprW,   iWprW,   iWprW,
iWprW,   iWprW,   iWprW,   iWprW,   iWprW,   iWprW,   iWprW,   iWprW,   iWprW,   iWprW,
iWprW,   iWprW,   iWprW,   riW,   iWprW]
        ]
      @conversion_matrix.freeze

      @conversion_from_real_to_planned_vector = [prR, piR, priR, prW, piW, priW, prR, piR,
priR, prW, piW, priW, priR, prW, piW, priW, priR, prW, piW, priW, prW, piW, priW, priW,
priW]
```

```
      @conversion_from_real_to_planned_vector.freeze

      def initialize(mgl=true)
        @mgl = mgl

        #hash mapeia o data-item no seu lock-cell (lista de lock-items)
        @lock_table = {}
        @lock_table_mutex = Mutex.new

        #hash que mapeia o transaction-id na sua lista de lock-items
        @transaction_table = {}
        @transaction_table_mutex = Mutex.new
      end

      def mgl?
        @mgl
      end

      #metodo friend, usada pelo RDFLockCell para se adicionar ao ser criado
      def add_lock_cell(lock_cell)
        @lock_table[lock_cell.data_item] = lock_cell
      end
      private :add_lock_cell

      #metodo friend, usada pelo RDFLockCell para se remover
      def remove_lock_cell(lock_cell)
        @lock_table_mutex.synchronize do
           @lock_table.delete lock_cell.data_item
        end
      end
      private :remove_lock_cell

      def mount_data_item(granule, uris)
        result = granule.to_s
        case granule
        when graph
          result
        when property
          result += uris[:property].to_s
        when resource
          result += uris[:resource].to_s
        when property_of_resource
          result += property.to_s + uris[:property].to_s
          result += resource.to_s + uris[:resource].to_s
        end
        result
      end
      private :mount_data_item

      def lock_cell(granule, uris)
        @lock_table_mutex.synchronize do
          data_item = mount_data_item granule, uris
          lock_cell = @lock_table[data_item]
          unless lock_cell
            #Ao criar o RDFLockCell, este se adiciona na @lock_table
            lock_cell = Synth::Transaction::RdfLockCellFactory.
              instance.create_lock_cell self, granule, data_item
          end
          lock_cell
        end
      end
      private :lock_cell

      #metodo friend usado pelo RDFTransactionLockList para se adicionar
      #ao ser criado
      def add_transaction_lock_list(lock_list)
        @transaction_table[lock_list.transaction_id] = lock_list
      end
      private :add_transaction_lock_list

      #metodo friend usado pelo RDFTransactionLockList para se remover
      def remove_transaction_lock_list(lock_list)
        @transaction_table_mutex.synchronize do
           @transaction_table.delete lock_list.transaction_id
```

```
        end
      end
      private :remove_transaction_lock_list

      def transaction_lock_list(transaction_id)
        @transaction_table_mutex.synchronize do
          lock_list = @transaction_table[transaction_id]
          unless lock_list
            #Ao criar o RDFTransactionLockList, este se adiciona na
            #@transaction_table
            lock_list =
              Synth::Transaction::RdfTransactionLockList.new self,transaction_id
          end
          lock_list
        end
      end
      private :transaction_lock_list

      def granule_name(granule)
         self.class.instance_eval do
            @granule_names[granule]
         end
      end

      def lock_mode_name(lock_mode)
         self.class.instance_eval do
            @lock_mode_names[lock_mode]
         end
      end

      def write_lock_mode?(lock_mode)
        lock_mode_name(lock_mode).include? 'W'
      end

      def compatible?(current_lock_mode, requested_lock_mode)
         self.class.instance_eval do
            @compatibility_matrix[current_lock_mode][requested_lock_mode]
         end
      end

      def conversion(old_lock_mode, requested_lock_mode)
         self.class.instance_eval do
            @conversion_matrix[old_lock_mode][requested_lock_mode]
         end
      end

      def planned_lock_mode(lock_mode)
         self.class.instance_eval do
            @conversion_from_real_to_planned_vector[lock_mode]
         end
      end

      def lock(transaction_id, granule, lock_mode, uris={})
        result = nil
        result = lock_cell(granule, uris).lock(transaction_id, lock_mode, uris) while
result.nil?

        return result unless result

        #locar a propriedade inversa, caso haja
        if (granule == property) || (granule == property_of_resource)
          inv_property = uris[:inv_property]
          if inv_property
            inv_uris = {:property => inv_property}
            result = lock(transaction_id, property, lock_mode, inv_uris)
            unless result
              unlock(transaction_id, granule, uris)
            end
          end
        end
        result
      end

      def unlock(transaction_id, granule, uris={})
```

```
        result = lock_cell(granule, uris).unlock(transaction_id)

        #unlocar a propriedade inversa, caso haja
        if (granule == property) || (granule == property_of_resource)
          inv_property = uris[:inv_property]
          if inv_property
            inv_uris = {:property => inv_property}
            result = unlock(transaction_id, property, inv_uris)
          end
        end
        result
      end

      def unlock_all(transaction_id)
          transaction_lock_list(transaction_id).unlock_all
      end

      #operação que deve ser chamada, periodicamente, quando o sistema
      #estiver ocioso, para remover alguns lock_cells para que a lock_table
      #não fique muito cheia.
      #Esta operação somente precisa ser chamada, caso a lock_cell não esteja
      #se auto-removendo ou seja, caso o conteúdo da operação
      #check_removal da module RdfLockCell esteja comentado.
      #Por default, esta operação deve ser usada, ou seja, o conteúdo de
      #check_removal da module RdfLockCell, está comentado.
      def remove_empty_lock_cells
        lock_cells =
          @lock_table_mutex.synchronize do
             @lock_table.values
          end
        lock_cells.each {|lock_cell| lock_cell.remove}
      end

      def to_s

        result = "\nLock Table:\n"
        @lock_table_mutex.synchronize do
          @lock_table.values.sort.each do |v|
            result += "\n" + v.to_s
          end
        end


        result += "\nTransaction Table:\n"
        @transaction_table_mutex.synchronize do
          @transaction_table.values.sort.each do |v|
            result += "\n" + v.to_s
          end
        end
        result
      end

    end
  end
end
```

rdf_lock_cell.rb

```
require 'thread'
require 'rdf_lock_item'

module Synth
  module Transaction
    module RdfLockCell

      attr_reader :data_item, :lock_manager

      def initialize(lock_manager, data_item)
        @mutex = Mutex.new
        @data_item  = data_item
        @lock_item  = nil #Pattern Chain of Responsibility
        @lock_manager = lock_manager
        @lock_manager.send :add_lock_cell, self
        @removed = false
      end
```

```
def synchronized(&block)
    @mutex.synchronize &block if block
end
private :synchronized


def lock_mutex()
  @mutex.lock
end
private :lock_mutex

#metodo friend usado pelo RDFLockItem no seu unlock
def unlock_mutex()
  @mutex.unlock
end
private :unlock_mutex

#metodo friend usado pelo RDFLockItem
def set_lock_item(lock_item)
  @lock_item = lock_item
end
private :set_lock_item

def lock_item(transaction_id)
  #Ao criar o RDFLockItem, este invoca a operacao set_lock_item
  unless @lock_item
    Synth::Transaction::RdfLockItem.new(transaction_id, self)
  end
  @lock_item.lock_item transaction_id
end
private :lock_item

def conflict?(transaction_id, lock_mode)
  return false unless @lock_item
  @lock_item.conflict?(transaction_id, lock_mode)
end
private :conflict?

def lock_cell(granule, uris)
  @lock_manager.send :lock_cell, granule, uris
end
private :lock_cell

def propagate_read_lock(transaction_id, lock_mode, child, uris)
  raise NotImplementedError,
  "#{self.class.name}#propagate_read_lock() abstract method is not implemented."
end
private :propagate_read_lock

def propagate_write_lock(transaction_id, lock_mode, child, uris)
  raise NotImplementedError,
  "#{self.class.name}#propagate_write_lock() abstract method is not implemented."
end
private :propagate_write_lock

def propagate_lock(transaction_id, lock_mode, child, uris)
  synchronized do
    return nil if @removed
    return false if conflict?(transaction_id, lock_mode)

    lock_item = lock_item(transaction_id)

    result = if @lock_manager.write_lock_mode? lock_mode
               propagate_write_lock(transaction_id,
                                    lock_mode,
                                    lock_item,
                                    uris)
             else
               propagate_read_lock(transaction_id,
                                   lock_mode,
                                   lock_item,
                                   uris)
             end
```

```
        if result #se não teve conflito na propagacao
          lock_item.propagate_lock lock_mode, child
        else
          #remover o lock_item, se recem criado
          lock_item.unlock(self) unless lock_item.locked?

          check_removal()
        end

        return result
      end
    end

    def lock(transaction_id, lock_mode, uris)
      synchronized do
        return nil if @removed
        return false if conflict?(transaction_id, lock_mode)

        lock_item = lock_item(transaction_id)

        result = true
        if @lock_manager.mgl?
          planned_lock_mode = @lock_manager.planned_lock_mode(lock_mode)
          result = if @lock_manager.write_lock_mode? lock_mode
                     propagate_write_lock(transaction_id,
                                          planned_lock_mode,
                                          lock_item,
                                          uris)
                   else
                     propagate_read_lock(transaction_id,
                                         planned_lock_mode,
                                         lock_item,
                                         uris)
                   end
        end

        if result #se não teve conflito na propagacao
          lock_item.lock lock_mode

          #File.open("log.txt", "a") do |log|
          #   log.puts(transaction_id.to_s + " - " + self.to_s + "\n")
          #end
        else
          #remover o lock_item, se recem criado
          lock_item.unlock(self) unless lock_item.locked?

          check_removal()
        end
        return result
      end

    end

    def propagate_unlock(transaction_id)
      synchronized do
        lock_item = lock_item(transaction_id)
        lock_item.propagate_unlock(self)
      end
    end

    def unlock(transaction_id)
      synchronized do
        lock_item = lock_item(transaction_id)
        lock_item.unlock(self)
      end
    end

    def handle_remove
      return if @removed

      unless @lock_item && @lock_item.locked?
         @lock_manager.send(:remove_lock_cell, self)
         @removed = true
```

```
        end
      end
      private :handle_remove

      #metodo friend usado pelo RDFLockItem
      def check_removal()
        #retirar o comentário da chamada da operação "handle_remove" abaixo
        #se quiser que este lock_cell se auto-remova do Lock Manager
        #Caso a linha abaixo permaneça comentada, a aplicação deve
        #fazer uso da operação remove_empty_lock_cells da classe RDFLockManager
        #Isto é uma escolha que o projetista da aplicação deve fazer.

        #handle_remove
      end
      private :check_removal

      #chamado pela operação remove_empty_lock_cells da classe RDFLockManager
      def remove
        synchronized do
          handle_remove
        end
      end

      def to_s
        "#{self.object_id} - #{@data_item} \t #{@lock_item ? @lock_item.to_s : 'no
locks'}"
      end

      def <=>(other)
        @data_item <=> other.data_item
      end

    end
  end
end
```

rdf_lock_item.rb

```
module Synth
  module Transaction
    class RdfLockItem

      attr_reader :transaction_id, :lock_mode

      #Pattern Chain of Responsibility
      def successor=(successor)
        @successor = successor
      end
      protected :successor=

      def predecessor=(predecessor)
        @predecessor = predecessor
        @lock_cell.send(:set_lock_item, self) unless @predecessor
      end
      protected :predecessor=

      def add_parent?(parent)
        return false if @parents.include? parent
        @parents << parent
        return true
      end
      protected :add_parent?

      #metodo friend, usada pelo RdfTransactionLockList
      def transaction_list_successor=(successor)
        @transaction_list_successor = successor
      end
      protected :transaction_list_successor=

      #metodo friend, usada pelo RdfTransactionLockList
      def transaction_list_predecessor=(predecessor)
        @transaction_list_predecessor = predecessor
        unless @transaction_list_predecessor
          @transaction_list.send(:set_lock_item, self)
        end
```

```
      end
      protected :transaction_list_predecessor=

      #metodo friend, usado pelo RdfTransactionLockList
      def remove_from_transaction_list()
        #faz o successor do meu predecessor = meu successor
        unless @transaction_list_predecessor
          @transaction_list.send(:set_lock_item, nil) #sai da Transaction List
        else
          @transaction_list_predecessor.transaction_list_successor =
            @transaction_list_successor
        end

        #se tiver successor,
        #faz o predecessor do meu successor  = meu predecessor
        @transaction_list_successor.transaction_list_predecessor =
          @transaction_list_predecessor if @transaction_list_successor
      end

      def initialize(transaction_id, lock_cell, predecessor=nil)
        @transaction_id = transaction_id
        @lock_cell = lock_cell
        @transaction_list =
          lock_cell.lock_manager.send :transaction_lock_list, transaction_id
        @transaction_list_successor = nil
        @transaction_list_predecessor = nil
        @is_in_transaction_list = false

        @lock_mode = nil
        @propagation = true
        @children_count = 0
        self.successor = nil
        self.predecessor = predecessor
        @parents = []
      end

      def lock_manager()
        @lock_cell.lock_manager
      end
      private :lock_manager

      def locked?()
        @lock_mode
      end

      def conflict?(transaction_id, lock_mode)
        result = false

        if locked? && (@transaction_id != transaction_id)
          result = !lock_manager.compatible?(@lock_mode, lock_mode)
        end

        if @successor && !result
          result = @successor.conflict?(transaction_id, lock_mode)
        end

        result
      end

      def lock_mode=(lock_mode)
        @lock_mode =   locked? ?
          lock_manager.conversion(@lock_mode, lock_mode) : lock_mode
      end
      private :lock_mode=

      def lock_item(transaction_id)
        if @transaction_id == transaction_id
          self
        else
          self.successor =
            self.class.new(transaction_id, @lock_cell, self) unless @successor
          @successor.lock_item transaction_id
        end
      end
```

```
      def propagate_lock(lock_mode, child)
        self.lock_mode = lock_mode
        @children_count += 1 if child.add_parent?(self)
      end

      def lock(lock_mode)
        self.lock_mode = lock_mode
        @propagation = false
        unless @is_in_transaction_list
          @transaction_list.add_lock_item(self)
          @is_in_transaction_list = true
        end
      end

      def remove_from_chain()
        #faz o successor do meu predecessor = meu successor
        unless @predecessor
          @lock_cell.send(:set_lock_item, nil) #sai do lock Cell
        else
          @predecessor.successor = @successor
        end

        #se tiver successor,
        #faz o predecessor do meu successor  = meu predecessor
        @successor.predecessor = @predecessor if @successor
      end
      private :remove_from_chain

      def propagate_unlock_to_parents()
        @parents.each do |parent|
            parent.propagate_unlock self#transaction_id
        end
      end
      private :propagate_unlock_to_parents

      def remove_itself()
        propagate_unlock_to_parents
        remove_from_chain
        @lock_cell.send :check_removal
      end
      private :remove_itself

      def propagate_unlock(caller)
        @lock_cell.send :lock_mutex unless (caller == @lock_cell)

        @children_count -= 1
        puts "#{@lock_cell.object_id} Numero de filhos negativo #{caller.class}" if
@children_count < 0

        if (@propagation && @children_count == 0)
          remove_itself
        end
        @lock_cell.send :unlock_mutex unless (caller == @lock_cell)

        true
      end

      def unlock(caller)

        @lock_cell.send :lock_mutex unless (caller == @lock_cell)
        if @children_count > 0
          @lock_mode = lock_manager.planned_lock_mode @lock_mode
          @propagation = true
        else
          remove_itself
        end
        @lock_cell.send :unlock_mutex unless (caller == @lock_cell)


        @transaction_list.remove_lock_item(self, caller) if @is_in_transaction_list

        true
      end
```

```
      def to_s
        result = "Lock cell: #{@lock_cell.object_id} "
        result += "[#{@transaction_id}, #{locked? ?
lock_manager.lock_mode_name(@lock_mode) : 'unlocked'}, #{@children_count},
#{@parents.size}]"
        result += ", #{@successor.to_s}" if @successor
        result
      end

      def transaction_list_to_s
        result = "Lock cell: #{@lock_cell.object_id} "
        result += "[#{@transaction_id}, #{locked? ?
lock_manager.lock_mode_name(@lock_mode) : 'unlocked'}, #{@children_count},
#{@parents.size}]"
        result += ", #{@transaction_list_successor.transaction_list_to_s}" if
@transaction_list_successor
        result
      end

    end
  end
end
```

graph_lock_cell.rb

```
require 'rdf_lock_cell'

module Synth
  module Transaction
    class GraphLockCell
      include Synth::Transaction::RdfLockCell

      def propagate_read_lock(transaction_id, lock_mode, child, uris)
        true
      end

      def propagate_write_lock(transaction_id, lock_mode, child, uris)
        true
      end

    end
  end
end
```

property_lock_cell.rb

```
require 'rdf_lock_cell'

module Synth
  module Transaction
    class PropertyLockCell
      include Synth::Transaction::RdfLockCell

      def propagate_read_lock(transaction_id, lock_mode, child, uris)
        #nao precisa gerar novas uris
        lock_cell = lock_cell lock_manager.graph, uris
        lock_cell.propagate_lock transaction_id, lock_mode, child, uris
      end

      def propagate_write_lock(transaction_id, lock_mode, child, uris)
        #nao precisa gerar novas uris
        lock_cell = lock_cell lock_manager.graph, uris
        lock_cell.propagate_lock transaction_id, lock_mode, child, uris
      end

    end
  end
end
```

resource_lock_cell.rb

```
module Synth
  module Transaction
    class ResourceLockCell
      include Synth::Transaction::RdfLockCell
```

```
      def propagate_read_lock(transaction_id, lock_mode, child, uris)
        #nao precisa gerar novas uris
        lock_cell = lock_cell lock_manager.graph, uris
        lock_cell.propagate_lock transaction_id, lock_mode, child, uris
      end

      def propagate_write_lock(transaction_id, lock_mode, child, uris)
        #nao precisa gerar novas uris
        lock_cell = lock_cell lock_manager.graph, uris
        lock_cell.propagate_lock transaction_id, lock_mode, child, uris
      end

    end
  end
end
```

property_of_resource_lock_cell.rb

```
require 'rdf_lock_cell'

module Synth
  module Transaction
    class PropertyOfResourceLockCell
      include Synth::Transaction::RdfLockCell

      def propagate_read_lock(transaction_id, lock_mode, child, uris)
        #nao precisa gerar novas uris
        lock_cell = lock_cell lock_manager.resource, uris
        lock_cell.propagate_lock transaction_id, lock_mode, child, uris
      end

      def propagate_write_lock(transaction_id, lock_mode, child, uris)
        #nao precisa gerar novas uris
        resource_lock_cell = lock_cell lock_manager.resource, uris
        result = resource_lock_cell.propagate_lock transaction_id,
                                                   lock_mode,
                                                   child,
                                                   uris

        if result
          #nao precisa gerar novas uris
          property_lock_cell = lock_cell lock_manager.property, uris
          result = property_lock_cell.propagate_lock transaction_id,
                                                      lock_mode,
                                                      child,
                                                      uris
        end

        result
      end

    end
  end
end
```

rdf_lock_cell_factory.rb

```
require 'synth/singleton'
require 'graph_lock_cell'
require 'property_lock_cell'
require 'resource_lock_cell'
require 'property_of_resource_lock_cell'



module Synth
  module Transaction
    class RdfLockCellFactory
      include Synth::Singleton

      def create_lock_cell(lock_manager, granule, data_item)
        case granule
        when lock_manager.graph
          Synth::Transaction::GraphLockCell.new lock_manager, data_item
        when lock_manager.property
          Synth::Transaction::PropertyLockCell.new lock_manager, data_item
```

```
        when lock_manager.resource
          Synth::Transaction::ResourceLockCell.new lock_manager, data_item
        when lock_manager.property_of_resource
          Synth::Transaction::PropertyOfResourceLockCell.new lock_manager,
                                                             data_item
        end

      end

    end
  end
end
```

rdf_transaction_lock_list.rb

```
require 'thread'

module Synth
  module Transaction
    class RdfTransactionLockList

      attr_reader :transaction_id

      def initialize(lock_manager, transaction_id)
        @mutex = Mutex.new
        @transaction_id = transaction_id
        @lock_item = nil
        @lock_manager = lock_manager
        @lock_manager.send :add_transaction_lock_list, self
      end

      def synchronized(&block)
          @mutex.synchronize &block if block
      end
      private :synchronized

      #metodo friend, usada pelo RDFLockItem
      def set_lock_item(lock_item)
        @lock_item = lock_item
      end
      private :set_lock_item

      def add_lock_item(lock_item)
        synchronized do
          temp = @lock_item
          @lock_item = lock_item
          if temp
            @lock_item.send :transaction_list_successor=, temp
            temp.send :transaction_list_predecessor=, @lock_item
          end
        end
      end

      def remove_itself()
        @lock_manager.send(:remove_transaction_lock_list,
                           self) unless @lock_item
      end
      private :remove_itself

      def handle_remove_lock_item(lock_item)
        lock_item.send :remove_from_transaction_list
        remove_itself
      end
      private :handle_remove_lock_item

      def remove_lock_item(lock_item, caller)
        if caller == self
          handle_remove_lock_item lock_item
        else
          synchronized {handle_remove_lock_item(lock_item)}
        end
      end

      def unlock_all()
        synchronized do
```

```
          @lock_item.unlock(self) while @lock_item
          remove_itself #pois a lista pode estar vazia.
          true
        end
      end

      def to_s
        "#{@transaction_id} \t #{@lock_item ? @lock_item.transaction_list_to_s : 'no
locks'}"
      end

      def <=>(other)
        @transaction_id <=> other.transaction_id
      end

    end
  end
end
```

singleton_util.rb

```
require 'thread'

module Synth
  module SingletonUtil
    def self.prepare_class(c)
      raise TypeError, "Class expected" unless c.is_a? Class

      c.class_eval do
        private_class_method :new, :allocate
        undef_method :dup, :clone

        @singleton_mutex = Mutex.new
        def self.synchronized(&block)
          @singleton_mutex.synchronize &block if block
        end

        def synchronized(&block)
          self.class.synchronized &block
        end

        def self.create_instance(*args)
          private_method_defined?(:initialize) && instance_method(:initialize).arity != 0
? new(*args) : new
        end

      end
    end
    private_class_method :prepare_class

    def self.prepare_singleton_class(c)
      prepare_class(c)
      c.class_eval do
        def self.instance(*args)
            unless instance_variable_defined? :@sole_instance
              synchronized do
                unless instance_variable_defined? :@sole_instance
                  @sole_instance = create_instance(*args)
                end
              end
            end
            @sole_instance
        end

        def _dump(limit)
          "dummy"
        end

        def self._load(s)
          instance
        end
      end
    end

    def self.prepare_multiton_class(c)
```

```
      prepare_class(c)
      c.class_eval do

        def self.instance(key, *args)
          if !instance_variable_defined?(:@instances) ||
             !@instances.has_key?(key)
             synchronized do
                @instances ||= {}
                unless @instances.has_key? key
                  @instances[key] = create_instance(*args)
                end
             end
          end
          @instances[key]
        end

        def _dump(limit)
          this = self
          synchronized do
            self.class.instance_eval do
              @instances.index(this).to_s
            end
          end
        end

        def self.string_to_key(s)
          raise NotImplementedError,
          "#{self.class.name}#srint_to_key() abstract method is not \
           implemented."
        end

        def self._load(s)
          instance(s)
        end

      end
    end

  end
end
```

singleton.rb

```
require 'singleton_util'

module Synth
  module Singleton
    def self.included(c)
      Synth::SingletonUtil.prepare_singleton_class(c)
    end
  end
end
```

A.3.
Vocabulário para Aquisição de Bloqueios

locking.owl

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<rdf:RDF xmlns="http://www.tecweb.inf.puc-rio.br/synth/locking.owl#"
     xml:base="http://www.tecweb.inf.puc-rio.br/synth/locking.owl"
```

```
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about="http://www.tecweb.inf.puc-rio.br/synth/locking.owl">
        <owl:imports rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
    </owl:Ontology>



    <!--
    ////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ////////////////////////////////////////////////////////////////////////
     -->



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#iRLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#iRLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#iRpiWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#iRpiWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#iRprRLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#iRprRLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#iRprWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#iRprWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#iRpriWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#iRpriWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#iWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#iWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>
```

```
    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#iWprWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#iWprWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#piRLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#piRLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#piWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#piWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#prRLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#prRLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#prWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#prWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#priRLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#priRLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#priWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#priWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#rRLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#rRLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>
```

```
    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#rRpiRLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#rRpiRLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#rRpiWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#rRpiWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#rRprWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#rRprWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#rRpriWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#rRpriWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#rWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#rWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#rWpiWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#rWpiWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#riRLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#riRLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#riRpiWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#riRpiWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#riRprWLockAt -->
```

```
    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#riRprWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#riRpriWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#riRpriWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#riWLockAt -->

    <owl:ObjectProperty rdf:about="http://www.tecweb.inf.puc-
rio.br/synth/locking.owl#riWLockAt">
        <rdfs:range rdf:resource="&rdf;Property"/>
    </owl:ObjectProperty>



    <!--
    ///////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////
     -->




    <!-- http://www.w3.org/1999/02/22-rdf-syntax-ns#Property -->

    <owl:Class rdf:about="&rdf;Property"/>



    <!--
    ///////////////////////////////////////////////////////////////////////
    //
    // Individuals
    //
    ///////////////////////////////////////////////////////////////////////
     -->




    <!-- http://www.tecweb.inf.puc-rio.br/synth/locking.owl#all -->

    <owl:Thing rdf:about="http://www.tecweb.inf.puc-rio.br/synth/locking.owl#all">
        <rdf:type rdf:resource="&owl;NamedIndividual"/>
        <rdfs:label xml:lang="en">Everything</rdfs:label>
        <rdfs:comment xml:lang="en">Pseudo-resource that represents all
elements.</rdfs:comment>
    </owl:Thing>
</rdf:RDF>



<!-- Generated by the OWL API (version 3.2.3.1824) http://owlapi.sourceforge.net -->
```

A.4.
Código Fonte da DSL Interna para *Workflow* de Transações Web

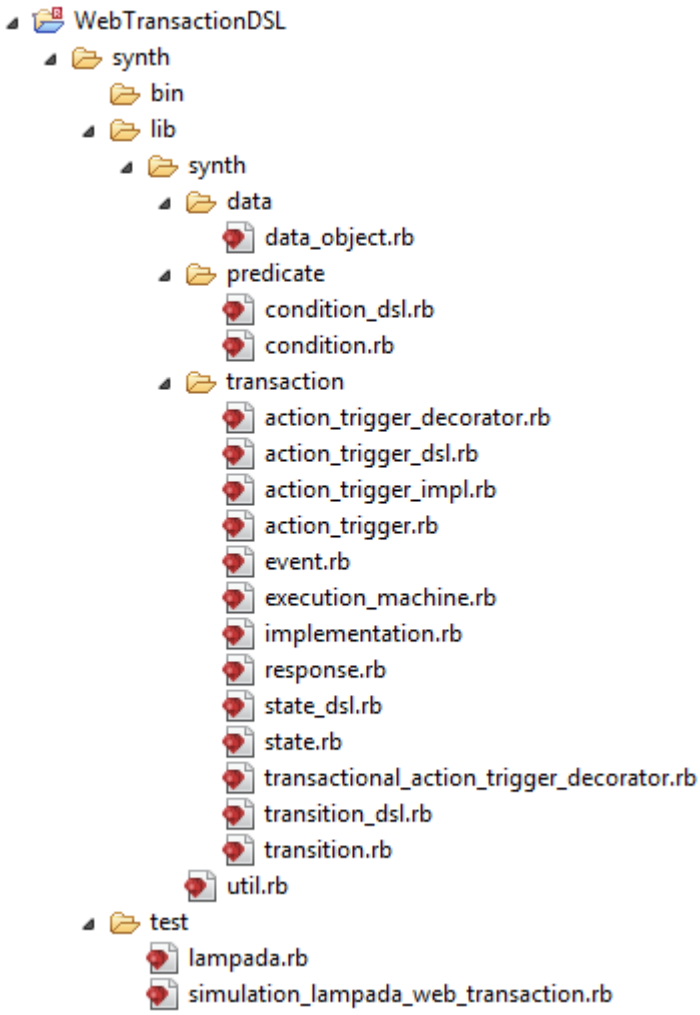

Figura 87 - Pacotes da DSL interna de *workflow*

data_object.rb

```
require 'ostruct'

module Synth

  module Data
    class DataObject < OpenStruct

      attr_reader :name

      def initialize(name=nil, hash=nil)
        super hash
        @name = name
      end

      def to_s
        if @name
          super << " name = #{@name} "
        else
          super
```

```
        end
      end
    end
  end

end
```

condition.rb

```
module Synth
  module Predicate

    class Condition

      attr_reader :body

      def initialize(body)
        raise TypeError, "body - Proc argument expected" unless body.is_a? Proc
        @body = body
      end

      def true_at?(context)
        raise ArgumentError, "Expected context" unless context
        context.instance_eval &@body
      end

      def ==(other)
        if other.is_a? Condition
          @body == other.body
        else
          false
        end
      end

      alias eql? ==

      def hash
        result = 17
        result = 37*result + @body.hash
        result
      end

      #constants

      TRUE  = Condition.new lambda {true}
      FALSE = Condition.new lambda {false}
    end

  end

end
```

condition_dsl.rb

```
require 'synth/util'

module Synth

  module Predicate

    module ConditionDSL

      def condition(&block)
        Synth::Util.madantory_block(block)
        Synth::Predicate::Condition.new block
      end

    end

  end

end
```

state.rb

```
module Synth

  module Transaction

    class State

      attr_reader :name, :definition

      def initialize(name, definition, initial=false)
        raise TypeError, "name - Symbol-like argument expected" \
              unless name.respond_to? :to_sym
        raise TypeError,
              "definition - Synth::Predicate:Condition-like argument expected" \
               unless definition.respond_to? :true_at?

        @name = name
        @definition = definition
        @initial = initial
        @event_transitions = {}
        @automatic_transitions = []
        @dummy_transitions = []

      end

      def add_transitions(*transitions)

        transitions.each do |transition|
          raise TypeError,
           "transition - Synth::Transaction::Transition-like argument expected"\
           if (Synth::Transaction::Transition.public_instance_methods -
                transition.public_methods).length > 0
        end

        transitions.each do |transition|

          if transition.dummy?
              @dummy_transitions << transition
            elsif transition.automatic?
              @automatic_transitions << transition
            else
              temp = @event_transitions[transition.event_name]
              unless temp
                temp = []
                @event_transitions[transition.event_name] = temp
              end
              temp << transition
          end
        end
      end

      def current?(context)
        raise ArgumentError, "Expected context" unless context
        @definition.true_at? context
      end

      def initial?
        @initial
      end

      def final?
        @dummy_transitions.empty? and
        @automatic_transitions.empty? and
        @event_transitions.empty?
      end

      def activate_automatic_transition(context)
        activated = false
        response = nil

        activated, response = activate_transition @dummy_transitions, context

        activated, response = \
```

```
          activate_transition @automatic_transitions, context unless activated

        nil
      end

      def handle_event(context, event)
          raise TypeError,
                "event - Synth::Transaction::Event like argument expected" \
                unless event.respond_to? :name

          transitions = @event_transitions[event.name]
          raise "Missing Transition for the #{event.name} event at #{name} state" \
                 unless transitions

          activated, response = activate_transition transitions, context, event

          response
      end

      def activate_transition(transitions, context, event=nil)
        raise ArgumentError, "Expected context" unless context
        activated = false
        response = nil

        transitions.each do |t|
          activated, response = t.activate context, event
          break if activated
        end

        return activated, response
      end

      private :activate_transition, :handle_event,
              :activate_automatic_transition, :add_transitions

      def self.initial
        State.new :initial, Synth::Predicate::Condition::FALSE
      end
    end

  end

end
```

state_dsl.rb

```
require 'synth/util'
require 'condition_dsl'

module Synth

  module Transaction

    module StateDSL
      include Synth::Predicate::ConditionDSL

      def add_state(name)
        raise NotImplementedError,
        "#{self.class.name}#add_state() abstract method is not
implemented."
      end

      def state(*state_obj, &block)
        result = Synth::Util.obj_xor_block(state_obj, block)
        unless result.is_a? Proc
          add_state result
          return result
        end
```

```
        eigen_class = class << self; self; end
        unless respond_to? :name= and
               respond_to? :definition and
               respond_to? :initial #métodos da DSL a definir

          name = nil
          definition = nil
          initial_flag = false

          #métodos da DSL usando closure para encapsular as variáveis
          #métodos na eigenclass para cada instância ter suas próprias
variáveis
          #Desta forma fica thread safe entre instâncias. Ou seja, se
apenas um
          #único thread manipular a instância é thread-safe.
          eigen_class.class_eval do

            define_method :name do |n|
              name = n
            end

            define_method :definition do |*definition_obj, &body|
              definition = Synth::Util.obj_xor_block(definition_obj,
body)
              return unless definition.is_a? Proc

              definition = condition &body
            end

            define_method :initial do
              initial_flag = true
            end

            private :name, :definition, :initial

            @clear_last_state_definition = lambda {
              name = nil
              definition = nil
              initial = false
            }

            @last_state_name = lambda {name}
            @last_state_definition = lambda {definition}
            @last_state_initial_flag = lambda {initial_flag}

          end

        end

        #reset
        eigen_class.instance_eval {@clear_last_state_definition.call}

        instance_eval &block

        result = eigen_class.instance_eval {
          Synth::Transaction::State.new @last_state_name.call,
                                        @last_state_definition.call,
                                        @last_state_initial_flag.call
        }
```

```
        add_state result
        result
      end

      private :state
    end

  end

end
```

transition.rb

```
require 'condition'
require 'action_trigger_impl'
require 'action_trigger'

module Synth

  module Transaction

    class Transition

      #attr_reader :from_state
      attr_reader :guard
      attr_reader :action
      attr_reader :event_name

      def initialize(action=Synth::Transaction::ActionTriggerImpl::NIL,
                     event_name=nil,
                     guard=Synth::Predicate::Condition::TRUE,
                     *target_states)
        #raise ArgumentError, "Expected from state" unless from_state


        raise TypeError, "action - ActionTrigger-like argument expected"\
              if (Synth::Transaction::ActionTrigger.public_instance_methods -
              action.public_methods).length > 0

        raise TypeError, "event_name - Symbol-like argument expected" \
              if event_name and not event_name.respond_to? :to_sym

        raise TypeError,
              "guard - Synth::Predicate:Condition-like argument expected" \
              unless guard.respond_to? :true_at?

        raise ArgumentError, "Expected one or more target states" \
              unless target_states.length > 0

        @action = action
        @event_name = event_name.to_sym if event_name
        @guard = guard
        @target_states = target_states
        @target_states.freeze

      end

      def target_states
        @target_states.frozen? ? @target_states : @target_states.clone
      end

      def automatic?
        not @event_name
      end

      def has_guard?
        @guard != Synth::Predicate::Condition::TRUE
      end

      def has_action
        @action != Synth::Transaction::ActionTriggerImpl::NIL
```

```
      end

      def dummy?
        automatic? and not has_guard? and not has_action
      end

      def activate(context, event=nil)
        raise TypeError,
        "context - Synth::Transaction::Implementation like argument expected" \
        unless context.respond_to?(:register_action_trigger, true) and
               context.respond_to?(:update_state, true)

        raise ArgumentError, "Expected event" if not event and not automatic?

        result = false
        response = nil
        if guard.true_at? context
          action = @action.copy context, event
          response = action.execute

          target_state = @target_states.select {|s| s.current? context}.first

          if target_state
            context.send :update_state, target_state
            context.send :register_action_trigger, action
            result = true
          else
            action.unexecute #a transicao nao deveria ter ocorrido
            response = nil
            raise "Missing target state" unless dummy?
          end

        end

        return result, response
      end

      def self.dummy(*target_states)
        Transition.new Synth::Transaction::ActionTriggerImpl::NIL,
                       nil,
                       Synth::Predicate::Condition::TRUE,
                       *target_states
      end

    end

  end

end
```

transition_dsl.rb

```
require 'synth/util'
require 'condition_dsl'

module Synth

  module Transaction

    module TransitionDSL
      include  Synth::Transaction::ActionTriggerDSL
      include  Synth::Predicate::ConditionDSL

      def state_by_name(name)
        raise NotImplementedError,
        "#{self.class.name}#state_by_name() abstract method is not implemented."
      end

      def transition(*transition_obj, &block)
        result = Synth::Util.obj_xor_block(transition_obj, block)
        return result unless result.is_a? Proc

        eigen_class = class << self; self; end
        unless respond_to? :action     and
               respond_to? :event_name and
```

```
            respond_to? :guard      and
            respond_to? :target_states #métodos da DSL a definir

          action = Synth::Transaction::ActionTriggerImpl::NIL
          event_name = nil
          guard = Synth::Predicate::Condition::TRUE
          target_states = []

          #métodos da DSL usando closure para encapsular as variáveis
          #métodos na eigenclass para cada instância ter suas próprias variáveis
          #Desta forma fica thread safe entre instâncias. Ou seja, se apenas um
          #único thread manipular a instância é thread-safe.
          eigen_class.class_eval do

            define_method :action do |*action_obj, &body|
              action = Synth::Util.obj_xor_block(action_obj, body)
              return unless action.is_a? Proc

              action = action_trigger &body
            end

            define_method :event_name do |name|
              event_name = name
            end

            define_method :guard do |*guard_obj, &body|
              guard = Synth::Util.obj_xor_block(guard_obj, body)
              return unless guard.is_a? Proc

              guard = condition &body
            end

            define_method :target_states do |*states|
              states.each do |s|
                if s.respond_to? :to_sym
                  state = state_by_name s.to_sym #operacao abstrata
                else
                  state = s
                end
                target_states << state
              end
            end

            alias :target_state :target_states

            private :action, :event_name, :guard, :target_states, :target_state

            @clear_last_transition_definition= lambda {
              action = Synth::Transaction::ActionTriggerImpl::NIL
              event_name = nil
              guard = Synth::Predicate::Condition::TRUE
              target_states = []
            }

            @last_transition_action = lambda {action}
            @last_transition_event_name = lambda {event_name}
            @last_transition_guard = lambda {guard}
            @last_transition_target_states = lambda {target_states}

          end

        end

        #reset
        eigen_class.instance_eval {@clear_last_transition_definition.call}

        instance_eval &block

        eigen_class.instance_eval {
          Synth::Transaction::Transition.new(
            @last_transition_action.call,
            @last_transition_event_name.call,
            @last_transition_guard.call,
            *@last_transition_target_states.call)
```

```
        }

      end

      private :transition

    end

  end

end
```

action_trigger.rb

```
module Synth
  module Transaction
    module ActionTrigger

      def copy(context, event=nil)
        raise NotImplementedError,
        "#{self.class.name}#copy() abstract method is not implemented."
      end

      def execute
        raise NotImplementedError,
        "#{self.class.name}#execute() abstract method is not implemented."
      end

      def unexecute
        raise NotImplementedError,
        "#{self.class.name}#unexecute() abstract method is not implemented."
      end

      def previous_state
        raise NotImplementedError,
        "#{self.class.name}#previous_state() abstract method is not implemented."
      end

      def isolation_level()
        raise NotImplementedError,
        "#{self.class.name}#isolation_level() abstract method is not implemented."
      end

      #retorna o contexto, ou seja, a transação Web em curso.
      def context()
        raise NotImplementedError,
        "#{self.class.name}#context() abstract method is not implemented."
      end

      #retorna o evento que dispareou este ActionTrigger
      def event()
        raise NotImplementedError,
        "#{self.class.name}#event() abstract method is not implemented."
      end

      #retorna o a resposta, caso haja, após a execução do ActionTrigger
      def response()
        raise NotImplementedError,
        "#{self.class.name}#response() abstract method is not implemented."
      end

    end
  end
end
```

action_trigger_impl.rb

```
require 'action_trigger'

module Synth

  module Transaction

    class ActionTriggerImpl
      include Synth::Transaction::ActionTrigger
```

```
      #@decorator = nil
      def self.decorator=(decorator)
        @decorator = decorator
      end

      attr_reader :isolation_level, :context, :event, :response
      attr_reader :execute_body, :unexecute_body
      protected :execute_body, :unexecute_body

      def self.new(execute_body, unexecute_body, isolation_level=nil)
        result = super
        if @decorator
          decorator = @decorator.clone
          decorator.component = result
          result = decorator
        end
        result
      end

      def initialize(execute_body, unexecute_body, isolation_level=nil)
        raise TypeError, "isolation_level - Symbol-like argument expected" \
              if isolation_level and not isolation_level.respond_to? :to_sym

        raise TypeError, "execute_body - Proc argument expected" \
              unless execute_body.is_a? Proc

        raise TypeError, "unexecute_body - Proc argument expected" \
              unless execute_body.is_a? Proc

        @execute_body = execute_body
        @unexecute_body = unexecute_body
        @isolation_level = isolation_level ? isolation_level.to_sym : nil
        @context = nil
        @event = nil
        @response = nil
        @executed = false

      end

      def copy(context, event=nil)
          raise ArgumentError, "Expected context" unless context
          result = clone

          #define os métodos abaixo na eigenclass de result, usando closure
          #para que o context e o evento nunca sejam alterados
          eigen_class = class << result; self; end

          eigen_class.class_eval do
            define_method :context do
              context
            end

            define_method :event do
              event
            end

          end

          result
      end

      def execute
          raise "Context undefined" unless context

          #define os métodos abaixo na eigenclass de result, usando closure
          #para que o context e o evento nunca sejam alterados
          state = context.instance_eval {state}
          eigen_class = class << self; self; end
          eigen_class.class_eval do
            define_method :previous_state do
              state
            end
          end
```

```
          @response = instance_eval &@execute_body
          @executed = true
          @response
      end

      def unexecute
          raise "Context undefined" unless context

          instance_eval &@unexecute_body if @executed
          @executed = false
          @response = nil
      end

      def ==(other)
        return false unless other.is_a? ActionTriggerImpl

        @isolation_level == other.isolation_level &&
        @execute_body == other.execute_body &&
        @unexecute_body == other.unexecute_body
      end

      def eql?(other)
        self == other &&
        @context == other.context &&
        @event == other.event &&
        @response == other.response
      end

      def hash
        result = 17
        result = 37*result + @isolation_level.hash
        result = 37*result + @execute_body.hash
        result = 37*result + @unexecute_body.hash
        result = 37*result + @context.hash
        result = 37*result + @event.hash
        result = 37*result + @response.hash
      end

      #constants
      NIL = ActionTriggerImpl.new lambda {}, lambda {}

    end

  end

end
```

action_trigger_dsl.rb

```
require 'synth/util'

module Synth

  module Transaction

    module ActionTriggerDSL

      def action_trigger(*action_trigger_obj, &block)
        result = Synth::Util.obj_xor_block(action_trigger_obj, block)
        return result unless result.is_a? Proc

        eigen_class = class << self; self; end
        unless respond_to? :isolation_level   and
               respond_to? :execute_statement and
               respond_to? :unexecute_statement #métodos da DSL a definir

          isolation_level = nil
          execute_body    = nil
          unexecute_body  = nil

          #métodos da DSL usando closure para encapsular as variáveis
          #métodos na eigenclass para cada instância ter suas próprias variáveis
          #Desta forma fica thread safe entre instâncias. Ou seja, se apenas um
          #único thread manipular a instância é thread-safe.
          eigen_class.class_eval do
```

```
            define_method :isolation_level do |level|
              isolation_level = level
            end

            define_method :execute_statement do |&body|
              execute_body = Synth::Util.madantory_block(body)
            end

            define_method :unexecute_statement do |&body|
              unexecute_body = Synth::Util.madantory_block(body)
            end

            private :isolation_level, :execute_statement, :unexecute_statement

            @clear_last_action_trigger_definition = lambda {
              isolation_level = nil
              execute_body    = nil
              unexecute_body  = nil
            }

            @last_action_trigger_isolation_level = lambda {isolation_level}
            @last_action_trigger_execute_body    = lambda {execute_body}
            @last_action_trigger_unexecute_body  = lambda {unexecute_body}

          end

        end

        #reset
        eigen_class.instance_eval {@clear_last_action_trigger_definition.call}

        instance_eval &block

        eigen_class.instance_eval {
          Synth::Transaction::ActionTriggerImpl.new(
            @last_action_trigger_execute_body.call,
            @last_action_trigger_unexecute_body.call,
            @last_action_trigger_isolation_level.call)
        }
      end

      private :action_trigger

    end

  end

end
```

action_trigger_decorator.rb

```
require 'action_trigger'

module Synth
  module Transaction
    module ActionTriggerDecorator
      include Synth::Transaction::ActionTrigger

      def component=(component)
        @component = component
      end

      def copy(context, event=nil)
        result = clone
        result.component = @component.copy context, event
        result
      end

      def execute()
        @component.execute
      end

      def unexecute
        @component.unexecute
```

```
      end

      def previous_state
        @component.previous_state
      end

      def isolation_level()
        @component.isolation_level
      end

      #retorna o contexto, ou seja, a transação Web em curso.
      def context()
        @component.context
      end

      #retorna o evento que dispareou este ActionTrigger
      def event()
        @component.event
      end

      #retorna o a resposta, caso haja, após a execução do ActionTrigger
      def response()
        @component.response
      end

    end
  end
end
```

transactional_action_trigger_decorator.rb

```
require 'action_trigger_decorator'

module Synth
  module Transaction
    module TransactionalActionTriggerDecorator
      include Synth::Transaction::ActionTriggerDecorator

      def begin_database_transaction(isolation_level)
        raise NotImplementedError,
        "#{self.class.name}#begin_database_transaction() abstract method is not
implemented."
      end

      def commit_database_transaction()
        raise NotImplementedError,
        "#{self.class.name}#commit_database_transaction() abstract method is not
implemented."
      end

      def rollback_database_transaction()
        raise NotImplementedError,
        "#{self.class.name}#rollback_database_transaction() abstract method is not
implemented."
      end

      def clean_up()
        raise NotImplementedError,
        "#{self.class.name}#close() abstract method is not implemented."
      end

      def execute()
        begin
          begin_database_transaction(isolation_level)
          response = super
          commit_database_transaction
          response
        rescue Exception => ex
          rollback_database_transaction
          raise ex
        ensure
          clean_up
        end
      end
```

```
    end
  end
end
```

event.rb

```
require 'data_object'

module Synth

  module Transaction

    class Event < Synth::Data::DataObject

      def initialize(name, hash=nil)
        super
      end

    end

  end

end
```

response.rb

```
require 'data_object'

module Synth

  module Transaction

    class Response < Synth::Data::DataObject

    end

  end

end
```

execution_machine.rb

```
require 'condition_dsl'
require 'action_trigger_dsl'
require 'state_dsl'
require 'transition_dsl'
require 'state'
require 'transition'
require 'synth/util'


module Synth

  module Transaction

    class ExecutionMachine

      include Synth::Predicate::ConditionDSL
      include Synth::Transaction::ActionTriggerDSL
      include Synth::Transaction::StateDSL
      include Synth::Transaction::TransitionDSL

      attr_reader :initial_pseudo_state

      def initialize(owner, &block)
        raise TypeError, "Module argument expected" \
            unless owner.is_a? Module
        Synth::Util.madantory_block(block)
        @states = {}
        @initial_pseudo_state = Synth::Transaction::State.initial
        instance_eval &block
        this = self
        owner.instance_eval {@execution_machine = this}
      end

      def states
```

```
        @states.clone #copia defensiva (encpasulamento)
      end

      def add_state(state)
        @states ||= {}
        @states[state.name] = state
        if state.initial?
          transition = Synth::Transaction::Transition.dummy state
          @initial_pseudo_state.send :add_transitions, transition
        end
      end

      def state_by_name(name)
        @states[name]
      end

      def assign_transitions(state, *transitions)
        if state.respond_to? :to_sym
          state = state_by_name state.to_sym
        end
        state.send :add_transitions, *transitions
      end

      private :add_state, :assign_transitions

      def method_missing(name, *args, &block)
        return super unless name.to_s =~ /on_.*/
        parts = name.to_s.split("_")
        parts.shift
        parts.each { |state_name| assign_transitions state_name, *args }
      end

    end



  end

end
```

implementation.rb

```
module Synth

  module Transaction

    module Implementation

      attr_reader :id

      def initialize(id)
        @id = id
      end

      def initial_pseudo_state
        self.class.instance_eval {@execution_machine.initial_pseudo_state}
      end

      #para ser chamado dinamicamente pela classe Transition
      def register_action_trigger(action_trigger)
        @action_triggers ||= []
        @action_triggers << action_trigger
      end

      #para ser chamado dinamicamente pela classe Transition
      def update_state(state)
        @state = state
      end

      def state
        @state ||= initial_pseudo_state
        @state.send :activate_automatic_transition, self
        @state
      end
```

```
      private :initial_pseudo_state, :register_action_trigger, :update_state

      #tratar um evento externo (em geral, oriundo de uma requisição http)
      def handle_event(event)
        state.send :handle_event, self, event
      end

      def commit
        #se não for nested transaction, ou seja, se for a transação raiz da
        #árvore de transações, chamar a operação commit do UnitOfWork,
        #a ser implementado ainda, para persistir todos as modificações no BD.
      end

      def rollback
        @action_triggers.reverse_each do |current|
          current.unexecute
          update_state current.previous_state
        end
      end

    end

  end

end
```

util.rb

```
module Synth

  module Util

    def self.obj_xor_block(obj, block)
      raise "Expected mutually exclusive a block or only one argument" \
      unless obj.is_a?(Array) && (obj.length <= 1) &&
             (!obj[0] && block && block.is_a?(Proc) || obj[0] && !block)

      obj[0] ? obj[0] : block
    end

    def self.madantory_block(block)
      raise ArgumentError, "Expected block" unless block
      block
    end

  end

end
```

lampada.rb

```
class Lampada

  def initialize
    @filamento_aceso = false
    @filamento_ok = true
    @num_vezes_acendeu = 0
  end

  def acender
    unless queimada?
      unless acesa?
        @filamento_aceso = true
        @num_vezes_acendeu += 1
        queimar if @num_vezes_acendeu == 2
      end
    end
  end

  def apagar
    @filamento_aceso = false
  end

  def acesa?
    @filamento_aceso
  end
```

```
  def apagada?
    not acesa?
  end

  def queimada?
    not @filamento_ok
  end

  def queimar
    @filamento_aceso = false
    @filamento_ok = false
  end
  private :queimar

  #create state snapshot - para fins de rollback
  def create_memento
    Memento.new @filamento_aceso, @filamento_ok, @num_vezes_acendeu
  end

  #restore state snapshot durante a execução de rollback
  def set_memento(memento)
    @filamento_aceso = memento.filamento_aceso
    @filamento_ok = memento.filamento_ok
    @num_vezes_acendeu = memento.num_vezes_acendeu
  end

  #class que gera o snapshot, sem quebrar encapsulamento
  class Memento
    attr_reader :filamento_aceso, :filamento_ok, :num_vezes_acendeu
    def initialize(filamento_aceso, filamento_ok, num_vezes_acendeu)
      @filamento_aceso, @filamento_ok, @num_vezes_acendeu =
        filamento_aceso, filamento_ok, num_vezes_acendeu
    end
  end

end
```

simulacao_lampada_Web_transaction.rb

```
require 'execution_machine'
require 'implementation'


class SimulacaoLampadaTransaction
  include Synth::Transaction::Implementation

  attr_reader :subject

  def initialize(id, subject)
    super id
    @subject = subject
  end


  Synth::Transaction::ExecutionMachine.new(self) {

    #estados
    state {
      initial
      name  :apagada
      definition {subject.apagada?}
    }

    state {
      name  :acesa
      definition {subject.acesa?}
    }


    state {
      name  :queimada
      definition {subject.queimada?}
    }
```

```
    #transições

    acender = transition {
      action {
        isolation_level  :read_comitted
        execute_statement {
          @memento = context.subject.create_memento
          context.subject.acender
          context.commit if context.subject.queimada? #commit da Web Transaction
        }
        unexecute_statement {
          context.subject.set_memento @memento
        }
      }

      event_name :acender_event
      target_states :acesa, :queimada

    }

    on_apagada acender

    apagar = transition {
      action {
        isolation_level :read_comitted
        execute_statement {
          @memento = context.subject.create_memento
          context.subject.apagar
        }
        unexecute_statement {
          context.subject.set_memento @memento
        }
      }

      event_name :apagar_event

      target_state :apagada

    }

    on_acesa apagar

  }

end
```

A.5.
Código Fonte da Simulação

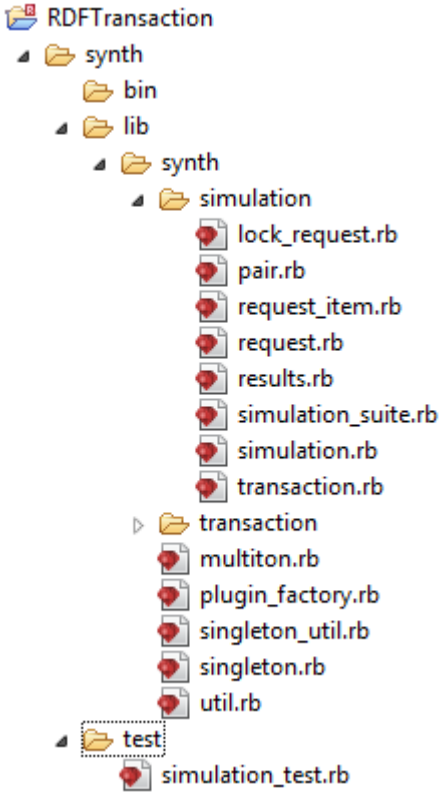

Figura 88 - Pacotes da simulação

Pair.rb

```
module Synth
  module Simulation
    class Pair

      attr_reader :property, :resource

      def initialize(property, resource)
        @property = property
        @resource = resource
      end

      def to_s
        "(p:#{@property}, r:#{@resource})"
      end

    end
  end
end
```

simulation.rb

```
require 'thread'
require 'pair'
require 'synth/simulation/transaction'
require 'synth/simulation/results'

module Synth
  module Simulation
    class Simulation
      attr_reader :lock_manager,
```

```
                :num_resources,
                :num_properties,
                :num_transactions,
                :min_requests_per_transaction,
                :max_requests_per_transaction,
                :min_pairs_per_request,
                :max_pairs_per_request,
                :min_user_thinking_time,
                :max_user_thinking_time,
                :min_restart_time,
                :max_restart_time,
                :io_time,
                :inv_properties_percentage,
                :read_lock_modes_range,
                :write_lock_modes_range,
                :write_lock_percentages,
                :arrival_rates



    def initialize(lock_manager,
                   num_resources,
                   num_properties,
                   num_transactions,
                   min_requests_per_transaction,
                   max_requests_per_transaction,
                   min_pairs_per_request,
                   max_pairs_per_request,
                   min_user_thinking_time,
                   max_user_thinking_time,
                   min_restart_time,
                   max_restart_time,
                   io_time,
                   inv_properties_percentage,
                   avoid_starvation,
                   read_lock_modes_range=nil,
                   write_lock_modes_range=nil)

      num_max_pairs = num_resources * num_properties
      num_max_pairs_per_transaction =
        max_requests_per_transaction * max_pairs_per_request
      raise ArgumentError,
            "Maximum Locks per Transaction to high!" if (
              num_max_pairs_per_transaction > num_max_pairs)

      @lock_manager = lock_manager
      @num_resources  = num_resources
      @num_properties = num_properties
      @num_transactions = num_transactions
      @min_requests_per_transaction = min_requests_per_transaction
      @max_requests_per_transaction = max_requests_per_transaction
      @min_pairs_per_request = min_pairs_per_request
      @max_pairs_per_request = max_pairs_per_request

      @read_lock_modes_range  =
        read_lock_modes_range ?
          read_lock_modes_range : lock_manager.rR..lock_manager.riR
      @write_lock_modes_range =
        write_lock_modes_range ?
          write_lock_modes_range : lock_manager.rW..lock_manager.riW

      @min_user_thinking_time = min_user_thinking_time
      @max_user_thinking_time = max_user_thinking_time
      @min_restart_time = min_restart_time
      @max_restart_time = max_restart_time
      @io_time = io_time
      @inv_properties_percentage = inv_properties_percentage

      @write_lock_percentages = nil
      @arrival_rates = nil

      create_pairs_table
      create_transactions
```

```
    @starvation = false
    @mutex = Mutex.new

    @avoid_starvation = avoid_starvation
    @acquire_locks_condition_variable = ConditionVariable.new
    @acquire_locks_busy = false
  end

  def synchronized(&block)
      @mutex.synchronize &block if block
  end

  def starvation?()
    synchronized {@starvation}
  end

  def starvation=(flag)
     synchronized {@starvation = flag}
  end

  def avoid_starvation?()
    @avoid_starvation
  end

  def before_try_acquire_locks()
    return unless avoid_starvation?
    synchronized do
      @acquire_locks_condition_variable.wait(@mutex) while (@acquire_locks_busy)
      @acquire_locks_busy = true
    end
  end

  def after_try_acquire_locks()
    return unless avoid_starvation?
    synchronized do
      @acquire_locks_busy = false
    end
    @acquire_locks_condition_variable.signal
  end

  def create_pairs_table()
    @pairs_table = []
    0.upto(@num_properties-1) do |p|
      0.upto(@num_resources-1) do |r|
          @pairs_table << Pair.new(p,r)
      end
    end
  end
  private :create_pairs_table

  def create_transactions
    @transactions = []
    1.upto(@num_transactions) do |id|
      @transactions << Synth::Simulation::Transaction.new(self, id)
    end
  end
  private :create_transactions

  def num_max_pairs()
    @pairs_table.length
  end

  def max_pair()
    num_max_pairs - 1
  end

  def inv_property(property)
    result = nil
    delta = @num_properties * @inv_properties_percentage
    delta = delta.round
    return result if delta == 0

    #arbitrariamente inv_p = N-1-p, onde N é o numero de propriedades
```

```
        #ou seja, propriedades equidistantes de acordo com seu número
        result = @num_properties-1-property

        if result == property ||
           !(result < delta || @num_properties-1-result < delta)
           result = nil
        end
        result
      end

      def pair(pair_index)
        @pairs_table[pair_index]
      end

      def user_thinking_time()
        delta = @max_user_thinking_time - @min_user_thinking_time
        @min_user_thinking_time + rand(delta+1)
      end

      def restart_time(num_aborts)
        min = @min_restart_time
        max = @max_restart_time
        delta = max - min
        min + rand(delta+1)
      end

      def set_write_lock_percentage(percentage)
        @transactions.each do |t|
          t.set_write_lock_percentage(percentage,
                                      @read_lock_modes_range,
                                      @write_lock_modes_range)
        end
      end
      private :set_write_lock_percentage

      def set_variable(variable)
        @transactions.each {|t| yield t, variable}
      end

      def measured_arrival_rate()
        arrival_rate = 0
        if @num_transactions > 1 && @arrival_latency_total
          average_arrival_latency =
            @arrival_latency_total.to_f/(@num_transactions-1)
          arrival_rate = 1.0/average_arrival_latency
        end
        arrival_rate
      end
      private :measured_arrival_rate

      def start_transactions(arrival_rate)
        @threads = []
        int_arrival_latency = 1.0/arrival_rate
        @arrival_latency_total = 0
        count = @num_transactions

        @transactions.each do |t|
          @threads << Thread.new { t.simulate }
          count -= 1
          if int_arrival_latency > 0 && count > 0
            before = Time.new
            sleep(int_arrival_latency)
            after = Time.new
            @arrival_latency_total += (after-before)
          end

        end
      end

      private :start_transactions

      def print_threads_status
        @threads.each {|t| puts "\n\nThread status error = #{t.value}" if t.status ==
nil}
```

```
      end
      private :print_threads_status

      def wait_for_transactions()
        main = Thread.main
        current = Thread.current
        Thread.list.each {|t| t.join unless t == current or t == main}
      end
      private :wait_for_transactions

      def collect_results(start_time, end_time)
        puts "Simulacao : #{@simulation_id}\n"
        response_time = 0
        real_response_time = 0
        turnaround_time = 0
        real_turnaround_time = 0
        restart_time = 0
        num_aborts = 0
        num_graph_locks = 0
        num_property_locks = 0
        num_resource_locks = 0
        num_property_of_resource_locks = 0
        num_locks = 0
        lock_time = 0
        unlock_all_time = 0

        total_user_thinking_time = 0

        @transactions.each do |t|
          response_time += t.response_time
          real_response_time += t.real_response_time
          turnaround_time += t.turnaround_time
          real_turnaround_time += t.real_turnaround_time
          restart_time += t.restart_time
          num_aborts += t.num_aborts
          lock_time += t.lock_time
          unlock_all_time += t.unlock_all_time

          temp = t.num_locks_per_granule
          num_graph_locks += temp[@lock_manager.graph]
          num_property_locks += temp[@lock_manager.property]
          num_resource_locks += temp[@lock_manager.resource]
          num_property_of_resource_locks += temp[@lock_manager.property_of_resource]

          num_locks += t.num_locks

          total_user_thinking_time += t.total_user_thinking_time
        end

        simulation_time = end_time - start_time
        troughput_with_arrival_latency = @num_transactions.to_f/simulation_time

        simulation_time_without_arrival_latency =
          simulation_time - @arrival_latency_total
        troughput_without_arrival_latency =
          @num_transactions.to_f/simulation_time_without_arrival_latency

        puts "Taxa de chegada de transações medida: #{measured_arrival_rate} t/s"
        puts "Tempo de resposta: #{response_time.to_f/@num_transactions} s"
        puts "Tempo de resposta real: #{real_response_time.to_f/@num_transactions/60.0}
min"
        puts "Turnaround: #{turnaround_time.to_f/@num_transactions} s"
        puts "Turnaround real: #{real_turnaround_time.to_f/@num_transactions/60.0} min"
        puts "Tempo para reiniciar: #{restart_time.to_f/@num_transactions} s"
        puts "Throughput with arrival latency: #{troughput_with_arrival_latency} t per s"
        puts "Throughput without arrival latency: #{troughput_without_arrival_latency} t
per s"
        puts "Numero de abortos: #{num_aborts.to_f/@num_transactions}"
        puts "Numero de locks - grafo: #{num_graph_locks.to_f/@num_transactions}"
        puts "Numero de locks - property: #{num_property_locks.to_f/@num_transactions}"
        puts "Numero de locks - recurso #{num_resource_locks.to_f/@num_transactions}"
        puts "Numero de locks - prop. de recurso
#{num_property_of_resource_locks.to_f/@num_transactions}"
        puts "Numero de locks: #{num_locks.to_f/@num_transactions}"
```

```
        puts "Tempo de para obter um lock: #{lock_time.to_f/@num_transactions*1000} ms"
        puts "Tempo de para liberar todos os locks:
#{unlock_all_time.to_f/@num_transactions*1000} ms"
        puts "Duracao da simulacao: #{(end_time - start_time)/60.0} min"

        puts "User Thinking Time: #{total_user_thinking_time.to_f/@num_transactions} s"
      end
      private :collect_results


      def calculate_results(start_time, end_time)
        response_time = 0
        real_response_time = 0
        turnaround_time = 0
        real_turnaround_time = 0
        restart_time = 0
        num_aborts = 0
        num_graph_locks = 0
        num_property_locks = 0
        num_resource_locks = 0
        num_property_of_resource_locks = 0
        num_locks = 0
        lock_time = 0
        unlock_all_time = 0

        total_user_thinking_time = 0

        @transactions.each do |t|
          response_time += t.response_time
          real_response_time += t.real_response_time
          turnaround_time += t.turnaround_time
          real_turnaround_time += t.real_turnaround_time
          restart_time += t.restart_time
          num_aborts += t.num_aborts
          lock_time += t.lock_time
          unlock_all_time += t.unlock_all_time

          temp = t.num_locks_per_granule
          num_graph_locks += temp[@lock_manager.graph]
          num_property_locks += temp[@lock_manager.property]
          num_resource_locks += temp[@lock_manager.resource]
          num_property_of_resource_locks +=
            temp[@lock_manager.property_of_resource]

          num_locks += t.num_locks

          total_user_thinking_time += t.total_user_thinking_time
        end
        response_time = response_time.to_f/@num_transactions
        real_response_time = real_response_time.to_f/@num_transactions/60.0 #min
        turnaround_time = turnaround_time.to_f/@num_transactions
        real_turnaround_time = real_turnaround_time.to_f/@num_transactions/60.0 #min
        restart_time = restart_time.to_f/@num_transactions
        num_aborts = num_aborts.to_f/@num_transactions
        num_graph_locks = num_graph_locks.to_f/@num_transactions
        num_property_locks = num_property_locks.to_f/@num_transactions
        num_resource_locks = num_resource_locks.to_f/@num_transactions
        num_property_of_resource_locks =
          num_property_of_resource_locks.to_f/@num_transactions
        num_locks = num_locks.to_f/@num_transactions
        lock_time = lock_time.to_f/@num_transactions*1000 #ms
        unlock_all_time = unlock_all_time.to_f/@num_transactions*1000 #ms

        simulation_time = end_time - start_time
        troughput_with_arrival_latency = @num_transactions.to_f/simulation_time

        simulation_time_without_arrival_latency =
          simulation_time - @arrival_latency_total
        troughput_without_arrival_latency =
          @num_transactions.to_f/simulation_time_without_arrival_latency

        total_user_thinking_time = total_user_thinking_time.to_f/@num_transactions

        {
```

```
          :measured_arrival_rate => measured_arrival_rate,
          :response_time => response_time,
          :real_response_time => real_response_time,
          :turnaround_time => turnaround_time,
          :real_turnaround_time => real_turnaround_time,
          :restart_time => restart_time,
          :num_aborts      => num_aborts,
          :num_graph_locks => num_graph_locks,
          :num_property_locks => num_property_locks,
          :num_resource_locks => num_resource_locks,
          :num_property_of_resource_locks => num_property_of_resource_locks,
          :num_locks       => num_locks,
          :lock_time       => lock_time,
          :unlock_all_time => unlock_all_time,
          :troughput_with_arrival_latency => troughput_with_arrival_latency,
          :troughput_without_arrival_latency => troughput_without_arrival_latency,

          :total_user_thinking_time => total_user_thinking_time
        }
      end
      private :calculate_results

      def execute(title,
                  variables,
                  write_lock_percentages = nil,
                  arrival_rates = nil,
                  &variable_block)

        raise ArgumentError, "write_lock_percentages undefined!" if (
                !write_lock_percentages && !@write_lock_percentages)

        raise ArgumentError, "write_lock_percentages must not be empty" if (
                write_lock_percentages && write_lock_percentages.length == 0)

        raise ArgumentError, "arrival_rates undefined!" if (
                !arrival_rates && !@arrival_rates)

        raise ArgumentError, "arrival_rates must not be empty!" if (
                arrival_rates && arrival_rates.length == 0)

        results = Synth::Simulation::Results.new title,
                                                 :measured_arrival_rate,
                                                 :response_time,
                                                 :real_response_time,
                                                 :turnaround_time,
                                                 :real_turnaround_time,
                                                 :restart_time,
                                                 :num_aborts,
                                                 :num_graph_locks,
                                                 :num_property_locks,
                                                 :num_resource_locks,
                                                 :num_property_of_resource_locks,
                                                 :num_locks,
                                                 :lock_time,
                                                 :unlock_all_time,
                                                 :troughput_with_arrival_latency,
                                                 :troughput_without_arrival_latency,
                                                 :total_user_thinking_time

        @write_lock_percentages =
          write_lock_percentages if write_lock_percentages

        @arrival_rates = arrival_rates if arrival_rates

        #srand 1234
        @simulation_id ||= 1
        @write_lock_percentages.each do |write_percentage|
          results.write_percentage = write_percentage
          set_write_lock_percentage(write_percentage) if write_lock_percentages
          variables.each do |variable|
            results.variable = variable
            set_variable variable, &variable_block
            @arrival_rates.each do |arrival_rate|
              results.arrival_rate = arrival_rate
```

```
              puts "Inicio Simulacao as #{Time.new}"

              self.starvation = false
              start_time = Time.new
              start_transactions arrival_rate
              wait_for_transactions
              end_time = Time.new
              print_threads_status

              results.values = calculate_results start_time, end_time
              collect_results start_time, end_time

              #puts "Lock Manager #{@lock_manager.object_id}: #{@lock_manager}\n"
              puts "\nFIM\n"

              @simulation_id += 1
            end
          end
        end

        results.close
      end

      def to_s
        result = "\nTransactions:\n"
        @transactions.each {|t| result += "\n#{t}" }
        result
      end

    end
  end
end
```

transaction.rb

```
require 'synth/simulation/request'
require 'lock_request'


module Synth
  module Simulation
    class Transaction

      attr_reader :simulation,
                  :id,
                  :status,
                  :num_aborts,
                  :lock_time,
                  :unlock_all_time,
                  :restart_time,
                  :total_user_thinking_time

      def initialize(simulation, id)
        @simulation = simulation
        @id = id
        pair_indexes = Range.new(0, simulation.max_pair).to_a
        delta = simulation.max_requests_per_transaction -
                simulation.min_requests_per_transaction
        num_requests = simulation.min_requests_per_transaction+rand(delta+1)
        @requests = []
        @num_pairs = 0
        @properties = []
        @resources = []
        1.upto(num_requests) do |i|
          request = Synth::Simulation::Request.new(self, pair_indexes)

          @requests << request
          @num_pairs += request.num_pairs
          @properties.concat(request.properties)
          @resources.concat(request.resources)
        end
        @properties = @properties.uniq
        @resources = @resources.uniq

      end
```

```
def lock_manager
  @simulation.lock_manager
end
private :lock_manager

def num_locks
  return 0 unless @lock_requests
  @lock_requests.length
end

def num_locks_per_granule()
  result = {}
  result [lock_manager.graph] = 0
  result [lock_manager.property] = 0
  result [lock_manager.resource] = 0
  result [lock_manager.property_of_resource] = 0
  return result unless @lock_requests
  @lock_requests.each do |lock_request|
    count = result[lock_request.granule]
    count += 1
    result[lock_request.granule] = count
  end
  result
end

def set_write_lock_percentage(percentage,
                              read_lock_modes_range,
                              write_lock_modes_range)
  @requests.each do |request|
    request.set_write_lock_percentage(percentage,
                                      read_lock_modes_range,
                                      write_lock_modes_range)
  end
end

def set_read_write_lock_modes(read_lock_mode, write_lock_mode)
  raise ArgumentError,
        "Read Lock Mode Expected" if (
          lock_manager.write_lock_mode? read_lock_mode)
  raise ArgumentError,
        "Write Lock Mode Expected" unless (
          lock_manager.write_lock_mode? write_lock_mode)
  @requests.each do |request|
    request.set_read_write_lock_modes read_lock_mode, write_lock_mode
  end

  fix_granule(lock_manager.property_of_resource)

end

def reset_lock_requests()
  pairs_to_process = @num_pairs
  if pairs_to_process == 0
    @lock_requests = []
    return pairs_to_process
  end

  @requests.each {|request| request.reset_lock_requests}
  @lock_requests = {}

  return pairs_to_process
end
private :reset_lock_requests

def sort_lock_requests()
  @lock_requests = @lock_requests.values.sort
end
private :sort_lock_requests

def fix_granule(granule)

  lock_manager.instance_eval {@mgl = false}
```

```
      pairs_to_process = reset_lock_requests
      return pairs_to_process if pairs_to_process == 0

      transaction = self
      granule_upgrade_dispatcher =
        self.class.send :granule_upgrade_dispatcher, lock_manager
      item = granule_upgrade_dispatcher[granule]
      pairs_to_process = item.check(transaction, pairs_to_process, 0, false)

      if pairs_to_process > 0
        item = granule_upgrade_dispatcher[lock_manager.property_of_resource]
        pairs_to_process = item.check(transaction, pairs_to_process, 0, false)
      end

     sort_lock_requests

    end

    def granule_upgrade_percentage=(percentage)

      lock_manager.instance_eval {@mgl = true}

      pairs_to_process = reset_lock_requests
      return pairs_to_process if pairs_to_process == 0

      transaction = self
      lm = lock_manager
      granule_upgrade_dispatcher = self.class.instance_eval do
        granule_upgrade_dispatcher(lm).values.sort do |x,y|
            x.compare(y, transaction) * -1 #decrescente
          end
      end

      granule_upgrade_dispatcher.each do |item|

        pairs_to_process = item.check(transaction,
                                      pairs_to_process,
                                      percentage,
                                      true)#mgl

        break if pairs_to_process == 0
      end

      sort_lock_requests

    end

    def check_graph(pairs_to_process,
                    total_pairs,
                    percentage,
                    mgl)
     pairs_to_process, lock_request =
        check(percentage,
              pairs_to_process,
              total_pairs,
              lock_manager.graph,
              nil,
              true,
              mgl) do |pair|
                true
              end
     pairs_to_process
    end
    private :check_graph

    def mount_data_item(granule, uris)
      lock_manager.instance_eval do
        mount_data_item granule, uris
      end
    end
    private :mount_data_item

    def lock_inv_property(inv_p,
                          pairs_to_process,
```

```
                        total_pairs,
                        mgl,
                        p_lock_request)

      return pairs_to_process unless inv_p

      data_item = mount_data_item(lock_manager.property,
                                  :property => inv_p)

      inv_p_lock_request = @lock_requests[data_item]

      unless inv_p_lock_request

        pairs_to_process, inv_p_lock_request =
          check(0,
                pairs_to_process,
                total_pairs,
                lock_manager.property,
                p_lock_request.lock_mode,
                false, #não fazer upgrade de lock mode
                mgl,
                :property => inv_p) do |pair|
                  pair.property == inv_p
                end

        unless inv_p_lock_request.confirmed?
          inv_p_lock_request.confirm
          @lock_requests[data_item] = inv_p_lock_request
        end

      end
      inv_p_lock_request.lock_mode = p_lock_request.lock_mode

      return pairs_to_process
    end
    private :lock_inv_property

    def check_property_p(pairs_to_process,
                         total_pairs,
                         percentage,
                         mgl)
      @properties.each do |p|

        pairs_to_process, lock_request =
          check(percentage,
                pairs_to_process,
                total_pairs,
                lock_manager.property,
                nil,
                true,
                mgl,
                :property => p) do |pair|
                  pair.property == p
                end

        if lock_request.confirmed?
          inv_p = @simulation.inv_property p
          pairs_to_process = lock_inv_property(inv_p,
                                               pairs_to_process,
                                               total_pairs,
                                               mgl,
                                               lock_request)
        end

        return pairs_to_process if pairs_to_process == 0
      end
      pairs_to_process
    end
    private :check_property_p

    def check_resource_x(pairs_to_process,
                         total_pairs,
                         percentage,
                         mgl)
```

```
        @resources.each do |x|
          pairs_to_process, lock_request =
            check(percentage,
                  pairs_to_process,
                  total_pairs,
                  lock_manager.resource,
                  nil,
                  true,
                  mgl,
                  :resource => x) do |pair|
                    pair.resource == x
                  end

          return pairs_to_process if pairs_to_process == 0
        end
        pairs_to_process
      end
      private :check_resource_x


      def check_property_p_of_resource_x(pairs_to_process,
                                         total_pairs,
                                         percentage,
                                         mgl)
        @requests.each do |request|
          request.items.each do |item|

            next if item.lock_requests_established?

            pairs_to_process, lock_request =
              check(percentage,
                    pairs_to_process,
                    total_pairs,
                    lock_manager.property_of_resource,
                    nil,
                    true,
                    mgl,
                    item) do |pair| #item no lugar de uris
                      true
                    end


            if lock_request.confirmed?
              inv_p = @simulation.inv_property item.pair.property
              pairs_to_process = lock_inv_property(inv_p,
                                                   pairs_to_process,
                                                   total_pairs,
                                                   mgl,
                                                   lock_request)
            end

            return pairs_to_process if pairs_to_process == 0
          end
        end

        if pairs_to_process > 0
          @requests.each do |request|
            request.items.each do |item|
               if not item.lock_requests_established?
                 puts "Transaction #{@id} - pair: #{item}"
                 puts "Lock Requests do item: \n"
                 item.lock_requests.each do |lock_request|
                   puts lock_request.to_s
                 end
               end
            end
          end
          msg = "There are #{pairs_to_process} request items "
          msg += "without lock requests established."
          puts "Error: #{msg}"
          raise RuntimeError.new(msg)
        end

      end
```

```
      private :check_property_p_of_resource_x

      def check(percentage,
                pairs_to_process,
                total_pairs,
                granule,
                lock_mode,
                upgrade_lock_mode,
                mgl,
                uris={},
                &block)

        if granule == lock_manager.property_of_resource
            request_item = uris #uris na verdade eh o request_item, neste caso
            uris = {:property => request_item.pair.property,
                    :resource => request_item.pair.resource}

        end

        data_item = mount_data_item(granule, uris)
        lock_request = @lock_requests[data_item]
        unless lock_request
          lock_request = Synth::Simulation::LockRequest.new(self,
                                                            lock_mode,
                                                            granule,
                                                            mgl,
                                                            uris)
        end

        total_selected = 0
        total_lock_requests_established = 0

        if granule != lock_manager.property_of_resource
          @requests.each do |request|
            x,y = request.num_items(lock_request, upgrade_lock_mode, &block)
            total_selected += x
            total_lock_requests_established += y
          end
        else
          x,y = Synth::Simulation::Request.send(:add_lock_request_to_item,
                                                request_item,
                                                lock_request,
                                                upgrade_lock_mode,
                                                &block)
          total_selected += x
          total_lock_requests_established += y
        end

        if total_selected > 0 && total_selected.to_f/total_pairs >= percentage
          lock_request.confirm
          @lock_requests[data_item] = lock_request
          pairs_to_process -= total_lock_requests_established
        end

        return pairs_to_process, lock_request
      end
      private :check

      #0
      def total_pairs_graph(num_resources, num_properties)
        num_resources * num_properties
      end
      private :total_pairs_graph

      #1
      def total_pairs_property_p(num_resources, num_properties)
        num_resources
      end
      private :total_pairs_property_p

      #2
      def total_pairs_resource_x(num_resources, num_properties)
        num_properties
      end
```

```
private :total_pairs_resource_x

#3
def total_pairs_property_p_of_resource_x(num_resources, num_properties)
  1
end
private :total_pairs_property_p_of_resource_x

class GranuleUpgrade

  attr_reader :total_pairs_method, :check_method

  def initialize(total_pairs_method, check_method)
    @total_pairs_method = total_pairs_method
    @check_method = check_method
  end

  def compare(other, transaction)

    self_total_pairs =
      @total_pairs_method.bind(transaction).call(
        transaction.simulation.num_resources,
        transaction.simulation.num_properties)
    other_total_pairs =
      other.total_pairs_method.bind(transaction).call(
        transaction.simulation.num_resources,
        transaction.simulation.num_properties)
    self_total_pairs <=> other_total_pairs
  end

  def check(transaction, pairs_to_process, percentage, mgl)

    total_pairs = @total_pairs_method.bind(transaction).call(
        transaction.simulation.num_resources,
        transaction.simulation.num_properties)
    @check_method.bind(transaction).call(pairs_to_process,
                                         total_pairs,
                                         percentage,
                                         mgl)
  end

end

def self.granule_upgrade_dispatcher(lock_manager)
  unless @granule_upgrade_dispatcher
    @granule_upgrade_dispatcher =
    {
      lock_manager.graph => GranuleUpgrade.new(
                  instance_method(:total_pairs_graph),
                  instance_method(:check_graph)),
      lock_manager.property => GranuleUpgrade.new(
                  instance_method(:total_pairs_property_p),
                  instance_method(:check_property_p)),
      lock_manager.resource => GranuleUpgrade.new(
                  instance_method(:total_pairs_resource_x),
                  instance_method(:check_resource_x)),
      lock_manager.property_of_resource => GranuleUpgrade.new(
                  instance_method(:total_pairs_property_p_of_resource_x),
                  instance_method(:check_property_p_of_resource_x))
    }
  end
  @granule_upgrade_dispatcher
end
private_class_method :granule_upgrade_dispatcher

def acquire_locks()

  @lock_time = 0
  @lock_requests.each do |lock_request|

    before = Time.new
    ok = lock_request.execute
    after = Time.new
```

```
          @lock_time += (after - before)

          unless ok
            return false
          end
        end
        @lock_time = @lock_time.to_f/num_locks
        true
      end
      private :acquire_locks

      def unlock_all
        @unlock_all_time = 0
        before = Time.new
        lock_manager.unlock_all @id
        after = Time.new
        @unlock_all_time = (after - before)
      end
      private :unlock_all

      def abort()
        unlock_all
        @num_aborts += 1
        @status = :aborted
      end

      def commit()
        unlock_all
        @status = :committed
      end

      def start()
        @status = :started
        @start_time = Time.new

        @simulation.before_try_acquire_locks #evitar starvation
        ok = acquire_locks
        @simulation.after_try_acquire_locks #evitar starvation

        unless ok
          abort
          @end_time = Time.new
          return false
        end

        count = @requests.length
        user_thinking_time = @simulation.user_thinking_time
        @requests.each do |request|
          request.execute
          count -= 1
          if user_thinking_time > 0 && count > 0
            before = Time.new
            sleep(user_thinking_time)
            after = Time.new
            @total_user_thinking_time += (after - before)
          end
        end
        commit
        @end_time = Time.new
        true
      end

      def reset()
        @status = :not_started
        @num_aborts = 0
        @lock_time= nil
        @unlock_all_time = nil
        @start_time = nil
        @end_time = nil
        @turnaround_start_time = nil
        @turnaround_end_time = nil
        @restart_time = 0
        @total_user_thinking_time = 0
        @total_restart_time = 0
```

```
      end
      private :reset

      def simulate()
         reset

         @turnaround_start_time = Time.new

         success = false
         until success || (@simulation.starvation?) do
           success = start
           @restart_time = @simulation.restart_time(@num_aborts) unless success
           if !success && @restart_time > 0
            before = Time.new
            sleep(@restart_time)
            after = Time.new
            @total_restart_time += (after - before)
           end

           @simulation.starvation = true if @num_aborts >= 1000000
           puts "Transaction #{@id} - Abortos: #{@num_aborts}" if (@num_aborts > 0) &&
((@num_aborts % 1000000) == 0)

         end

         @turnaround_end_time = Time.new

      end

      def turnaround_time
        if @simulation.starvation?
          10000000000
        else
          @turnaround_end_time - @turnaround_start_time
        end

      end

      def real_turnaround_time
        result = turnaround_time

        #remover restart_time
        result -= @total_restart_time
        #acrescentar o restart_time real
        result += (@total_restart_time*60) #cada segundo como 60s (1 min)

        #remover user thinking time
        result -= @total_user_thinking_time
        #acrescentar o user thinking time real
        result += (@total_user_thinking_time*60) #cada segundo como 60s (1 min)

        result
      end

      def response_time()
        @end_time - @start_time
      end

      def real_response_time()
        result = response_time

        #remover user thinking time
        result -= @total_user_thinking_time
        #acrescentar o user thinking time real
        result += (@total_user_thinking_time*60) #cada segundo como 60s (1 min)

        result
      end

      def to_s
        result = "\nTransaction #{@id} - #{@status}\n"
        result += "\nRequests: "
        @requests.each {|request| result += "\n#{request}"}
        result += "\nLock Requests: "
```

```
        @lock_requests.each {|request| result += "\n#{request}"} if @lock_requests
        result += "\n"
        result
      end

    end
  end
end
```

request.rb

```
require 'request_item'

module Synth
  module Simulation
    class Request

      attr_reader :items,
                  :properties,
                  :resources,
                  :transaction


      def initialize(transaction, pair_indexes)

        @transaction = transaction
        delta = transaction.simulation.max_pairs_per_request -
                transaction.simulation.min_pairs_per_request
        num_pairs = transaction.simulation.min_pairs_per_request +
                    rand(delta+1)
        @items = []

        @properties = []
        @resources = []
        1.upto(num_pairs) do |i|
          pair_index = pair_indexes.delete_at(rand(pair_indexes.length))
          pair = transaction.simulation.pair(pair_index)
          @items << Synth::Simulation::RequestItem.new(self, pair)
          @properties << pair.property
          @resources << pair.resource
        end

        @properties = @properties.uniq
        @resources = @resources.uniq
      end

      def lock_manager
        @transaction.simulation.lock_manager
      end
      private :lock_manager


      def num_pairs()
        @items.length
      end

      def set_write_lock_percentage(percentage,
                                    read_lock_modes_range,
                                    write_lock_modes_range)
        total = @items.length
        write_lock_count = total * percentage
        write_lock_count = write_lock_count.round
        read_lock_count = total - write_lock_count
        @items.each do |item|
          if write_lock_count > 0 && read_lock_count > 0
            kind = rand(2)
          elsif read_lock_count > 0
            kind = 0
          else
            kind = 1
          end

          if kind == 0 #read
            delta = read_lock_modes_range.end - read_lock_modes_range.first
            lock_mode = read_lock_modes_range.first + rand(delta+1)
```

```
            read_lock_count -= 1
          else #write
            delta = write_lock_modes_range.end - write_lock_modes_range.first
            lock_mode = write_lock_modes_range.first + rand(delta+1)
            write_lock_count -= 1
          end
          item.lock_mode = lock_mode
        end
      end

      def set_read_write_lock_modes(read_lock_mode, write_lock_mode)
        @items.each do |item|
          raise RuntimeError, "lock mode undefined" unless item.lock_mode
          if lock_manager.write_lock_mode?(item.lock_mode)
            item.lock_mode = write_lock_mode
          else
            item.lock_mode = read_lock_mode
          end
        end
      end

      def self.add_lock_request_to_item(item, lock_request, upgrade_lock_mode)
         total_selected = 0
         total_will_have_lock_requests_established = 0
         unless item.lock_requests_established?
            selected = yield item.pair
            if selected
               item.add_lock_request(lock_request, upgrade_lock_mode)
               total_selected = 1
               if item.will_lock_requests_be_established? lock_request
                 total_will_have_lock_requests_established = 1
               end
            end
         end
         return total_selected, total_will_have_lock_requests_established
      end
      private_class_method :add_lock_request_to_item

      def num_items(lock_request, upgrade_lock_mode, &block)
        total_selected = 0
        total_will_have_lock_requests_established = 0
        @items.each do |item|

           x,y = self.class.send :add_lock_request_to_item,
                                 item,
                                 lock_request,
                                 upgrade_lock_mode,
                                 &block
           total_selected += x
           total_will_have_lock_requests_established += y
        end
        return total_selected, total_will_have_lock_requests_established
      end

      def reset_lock_requests()
        @items.each {|item| item.reset_lock_requests}
      end

      def execute()
        @items.each do |item|
             #acessar um arquivo qualquer para simular tempo de I/O
             x = IO.read("Duvida0.rb", 30, 1000)
       end
      end

      def to_s
        result = "Request (items):\n "
        @items.each {|item| result += "#{item} "}
        result
      end

    end
  end
end
```

request_item.rb

```ruby
module Synth
  module Simulation
    class RequestItem
      attr_reader :request, :pair, :lock_requests
      attr_accessor :lock_mode

      def initialize(request, pair)
        @request = request
        @pair = pair
        @lock_mode = nil
        @lock_requests = []
      end

      def lock_manager()
        @request.transaction.simulation.lock_manager
      end
      private :lock_manager

      def handle_lock_requests_established?(lock_requests)
        lock_of_property = false
        lock_of_resource = false
        lock_requests.each do |lock_request|
          next unless lock_request.cover_lock_mode?(@lock_mode)
          unless lock_of_property
            lock_of_property = lock_request.property?(@pair)
          end
          unless lock_of_resource
            lock_of_resource = lock_request.resource?(@pair)
          end
          if lock_manager.write_lock_mode?(@lock_mode)
            return true if (lock_of_property && lock_of_resource)
          else
            return true if (lock_of_property || lock_of_resource)
          end
        end
        return false
      end
      private :handle_lock_requests_established?

      def confirmed_lock_requests()
        @lock_requests.select do
          |lock_request| lock_request.confirmed?
        end
      end

      def lock_requests_established?()
        handle_lock_requests_established?(confirmed_lock_requests)
      end

      def will_lock_requests_be_established?(lock_request)
        lock_requests = confirmed_lock_requests
        lock_requests << lock_request
        handle_lock_requests_established?(lock_requests)
      end

      def add_lock_request(lock_request, upgrade_lock_mode)
        @lock_requests << lock_request
        lock_request.lock_mode = @lock_mode if upgrade_lock_mode
      end

      def reset_lock_requests
        @lock_requests = []
      end

      def to_s
        lock_mode = @lock_mode ?
          lock_manager.lock_mode_name(@lock_mode) : @lock_mode
        "[#{@pair}, #{lock_mode}]"
      end

    end
  end
```

```
end
```

lock_request.rb

```
module Synth
  module Simulation
    class LockRequest

      attr_reader :transaction, :granule, :lock_mode, :uris

      def initialize(transaction, lock_mode, granule, mgl, uris={})
        @transaction = transaction
        @granule = granule
        @mgl = mgl
        @lock_mode = lock_mode
        @uris = uris
        @confirmed = false
      end

      def mgl?
        @mgl
      end

      def lock_manager()
        @transaction.simulation.lock_manager
      end
      private :lock_manager

      def lock_mode=(lock_mode)
        @lock_mode =
        @lock_mode ? lock_manager.conversion(@lock_mode, lock_mode) : lock_mode
      end

      def confirm()
        @confirmed = true
      end

      def confirmed?()
        @confirmed
      end

      def cover_lock_mode?(lock_mode)
        return false unless @lock_mode
        @lock_mode == lock_manager.conversion(@lock_mode, lock_mode)
      end

      def property?(pair)
        return true if granule == lock_manager.graph
        result = (uris[:property] == pair.property)
        if !mgl? && granule != lock_manager.property_of_resource
          result ||= resource?(pair)
        end
        result
      end

      def resource?(pair)
        return true if granule == lock_manager.graph
        result = (uris[:resource] == pair.resource)
        if !mgl? && granule != lock_manager.property_of_resource
          result ||= property?(pair)
        end
        result
      end

      def data_item
        lock_manager.send :mount_data_item, granule, uris
      end

      def <=>(other)
        data_item <=> other.data_item
      end

      def execute()
        begin
          lock_manager.lock @transaction.id, @granule, @lock_mode, @uris
```

```
        rescue => ex
          puts "#{ex.class}: #{ex.message}"
          print ex.backtrace.join("\n")
        end
      end

      def to_s
        lock_mode = @lock_mode ?
          lock_manager.lock_mode_name(@lock_mode) : @lock_mode
        result = "[#{@granule}, #{@uris}, #{lock_mode}, "
        result += @confirmed ? "confirmed]" : "unconfirmed]"
        result
      end

    end
  end
end
```

results.rb

```
module Synth
  module Simulation
    class Results

      def initialize(title, *measurements)
        @measurements = measurements
        @files = {}
        now = Time.new
        @results_dir = "#{title}_at"
        @results_dir += "_#{now.year}_#{now.mon}_#{now.day}"
        @results_dir += "_#{now.hour}_#{now.min}_#{now.sec}"

        Dir.mkdir(@results_dir)
      end

      def write_percentage=(write_percentage)
        close if @dir

        Dir.chdir(@results_dir)
        @dir = "cenario_#{write_percentage*100}%_write"
        Dir.mkdir(@dir)
        Dir.chdir("..")
        @dir = @results_dir + "/" + @dir
        open
      end

      def variable=(variable)
        nl = "\n"
        @files.values.each {|file| file << nl}
      end

      def arrival_rate=(arrival_rate)
        space = " "
        @files.values.each {|file| file << space}
      end

      def open()
        dir = @dir + "/"
        @measurements.each do |measurement|
          @files[measurement] = File.open(dir+"#{measurement}.txt", "w")
        end
      end
      private :open

      def values=(hash_values)
        hash_values.keys.each do |measurement|
          @files[measurement] << ("%.5f" % hash_values[measurement])
        end
      end

      def close()
        @files.values.each {|file| file.close}
      end

    end
```

```
  end
end
```

simulation_suite.rb

```
require 'rdf_lock_manager'
require 'synth/simulation/simulation'

module Synth
  module Simulation
    class SimulationSuite
      def initialize(num_resources,
                     num_properties,
                     num_transactions,
                     min_requests_per_transaction,
                     max_requests_per_transaction,
                     min_pairs_per_request,
                     max_pairs_per_request,
                     min_user_thinking_time,
                     max_user_thinking_time,
                     min_restart_time,
                     max_restart_time,
                     io_time,
                     inv_properties_percentage,
                     write_lock_percentages,
                     arrival_rates,
                     avoid_starvation)

         @simulation = Synth::Simulation::Simulation.new(
                   Synth::Transaction::RdfLockManager.instance,
                   num_resources,
                   num_properties,
                   num_transactions,
                   min_requests_per_transaction,
                   max_requests_per_transaction,
                   min_pairs_per_request,
                   max_pairs_per_request,
                   min_user_thinking_time,
                   max_user_thinking_time,
                   min_restart_time,
                   max_restart_time,
                   io_time,
                   inv_properties_percentage,
                   avoid_starvation)

         @write_lock_percentages = write_lock_percentages
         @arrival_rates = arrival_rates

      end

      def write_lock_percentages
        @simulation.write_lock_percentages ? nil : @write_lock_percentages
      end
      private :write_lock_percentages

      def simulate_granule()
          title = 'Granule'
          @simulation.execute(title,
                    [Synth::Transaction::RdfLockManager.graph,
                     Synth::Transaction::RdfLockManager.property,
                     Synth::Transaction::RdfLockManager.resource,
                     Synth::Transaction::RdfLockManager.property_of_resource],
                    write_lock_percentages,
                    @arrival_rates) do |t, granule|
                        t.fix_granule(granule)
                    end
      end

      def simulate_granule_upgrade(upgrade_percentages)
          @simulation.execute('Upgrade',
                    upgrade_percentages,
                    write_lock_percentages,
                    @arrival_rates) do |t, upgrade_percentage|
                        t.granule_upgrade_percentage = upgrade_percentage
                    end
```

```
      end

      def simulate_lock_mode()
          incompatible_lock_modes = [Synth::Transaction::RdfLockManager.riR,
                                     Synth::Transaction::RdfLockManager.riW]

          compatible_lock_modes   = [Synth::Transaction::RdfLockManager.rR,
                                     Synth::Transaction::RdfLockManager.iW]

          @simulation.execute('Lock_mode',
                    [incompatible_lock_modes, compatible_lock_modes],
                    write_lock_percentages,
                    @arrival_rates) do |t, lock_modes|
                      t.set_read_write_lock_modes(lock_modes[0], lock_modes[1])
                    end
      end

    end
  end
end
```

simulation_test.rb (um dos testes)

```
#!/usr/bin/env ruby
$LOAD_PATH << '../lib'
$LOAD_PATH << '../lib/synth'
$LOAD_PATH << '../lib/synth/transaction'
$LOAD_PATH << '../lib/synth/simulation'

require 'simulation_suite'
require 'rdf_lock_manager'

num_resources = 300
num_properties = 100
num_transactions = 1000
min_requests_per_trans = 1
max_requests_per_trans = 1
min_pairs_per_request = 30
max_pairs_per_request = 3000
min_user_thinking_time = 0
max_user_thinking_time = 0
min_restart_time = 0
max_restart_time = 0
io_time = 0  #milisegundos
inv_properties_percentage = 0.0/100.0
avoid_starvation = true

percent_write_20  = [20.0/100.0]
percent_write_80  = [80.0/100.0]
percent_write_50  = [50.0/100.0]


arrival_rates = [10, 100, 1000000000]

granule_upgrade_percentages = [0.5/100.0,
                               1.0/100.0,
                               5.0/100.0,
                               10.0/100.0,
                               15.0/100.0,
                               20.0/100.0,
                               25.0/100.0,
                               30.0/100.0,
                               50.0/100.0]


#Cenário 80% operações de escrita
s = Synth::Simulation::SimulationSuite.new(num_resources,
                                           num_properties,
                                           num_transactions,
                                           min_requests_per_trans,
                                           max_requests_per_trans,
                                           min_pairs_per_request,
                                           max_pairs_per_request,
                                           min_user_thinking_time,
                                           max_user_thinking_time,
```

```
                                        min_restart_time,
                                        max_restart_time,
                                        io_time,
                                        inv_properties_percentage,
                                        percent_write_80,
                                        arrival_rates,
                                        avoid_starvation)

s.simulate_granule_upgrade granule_upgrade_percentages

s.simulate_granule #NO MGL

#tirar o comentário abaixo para simular os tipos de lock
#s.simulate_lock_mode


#Cenário 20% operações de escrita
s = Synth::Simulation::SimulationSuite.new(num_resources,
                                        num_properties,
                                        num_transactions,
                                        min_requests_per_trans,
                                        max_requests_per_trans,
                                        min_pairs_per_request,
                                        max_pairs_per_request,
                                        min_user_thinking_time,
                                        max_user_thinking_time,
                                        min_restart_time,
                                        max_restart_time,
                                        io_time,
                                        inv_properties_percentage,
                                        percent_write_20,
                                        arrival_rates,
                                        avoid_starvation)

s.simulate_granule_upgrade granule_upgrade_percentages

s.simulate_granule #NO MGL

#tirar o comentário abaixo para simular os tipos de lock
#s.simulate_lock_mode
```