KLASIFIKASI SECARA EFISIEN PADA DATABASE MULTI RELASI DENGAN ALGORITMA CROSSMINE

Sarwosri, Darlis Herumurti, Indri Sulistyowati
Jurusan Teknik Informatika, Fakultas Teknologi Informasi,
Institut Teknologi Sepuluh Nopember
e-mail: sri@its-sby.edu, darlis@its-sby.edu

Abstract

Multi-relation classifications can be widely used in many disciplines, such as financial decision making, medical research, and geographical applications, and information stored in multiple relations needs to be used in decision making. Crossmine is an efficient and scalable approach for multi-relation classification. Crossmine algoritm has three step, first is find-rules, the rule has been gotten from find a rule process than remove all positif tuples satisfying rule while there are more than ten percent positif tuple left. The second is find a rule, this step has input from the result of find best predicate process, that is the complex predicate with most foilgain. If foilgain value is more than mingain, the predicate is added with rule, and max rule length less than six. Third is find best predicate, in this step we find the best predicate with definition, if the foilgain value more than the max gain value, the predicate will be saved and the bigger gain value will replace the last gain value for next comperative process. In other side, the accuracy is computed from each rule that produce in find rules process. The test for this application use the sum tuple of 200, 500, 1000, 5000 for measuring the level of accuracy from rule which is produced by crossmine algoritm.

Keyword: *accuracy, crossmine, negative tuple, positive tuple, predicate, rule*

Abstrak

Klasifikasi multi-relasi dapat digunakan secara luas dalam banyak disiplin, seperti pembuatan keputusan keuangan, penelitian medis, dan aplikasi geografi, informasi yang disimpan dalam multi-relasi diperlukan dalam pembuatan keputusan. CrossMine, suatu pendekatan yang efisien dan dapat diskala untuk klasifikasi multi-relasi. Algoritma crossmine memiliki tiga langkah, pertama dimulai dari pencarian sekumpulan rule, rule didapatkan dari proses pencarian sebuah rule yang kemudian diambil tuple positifnya sampai tersisa lebih dari sepuluh persen. Langkah kedua adalah pencarian sebuah rule, pada langkah ini memiliki input dari hasil proses pencarian predikat terbaik, yaitu predikat komplek dengan nilai foilgain, nilai foil gain dibandingkan dengan min gain jika hasilnya lebih besar min gain maka predikat tersebut ditambahkan dengan rule, dan maksimal kedalaman relasi adalah kurang dari enam. Langkah ketiga adalah find best pred, pada langkah ini dilakukan pencarian predikat terbaik dengan ketentuan jika nilai foilgain lebih besar dari nilai max gain maka predikat akan disimpan dan nilai gain yang lebih besar akan menggantikan nilai gain sebelumnya untuk proses perbandingan selanjutnya. Sedangkan untuk accuracy dihitung dari masing masing rule yang dihasilkan pada proses pencarian sekumpulan rule. Uji coba aplikasi ini menggunakan jumlah tuple 200, 500, 1000, 5000 untuk mengukur tingkat efisiensi dan akurasi dari rule yang dihasilkan algoritma crossmine.

Kata kunci *: crossmine, predikat, akurasi, rule, tuple positif, tuple negative*

1. PENDAHULUAN

Kebanyakan data terstruktur saat ini disimpan dalam relasi *database*. *Database* ini terdiri dari banyak relasi yang berhubungan secara konseptual melalui *link-link entity-relation* dalam desain skema-skema relasi database. Klasifikasi multi-relasi dapat digunakan secara luas dalam banyak disiplin, seperti pembuatan keputusan keuangan, penelitian medis, dan

aplikasi geografi, dan informasi yang disimpan dalam multi-relasi diperlukan dalam pembuatan keputusan [1].

Multi-relasi dalam suatu database biasanya dihubungkan melalui link-link semantik seperti *link-link entity-relation* dari sebuah model ER yang digunakan dalam desain *database*. Data yang disimpan dalam relasi yang sama sering memiliki hubungan semantik yang lebih dekat daripada yang dapat dicapai melalui *link-link* yang jauh [2].

Kebanyakan pendekatan klasifikasi hanya bekerja pada relasi data "*flat*" tunggal. Biasanya sulit mengubah multi-relasi menjadi sebuah relasi flat tunggal dan konversi demikian dapat menyebabkan pembangkitan sebuah relasi universal yang besar serta dapat mengakibatkan kehilangan beberapa informasi semantik penting yang dibawa oleh *link-link* semantik dalam desain *database* [3].

Guna menyelesaikan masalah klasifikasi multi-relasi dalam relasi *database*, ada dua tantangan utama, yaitu: (1) efisiensi dan skalabilitas dan (2) ketelitian klasifikasi. Saat membuat klasifikasi untuk sebuah database dengan multi-relasi, ruang pencarian biasanya sangat besar, dan tak dapat digunakan untuk melakukan pencarian yang mendalam [4].

Di lain pihak, link-link semantik biasanya menjadi sangat lemah setelah melewati sebuah rantai panjang link-link [5]. Oleh karenanya, sebuah klasifikasi multi-relasi perlu menangani masalah efisiensi maupun ketelitian. Sampai sekarang, masih ada kekurangan metode-metode klasifikasi yang kurang teliti, efisien, dan multi-relasi yang dapat diskala untuk menangani *database-database* besar dengan skema-skema yang kompleks.

2. METODOLOGI PENELITIAN

CrossMine adalah suatu pendekatan yang efisien dan dapat diskala untuk klasifikasi multi-relasi. Beberapa metode baru dikembangkan dalam *CrossMine*, termasuk sebaran *tuple ID*, yang melakukan hubungan virtual yang mempertahankan-semantik untuk mencapai efisiensi tinggi pada *database* dengan skema-skema kompleks. Ide dasar dari *CrossMine* adalah mewujudkan *join relation* untuk menemukan predikat yang baik. Ada dua algoritma berbeda dari *CrossMine*: *CrossMine-tree* dan *CrossMine-rule*. *CrosMine-tree* adalah sebuah *decision-tree based classifier*, yang secara *recursive* menyeleksi atribut terbaik dan membagi *tuple* target menjadi partisi-partisi. *CrossMine-rule* adalah *rule-based classifier*, yang secara berulang membuat *predictive rule* dan kemudian memfokuskan pada sisa *tuple target* [3-4].

2.1. Sebaran *Tuple ID*

Sebaran *tuple id* adalah sebuah metode efisien dan fleksibel untuk join suatu relasi. Andaikan *primary key* relasi target adalah suatu atribut integer, yang menggambarkan ID dari setiap *tuple* target. Teorema berikut [3] menunjukkan kebenaran penyebaran *ID tuple*.

Teorema 1, Misalkan relasi R1 dan R2, yang dapat di-join-kan dengan atribut R1.A dan R2.A. R1 adalah target relasi, semua tuple R1 yang memenuhi *current rule*. Jika ID pada R1 disebarkan ke R2, kemudian gunakan penyebaran ID-ID ini, *foil gain* dari setiap *predicate* di R2 dapat dihitung.

Teorema 2, Misalkan dua relasi R2 dan R3 yang dapat dihubungkan dengan atribut R2.A dan R3.A. semua *tuple* di R2 yang memenuhi *current rule*. Untuk masing-masing tuple t di R2, ada suatu kumpulan sebagai gambaran dari ID-ID target relasi yang dapat join dengan t (gunakan jalur penghubung khusus pada *current rule*). Dengan menyebarkan ID-ID dari R2 hingga R3 melalui hubungan R2.A=R3.A ,untuk setiap *tuple t*' di R3 bisa mendapatkan kumpulan dari gambaran *IDs tuple* target relasi yang dapat join dengan t (gunakan *join-path* pada *current rule*, dan hubungan R2.A=R3.A).

2.2. Predikat

Predikat adalah sebuah pembatas pada atribut tertentu dalam relasi tertentu. *CrossMine* menggunakan *tuple ID propagation* untuk men-*transfer* informasi diantara relasi yang berbeda untuk membentuk classifiers. Untuk mengumpulkan informasi yang lebih baik dari *ID propagation*, digunakan komplek predikat sebagai bagian dari *clasifiers*.

1. *Prop-path*: mengindikasikan bagaimana menyebarkan IDs, Misalnya, "*Loan.account_id → Account.account_id*" menunjukkan menyebarkan IDs dari relasi *Loan* ke relasi *Account* menggunakan hubungan "*Loan.account_id = Account.account_id*"
2. *Constrains* (batasan) menunjukkan pembatas pada relasi di mana IDs disebarkan. Misalnya, "*Account.frequency = monthly*" menunjukkan bahwa *tuple-tuple* dalam relasi Account harus memiliki nilai "*monthly*" pada atribut *frequency*.
   Constrain atau pembatas, dibagi menjadi 2:
   a. Jika value berisi string
      atribut *Account.frequency* , setiap nilai f dapat dibuat predikat, yaitu "*frequency=f*".
   b. Jika value berisi number
      atribut account.date, setiap nilai d, dapat dibuat suatu predikat, yaitu "*date=d*", "*date<=d", dan "date>=d*".
2. Gabungan dari *Prop-path* dan *Constrain*,
   "*Loan(+):- [Loan.account_id → Account.account_id, Account.frequency = monthly]*".

2.3. *Rules*

CrossMine Rule mengarahkan pada aturan pembentukan yang dapat membedakan sampel positif dan negatif. Tiap *rule* adalan daftar dari predikat, bergabung dengan *class label*. *Tuple target* memenuhi sebuah rule jika dan hanya jika memenuhi setiap predikat dari *rule*. Perhatikan *rule* berikut:

"*r=Loan(+) : -[Loan.account_id→Account.account_id, Account.frequency=montly]*"

sebuah tuple t dalam Loan memenuhi rule r jika dan hanya jika beberapa tuple dalam account yang dapat bergabung dengan t memiliki nilai "monthly" dalam atribut frekuensi. Rule terdiri dari:

1. *Rule* sekarang, contoh:
   "*Loan(L,+):- Loan(L,A,?,?,?,?)*"
   jika di-*convert* dalam *sql query*:
   "select * from loan"
2. *Rule* = *rule* sekarang (r) + predikat (p), contoh:
   jika *p="Account(A,?,monthly,?)*"
   maka *r+p = "Loan(L,+):- Loan(L,A,?,?,?,?),account(A,?,monthly,?)*"
   jika di-*convert* dalam *sql query*:
   "*select loan.*, account.* from loan, account where loan.account_id = account.account_id and account.frequency='monthly*'"

2.4. *Foilgain*

Foilgain digunakan untuk mengevaluasi predikat dalam relasi dengan menggunakan sebaran ID. Definisi dasar dari foilgain [3] adalah: untuk suatu rule r, gunakan P (r) dan N (r) untuk menunjukkan banyaknya contoh positif dan negatif yang memenuhi r. Andaikan rule sekarang adalah r, gunakan r + p untuk menunjukkan rule yang dibuat dengan menambahkan predikat p ke r.

Berdasarkan intuisi *foil_gain(p)* menunjukkan jumlah total bit-bit yang disimpan dalam menggambarkan contoh-contoh positif dengan menambahkan p ke *rule* sekarang. Dengan foilgain orang dapat mengukur kebaikan setiap predikat target relasi Loan. Untuk sebuah predikat "*Account(A,?,momthly,?)*", andaikan *rule r = Loan(L,+) :- Loan(L,A,?,?,?,?), Account(A,?,monthly,?)*. Dapat dikatakan sebuah *tuple t* dalam *Loan* memenuhi r jika dan hanya jika *tuple* apapun dalam *Account* yang dapat dihubungkan dengan t memiliki nilai "*monthly*" dalam atribut frekuensi.

Rumus foilgain untuk rule (r) adalah:

$$I(r) = -\log \frac{P(r)}{P(r)+N(r)} \quad (1)$$

$$I(r) = - \log \left(\frac{P(r)}{P(r)+N(r)} \right) / \log 2 \quad (2)$$

Rumus foilgain untuk rule (r +p) adalah:

$$I(r+p) = - \log \frac{P(r+p)}{P(r+p)+N(r+p)} \quad (3)$$

$$I(r+p) = - \log \left(\frac{P(r+p)}{P(r+p)+N(r+p)} \right) / \log 2 \quad (4)$$

Rumus foilgain adalah:

$$Foilgain\ (p) = P(r+p) \,.\, [I(r) - I(r+p)] \quad (5)$$

dimana: P(r) = jumlah tuple positif yang memenuhi rule
N(r) = jumlah tuple negatif yang memenuhi rule
P(r+p) = jumlah tuple positif yang memenuhi rule (r+p)
N(r+p) = jumlah tuple negatif yang memenuhi rule (r+p)
I(r) = nilai Foilgain untuk rule (r)
I(r+p) = nilai Foilgain untuk rule (r+p)

2.5. Accuracy

Untuk mencapai ketelitian tinggi (high accuracy) dalam klasifikasi multi-relasi, suatu algoritma harus dapat menemukan banyak predikat yang berguna dalam database, dan membuat rule yang baik dengan predikat tersebut.

Perhitungan accuracy adalah sebagai berikut misalkan r.sup+ dan r.sup- adalah banyaknya tuple positif dan negatif yang memenuhi suatu rule r. Misalkan r.bg+ dan r.bg- adalah banyaknya tuple positif dan negatif dari mana r dibuat. Ketelitian r dapat diperkirakan menggunakan metode dalam persamaan berikut :

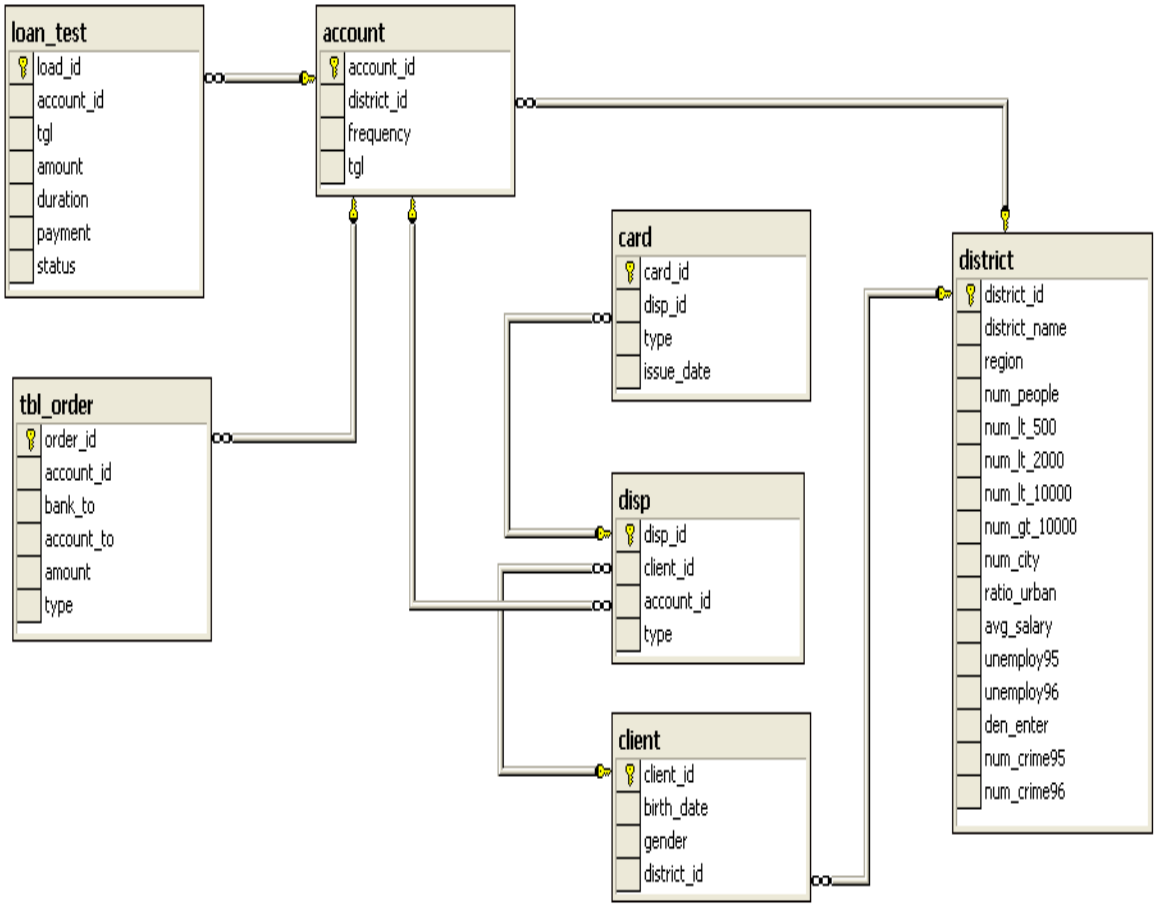
$$Accuracy(r) = (r.sup+ + 1)/(r.sup+ + r.sup- + c) \quad (6)$$

dimana: r.sup+ = jumlah tuple positif pada rule
r.sup- = jumlah tuple negatif pada rule
c = jumlah class label

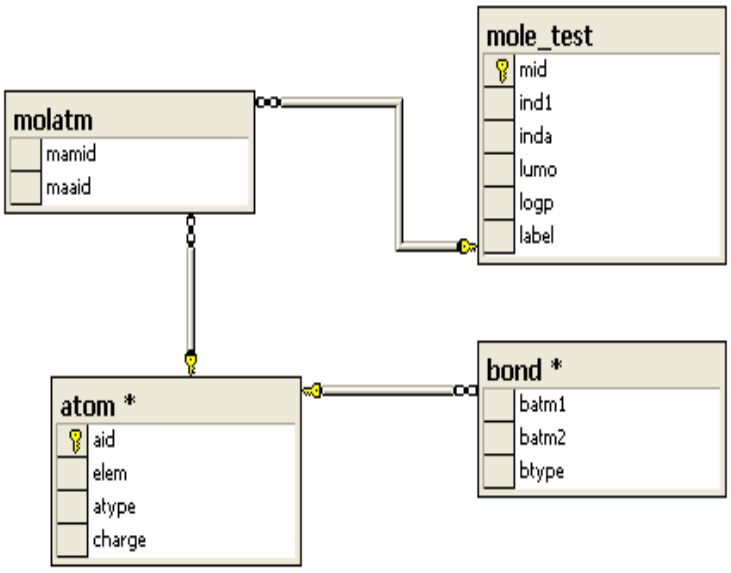
$$k^* = \arg \max_k y_k, \quad 1 \le k \le m \quad (7)$$

2.6. Perancangan Input

Relational database yang akan digunakan untuk uji coba dalam sistem ini ada dua yaitu finalcial database dan mutagenesis database. Financial database merupakan database keuangan yang menyimpan data tentang pinjaman, dan digambarkan pada skema ER pada Gambar 1. Sedangkan database mutagenesis adalah database yang menyimpan data tentang molekul atom, yang ditunjukkan pada Gambar 2.



Gambar 1. ER Diagram dari database financial



Gambar 2. ER Diagram dari database mutagenesis

2.5 Perancangan Proses

Perancangan proses sistem *crossmine* adalah pencarian sekumpulan *rule*, proses pencarian satu rule, proses pencarian predikat terbaik, proses perhitungan *accuracy*, dan arsitektur perangkat lunak *crossmine*.

2.5.1 Proses pencarian sekumpulan rule

Terdapat satu proses yaitu pembatasan *tuple* positif lebih dari 10%. *Rule* yang dihasilkan dari proses pencarian sebuah *rule* diambil *tuple* positifnya sampai 10% dari keseluruhan jumlah *tuple*. Jika *tuple* positifnya memenuhi 10% maka *rule* tersebut disimpan menjadi sekumpulan *rule*.

Input	sebuah relational database D dengan target relation Rt
Output	Sekumpulan rule untuk memprediksi class label pada tuple target

Algoritma :

```
Procedure:
            Rule set R = emptyset;
Do
            Rule r = Find-A-Rule();
      Tambahkan r ke R;
      Pindahkan semua tuple target positif
      yang sesuai dengan r;
While (tuple target positif tersisa lebih dari 10% )
Return R;
```

3. HASIL DAN PEMBAHASAN

Uji coba dilakukan untuk menguji jalannya program mulai dari proses input sampai proses output, selain itu juga digunakan untuk mengetahui kesesuaian aplikasi dengan fungsi-fungsi yang ingin buat.

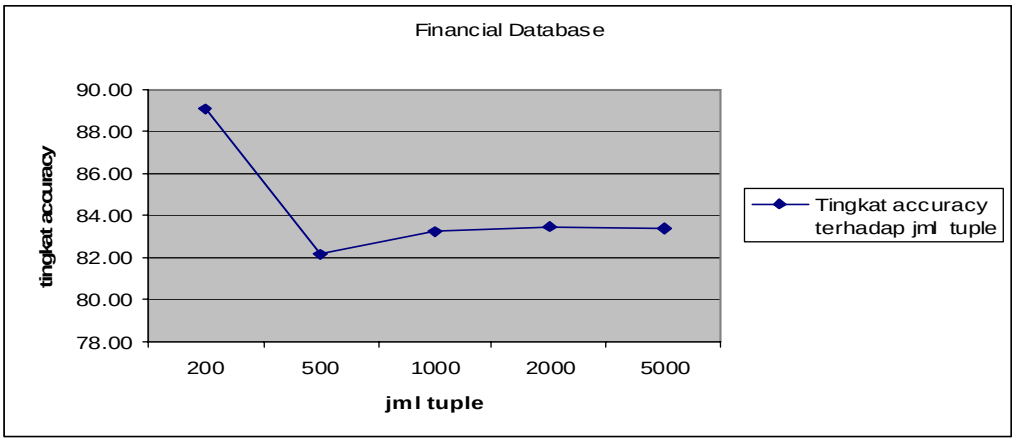
Skenario yang akan diujikan adalah sebagai berikut:

1. Ujicoba untuk menghasilkan *rule* terbaik dan tingkat *accuracy* per *rule* dengan menggunakan *financial database*
   a. Untuk jumlah *tuple* 200
   b. Untuk jumlah *tuple* 500
   c. Untuk jumlah *tuple* 1000
   d. Untuk jumlah *tuple* 2000
   e. Untuk jumlah *tuple* 5000
2. Ujicoba menghasilkan rule terbaik dan tingkat *accuracy* per *rule* dengan menggunakan mutagenesis database
   a. Untuk jumlah *tuple* 200
   b. Untuk jumlah *tuple* 500
   c. Untuk jumlah *tuple* 1000
   d. Untuk jumlah *tuple* 2000
   e. Untuk jumlah *tuple* 5000

Berikut ini adalah rata-rata tingkat *accuracy* tiap *rule* yang dihasilkan oleh sistem *crossmine* untuk *tuple* 200, 500, 1000, 2000, dan 5000 untuk *database financial*. Hasilnya ditunjukkan pada Tabel 1 dan Gambar 3.

Tabel 1. hasil accuracy per jumlah rule untuk database financial

Jml *tuple*	200	500	1000	2000	5000
Accuracy	89,07	82,16	83,21	83,45	83,41

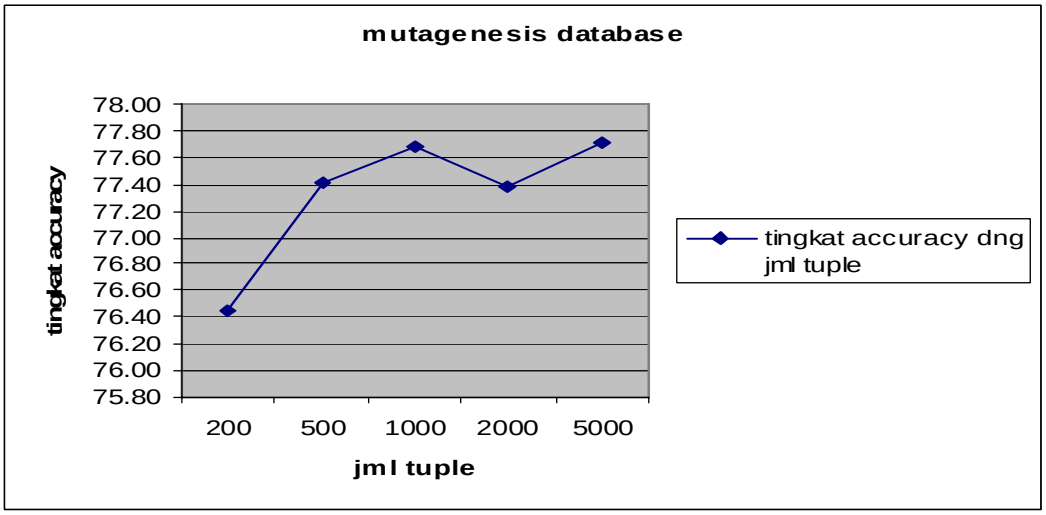


Gambar 3. Tingkat *accuracy financial database* dengan jumlah *tuple*

Berikut ini adalah rata-rata tingkat *accuracy* tiap *rule* yang dihasilkan oleh sistem *crossmine* untuk *tuple* 200, 500, 1000, 2000, dan 5000 untuk *database mutagenesis*. Hasilnya ditunjukkan pada Tabel 2 dan Gambar 4.

Tabel 2. Hasil accuracy per jumlah *rule* untuk *database mutagenesis*

Jml tuple	200	500	1000	2000	5000
Accuracy	76.45	77,41	77,69	77,39	77,71



Gambar 4. Perbandingan tingkat *accuracy mutagenesis database* dengan jumlah *tuple*

4. SIMPULAN

Simpulan yang dapat diambil dari penelitian ini adalah sebagai berikut ini:

1. *Rule* terbaik dalam *relational database* dapat dibangun dengan menerapkan langkah-langkah pada algoritma *crossmine* dan dapat dihitung tingkat *accuracy* pada tiap *rule*.
2. Naik turunnya tingkat *accuracy* baik pada *database financial* maupun *database mutagenesis* dipengaruhi oleh jumlah *tuple* positif dan *tuple* negatif yang ada pada tiap *rule*

3. Semakin besar jumlah *tuple* negatif pada sebuah *rule*, maka tingkat *accuracy*-nya semakin kecil.

DAFTAR PUSTAKA

[1] A. Baras, C. A. Galindo-Legaria, T. Grabs, B. Krishnaswamy, and S. Pal, "**Optimizing Similar Scalar Subqueries for XML Processing in Microsoft SQL Server**", IEEE 23rd International Conference on Data Engineering 2007 (ICDE 2007), 2007, pp. 1164-1173.
[2] H. Xue-Gang, H. Xie-Fei, W. De-Xing, Z. Dong-Yan, and H. Chun-Ling, "**A Classification Algorithm Based on Multi-Relation Domain Knowledge**", International Conference on Machine Learning and Cybernetics 2005 (ICMLC 2005), 2005, pp. 2067-2072.
[3] Y. Xiaoxin, H. Jiawei, Y. Jiong, and S. Y. Philip, "**Crossmine: Efficient Classification Across Multiple Database Relations**", 20th International Conference on Data Engineering, 2004., 2004, pp. 399-410.
[4] X. Yin, J. Han, J. Yang, and P. S. Yu, "**Efficient Classification Across Multiple Database Relations: A Crossmine Approach**," IEEE Transactions on Knowledge and Data Engineering, Vol. 18, 2006, pp. 770-783.
[5] Xiaoxin, Y., "**Financial Database and Mutagenesis Database**", http://www.ews.uiuc.edu/~xyin1/files/CrossMine_release.zip, 2004.