SNATI
Seminar Nasional
Aplikasi Teknologi Informasi 2004
Yogyakarta, 19 Juni 2004

Basis Data Temporal: Aplikasi dan Strategi Implementasinya

Sutrisno
Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Pelita Harapan
UPH Tower lt.9, Lippo Karawaci, Tangerang, 15811, Telp. 021-5460901, Fax: 021-5461082
e-mail: sutrisno@uph.edu, metrisno@yahoo.com

Abstract

Natures of most applications data evolving with time, and previous data state is still needed for various things besides current state, such as trend analysis, decision making, and others. Temporal data is a data inherently with time. Current data will be replaced with a new one. Database temporal is a bitemporal database. A database supports both historical and transaction database capabilities; it represented with valid-time, and transaction-time. Database temporal can be implemented on relational database model, or object-oriented database model. The low level implementation strategy of database temporal is proposed in this paper. As consequence, temporal database management system must be built from the scratch. A temporal database engine must be built. Prototype of temporal database engines have been built both on relational data model, and object-oriented data model.

Kata kunci: *data temporal, sistem basis data, basis data berorientasi obyek, basis data relasional, basis data spasio-temporal.*

1. Pendahuluan

Basis data menjadi salah satu elemen penting pada sebagian besar aplikasi berbasis komputer. Basis data digunakan untuk mendukung sistem operasi, contohnya Sistem Operasi Windows dengan Registry-nya. Sistem *fault-tolerant*, *networking* untuk *routing*, *real-time*, adalah beberapa contoh lain aplikasi yang memerlukan dukungan basis data.

Basis data pada dasarnya adalah koleksi data yang relevan yang diperlukan oleh suatu *enterprise* untuk mendukung aktifitasnya, Elmasri, et.al (2000). Pengorganisasian, dan pengelolaan koleksi data menjadi hal yang tidak dapat dihindarkan, karena volume data yang cenderung besar, dan nilai data (*state*) yang juga cenderung berubah seiring waktu. Berbagai teknik dan metoda dikembangkan agar data dapat dikelola secara efisien, dan mudah diekstraksi pada saat diperlukan.

Model data adalah suatu cara pandang mengenai data dan pengorganisasiannya khususnya pada tingkat abstraksi atau cara pandang *user*. Sebuah model data secara implisit memberikan implikasi aturan mengenai bagiamana data diorganisasikan, dikelola, dan teknologinya. Beberapa model data yang telah dikembangkan, diantaranya model: hirarki, jaringan, relasional, semantik, berorientasi obyek, relasional-obyek.

Teknologi pendukung yang ada saat itu, diantaranya adalah teknologi media penyimpanan (*storage*), memberikan andil pada pengembangan suatu model basis data, selain aplikasi itu sendiri. Kapasistas daya tampung media penyimpanan yang terbatas sehingga harus efisien dalam penggunaannya, yaitu hanya data yang perlu dan dianggap penting yang boleh disimpan. Aplikasi mengikuti aturan dan menyesuaikannya dengan teknologi yang ada, meski mungkin belum tentu seperti yang diharapkan.

2. Sistem Manajemen Basis Data

Sistem manajemen basis data (*database management system*, atau dbms) merupakan piranti lunak yang menjadi andalan utama untuk mengorganisasi dan mengelola basis data. Kerumitan penyimpanan pada media penyimpan fisik (*disk*), dan akses diambil alih oleh dbms. Sebuah piranti lunak sistem manajemen basis data (dbms) dikembangkan berdasarkan rancangan dan aturan untuk mendukung suatu model data tertentu.

Pengelolaan data pada aplikasi-aplikasi yang berdukungan sistem manajemen basis data tertentu, harus mengikuti aturan serta kaidah dari model data yang didukungnya. Pada tahapan rancangan konseptual maupun logikal, sebuah aplikasi harus mengikuti kaidah model data tertentu (misalnya relasional). Pada tahapan rancangan fisik basis data harus mengikuti aturan serta ketentuan yang ditetapkan oleh sebuah sistem manajemen basis data (dbms) tertentu (misalnya SQL Server, Oracle, atau DB2), Connolly, et.al (2003).

Sebuah sistem manajemen basis data berinteraksi dua arah, yaitu ke arah media fisik penyimpanan (*disk*), dan ke arah luar atau pengguna melalui aplikasi. Interaksi ke arah media fisik, berhubungan dengan tugas menyimpan data dari memori (*buffer*) ke media penyimpanan (*disk*). Interaksi ke luar yaitu berkenaan dengan membaca data dari *disk* dan memuatkannya ke memori (*buffer*) untuk selanjutnya diakses oleh program aplikasi.

Teknik serta organisasi penyimpanan pada media fisik tidak harus persis sama seperti apa yang dibayangkan atau dipersepsikan oleh pengguna (*user*) pada tingkat abstraksi atau aplikasi. Ekstraksi data harus direpresentasikan seperti pada saat penyimpanan, artinya rekonstruksi harus menjamin konsistensi dengan cara pandang data pada tingkat pemakai, Sutrisno (1997).

Fasilitas bahasa untuk pendefinisian data, dan bahasa untuk manipulasi data, serta fasilitas bahasa untuk pengelolaan kendali lingkungan (*concurrency*, *recovery*, *integrity*, dan *security*) disediakan oleh sistem manajemen basis data (dbms), yang dikenal dengan *database language*, Kim (1992). Fasilitas bahasa tersebut harus mampu mengakomodasi baik untuk penyimpanan maupun ekstraksi data. *Structured Query Language* (SQL) adalah salah satu contoh bahasa basis data yang dikenal paling baik (*powerful*) pada basis data model relasional.

Pada tataran fisik (*physical level*) berkenaan dengan manajemen data dikenal dengan istilah mesin manajemen data (*data mangement engine*). Organisasi data, metoda akses, manajemen memori data (*buffer*) adalah beberapa tugas yang dikerjakan oleh *data management engine*.

3. Basis Data Relasional

Basis data relasional adalah sebuah model data yang paling popular digunakan, khususnya pada aplikasi-aplikasi bisnis. Basis data relasional didukung oleh piranti lunak *relational database management system*. Piranti lunak ini mendukung model data relasional, karena dibangun mengikuti kaidah model data tersebut.

Model data relasional merupakan sebuah model data yang paling mantap (*solid*) hingga saat ini. Mantapnya model ini karena didukung oleh landasan teori yang jelas, dan mudah dipahami yaitu teori himpunan, dan aljabar relasional (*relational algebra*) maupun kalkulus relasional (*relational calculus*). Setiap model memiliki sejumlah aturan atau kaidah (*properties*) yang membedakan dengan model data lainnya, Connoly et.al (2003), Silberschatz (1997).

Setiap enititas data (*entity*) dipandang sebagai sebuah *relation* (tabel) yaitu baris (*rows*) dan kolom (*columns*). Baris (*row*) dinamakan *tuple*, terdiri dari satu set nilai data kolom (*columns*). Kolom (*column*) dinamakan pula sebagai atribut. Sel perpotongan (*pivot*) antara baris dengan kolom dinamakan nilai data (*state*) atribut.

Salah satu kaidah pada model relasional adalah pada sebuah relation tidak diperkenankan adanya duplikasi tuple, setidaknya berdasarkan kunci utama (*primary key*).

Kunci utama (*primary key*) berupa satu atau lebih atribut yang ada pada relation yang bersangkutan dipilih sebagai indentitas pembeda (unik dan minimal) diantara *tuple* yang ada.

Konsekuensi dari kaidah tersebut adalah apabila nilai data (*state*) satu atau beberapa atribut pada sebuah tuple berubah, maka cukup mengubah nilai atribut tersebut. Nilai data (*state*) yang sebelumnya digantikan oleh nilai baru. Nilai data atribut suatu *tuple* hanya menampung nilai data terbaru, dan menggambarkan keadaan terkini. Perubahan nilai data (*state*) di aplikasi terjadi seiring dengan waktu, dengan demikian pada basis datapun perlu diubah.

Sebagaimana telah dibahas pada poin sebelumnya, bahwa sebuah model data termasuk juga sistem manajemen basis data (dbms) dipengaruhi oleh aplikasi yang ada, dan teknologi media penyimpanan. Namun demikan basis data pada sebuah sistem aplikasi harus menyesuaikan diri dengan model data tertentu, karena pertimbangan teknologi yang tersedia. Jadi, secara logis dapat disimpulkan belum tentu semua aplikasi yang ada sesungguhnya telah dapat diakomodasi secara penuh oleh model data relasional.

Ada sejumlah aplikasi yang memerlukan selain data masa kini, namun juga memerlukan data masa lalu. Hal ini karena aplikasi tersebut perlu melakukan analisa ke masa lalu, untuk melakukan perencanaan strategi. Demikian pula ada aplikasi yang memerlukan melakukan pengubahan (koreksi) pada data masa lalu, contohnya pada aplikasi akuntansi, dan masih ada aplikasi lainnya.

Sebuah contoh, sebuah tuple {B1010, Pensil 2B Faber Castel, 2500} pada r*elation* (tabel) Barang terdiri dari atribut <Kode_Barang, Nama_Barang, Harga_Barang>. Bila harga barang tersebut berubah karena kenaikan harga menjadi 2650, maka tuple tersebut harus diubah dengan nilai baru tersebut, dan nilai harga sebelumnya 2500 dilupakan. Bagaimana penanganannya bila sistem aplikasi memerlukan data masa lalu, misalnya untuk melakukan analisa mengenai pengaruh jumlah barang yang terjual dengan berubahnya harga.

Salah satu solusinya adalah pada relation tersebut ditambahkan atribut waktu yaitu Tanggal, sehingga atribut *relation* tersebut menjadi <Kode_Barang, Nama_Barang, Harga_Barang, Tanggal>. Kunci utama (*primary key*) pada *relation* tersebut sebelum adanya tambahan atribut cukup menggunakan atribut <Kode_Barang> maka pada *relation* yang telah dimodifikasi, kunci utama (*primary key*) menjadi <Kode_Barang, Tanggal>. Sehingga data tuple manjadi {{B1010, Pensil 2B Faber Castel, 2500, 5-Jan-2004}, {{B1010, Pensil 2B Faber Castel, 2500, 15-Apr-2004}}. Hal tersebut harus dilakukan, karena bila tidak dimodifikasi akan melanggar kaidah utama model relasional yaitu tidak boleh ada duplikasi tuple (Kode_Barang).

Pemilihan atribut-atribut *primary key* harus diupayakan tidak dilakukan pengubahan nilai data (*state*). Bagaimana bila ternyata ada kesalahan nilai data atribut Tanggal, semestinya {13 Apr-2004}, dan bukan {15-Apr-2004}. Kejadian koreksi nilai atribut Tanggal mungkin terjadi, dan ini menimbulkan kesulitan, dan akan bertambah rumit bila terjadi pengubahan secara berantai (*cascade*).

Dalam jangka panjang, kebutuhan sistem aplikasi hampir dipastikan akan berkembang. Ada sejumlah atribut perlu ditambahkan untuk mengakomodasi kebutuhan baru, dan mungkin pula sejumlah atribut sudah tidak diperlukan lagi, demikian pula mungkin ada perubahan spesifikasi atribut. Jadi dalam jangka panjang sangat mungkin diperlukan pengubahan struktur skema basis data (*schema evolution*) pada satu atau lebih *relation type(s)*.

Sangat mungkin pada umumnya pada aplikasi sesungguhnya membutuhkan pula data masa lalu. Namun karena keterbatasan teknologi yang ada sehingga belum diakomodasi sepenuhnya. Teknologi media penyimpanan kini memungkinkan dan mampu menampung data yang sangat besar dibandingkan masa sebelumnya.

4. Basis Data Temporal

Seiring dengan semakin meningkatnya kapasitas media penyimpanan pada *non-volatile* (*disk*) maupun *volatile* (*memory*) maka aplikasi-aplikasi yang memerlukan data masa lalu, mulai dapat didukung. Nilai data atribut berubah karena adanya perubahan yang terjadi di dunia aplikasi, misalnya perubahan harga karena ada kenaikan harga, atau penurunan harga. Berubahnya nilai data secara implisit seiring dengan berjalannya waktu. Jadi perubahan nilai data selalu dihubungkan dengan waktu.

Basis data temporal, pada prinsipnya adalah basis data yang dimaksudkan untuk mendukung dan mendekati karakteristik alami (*nature*) sebuah sistem aplikasi yang hidup dan berkembang. Sebuah sistem aplikasi yang hidup dan berkembang dicirikan oleh adanya perubahan nilai data seiring waktu, demikian pula perubahan kebutuhan sistem (*system requirements*) yang juga perubahannya dihubungkan dengan waktu. Kecenderungan lain adalah basis data temporal diaplikasikan bersama data spasial, yang dikenal dengan spatio-temporal, Khatri, et.al (2001), Griffiths, et.al (2001), Griffiths, et.al (2001+).

Perubahan nilai data atribut pada sebuah tuple (meminjam istilah pada model relasional) dinamakan versi data. Perubahan kebutuhan direpresentasikan oleh perubahan skema atau struktur basis data, pada satu atau lebih *entity-type* atau *relation-type*, dan dikenal dengan evolusi skema (*schema evolution*).

Setiap perubahan nilai data yang tejadi pada dunia nyata aplikasi (*real world application*) disimpan pada basis data. Umumnya terjadi perbedaan waktu antara munculnya kejadian (*event*) di dunia nyata aplikasi dengan waktu data tersebut tersimpan pada basis data. Bilamana waktu kejadian (*event*) perubahan nilai data di dunia nyata terjadi sebelum waktu penyimpanan dinamakan sebagai *retroactive*. Sedangkan bilamana waktu penyimpanan perubahan di basis data dicatat lebih awal dibandingkan dengan kejadian sesungguhnya dalam realitas aplikasi, dinamakan *proactive,* Sutrisno (1997).

Taksonomi waktu terkait dengan waktu kejadian yang muncul pada realitas aplikasi, dan waktu saat penyimpanan dalam basis data dapat dilihat pada, Connolly (2003), Chakravarthy, et.al (1994), Kim, et.al (1994), Ling, et.al (1992), Worboys (1994), Jensen, et.al (1994), Pissinou, et.al (1994), Snodgrass (1992). Beberapa istilah mengenai waktu, diantaranya *event time* dan *transaction time*, Chakravarthy, et.al (1994), Kim, et.al (1994), *logical time* dan *physical time*, Ling, et.al (1992), *database time* dan *event time*, Worboys (1994), *valid-time*, *transaction-time*, dan *user-defined time*, Jensen, et.al (1994), Pissinnou, et.al (1994), Snodgrass (1992).

Basis data cuplikan (*snapshot database*) adalah basis data yang hanya mencatat nilai data atau status terbaru tanpa menyertakan waktu. Basis data ini merupakan kasus khusus dari basis data temporal. Basis data relasional adalah salah satu contohnya.

Basis data gulir balik (*rollback database*) adalah basis data yang mencatat waktu saat penulisan data ke dalam disk, dinamakan *transaction-time*. Basis data ini tidak mampu memberikan jawaban kapan sebuah data mulai berlaku di realitas sistem.

Tabel 1. Rollback database

Nama	***Divisi***	***Transaction Time***
Alice Johan Mary	Pemasaran Produksi Akunting	5
Alice Johan Mary Jimi	Pemasaran Distribusi Akunting Produksi	10
Johan Mary Jimi Evelyn	Distribusi Akunting Produksi Pemasaran	12

Basis data historis (*historical database*) adalah basis data yang mencatat kejadian perubahan data pada realitas aplikasi. *Valid-time* menyatakan kapan suatu nilai data muncul di realitas aplikasi.

Tabel 2. Sebelum perbaikan

Nama	***Divisi***	***Valid Time***
Johan	Produksi	2 Pebruari 1990
Johan	Distribusi	1 Nopember 1994

Tabel 3. Setelah perbaikan

Nama	***Divisi***	***Valid Time***
Johan	Produksi	12 Pebruari 1990
Johan	Distribusi	1 Nopember 1994

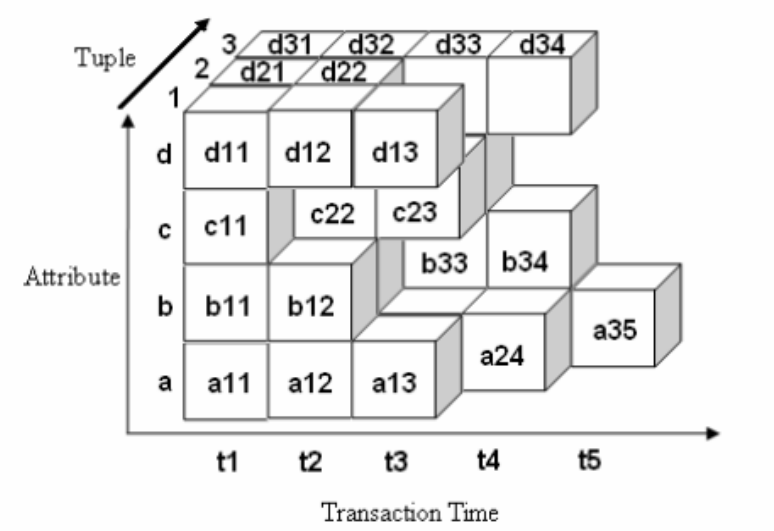
Basis data temporal (*temporal database*) adalah basis data yang menggabungkan kemampuan *rollback database* dan *historical database*. Basis data ini disebut pula sebagai bitemporal database, inilah yang dimaksudkan dalam tulisan ini sebagai temporal database.

Tabel 4. Basis data temporal visualisasi pandangan user

Eoid	A	B	C	D	Eoid version	Vt_s	Vt_e	Tt
1	a11	b11	c11	d11	1	1	3	1
2	a21	b21	c21	d21	1	1	3	2
3	a31	b31	c31	d31	1	1	3	3
1	a12	b12	c11	d12	2	4	8	7
2	a22	b22	c22	d22	2	4	8	8
3	a32	b32	c32	d32	2	4	8	9
1	a13	b12	c11	d13	3	9	-	12
2	a23	b22	c23	d22	3	9	12	13
3	a33	b33	c33	d33	3	9	12	14
2	a24	b22	c22	d22	4	13	-	17
3	a34	b34	c33	d34	4	13	17	18
3	a35	b34	c33	d34	5	18	-	23

Sebuah *relation* (tabel) basis data temporal (lihat Tabel 4), *valid-time* dinyatakan dalam bentuk waktu interval (*valid-time-start* Vt_s, dan *valid-time-end* Vt_e), sedangkan *transaction-time* Tt dinyatakan dalam bentuk waktu titik (*time point*). Setiap perubahan nilai data atribut sebuah *tuple* dibedakan menggunakan Eoid version. Setiap tuple diidentifikasi dengan Eoid (*entity occurence identity*) yang unik. Setiap versi data suatu *tuple* memiliki masa berlaku, dan tidak ada dua versi data berlaku pada waktu yang sama. Pada kolom *valid-time-end*, symbol tanda minus ("-") menyatakan bahwa *tuple* versi data tersebut masih berlaku hingga kini (sebagai *current value*).

Pengubahan atau pemutakhiran seperti tampak pada Tabel 4 maupun Gambar 1 terjadi dan dilakukan pada data terkini (*current*). Kotak berbayangan (*shadow*) menunjukkan versi data terkini dari setiap atribut. Versi terkini ditunjukkan oleh nilai *transaction-time* tertinggi pada setiap atributnya.



Gambar 1. Visualisasi basis data temporal pada tataran fisik

Basis data temporal yang mampu mendukung pengubahan pada versi data masa lalu, baik berupa pengubahan nilai data, waktu berlaku, atau nilai data dan waktu berlaku sekaligus, dinamakan basis data temporal dinamis, Siau (2003). Ada 38 skenario yang mungkin dan dibagi menjadi 6 kelompok, Sutrisno (1997), Sutrisno (2002).

Pada Tabel 4, pada *tuple* ketujuh, *tuple* dengan Eoid (1), dan Eoid Version (3), Vt_s(9), Vt_e(-), dan Tt(12) [20]. Meski pada realitas aplikasi *tuple* tersebut telah berubah, namun bila mengakses ke basis data pada *transaction-time* Tt(10), maka yang akan disajikan adalah {a12, b12, c11, d12}, dan bukan {a13, b12, c11, d13} yang baru dicatat pada Tt(12). Nilai data yang diperbaharui pada atribut A dan D, yaitu a13, dan d13.

Adanya penyertaan waktu memungkinkan data dapat ditelusuri berdasarkan waktu tertentu. Akses ke basis data akan memberikan informasi yang berbeda tergantung pada titik waktu pengamatan. Nilai data tertentu adalah merupakan data terbaru pada titik waktu tertentu.

Operasi-operasi primitif terdiri dari operasi pendefinisian data, dan operasi manipulasi data, Sutrisno (1997), Sutrisno (2002), Sutrisno (2003), Sutrisno (2004). Operasi-operasi primitif manipulasi data terdiri dari: *insert*, *update*, *delete*, *inactive*, *activate*, dan kelompok operasi *retrieve*. Operasi-operasi primitif pendefinisian data diantaranya: *create database*, *create schema*, *modify schema*, *add attribute*, dan operasi lainnya.

Pada kelompok operasi primitif *retrieve* disediakan sejumlah operasi yang berhubungan dengan ekstraksi data berdasarkan *valid-time*, dan *transaction-time*. Beberapa operasi tersebut diantaranya: *intersection*, *before*, *after*, *inside*, *between*, dan operasi lainnya, Sutrisno (2003), Sutrisno (2004).

5. Strategi Implementasi

Ada tiga pilihan strategi dalam implementasi penyertaan aspek waktu pada basis data, yaitu pada tataran atas (*high level*), tataran menengah (*intermediate level*), dan tataran bawah (*low level*). Strategi pada tataran tingkat atas dilakukan pada tingkat aplikasi, yaitu penyertaan waktu (pada model relasional) sebagai atribut biasa seperti atribut lainnya pada sebuah tabel (*relation*). Atribut waktu disertakan sebagai bagian dari kunci utama (*primary key*). Akses ke basis data selalu melibatkan atribut waktu sebagai filter kondisi.

Strategi pada tataran menengah (*intermediate level*) dilakukan dengan melakukan modifikasi pada bahasa *query* (SQL pada model relasional), seperti pada SQL-3. Modifikasi pada *query processsor*, yaitu menambahkan sejumlah operator temporal (*after*, *before*, dan

lain-lain), dan tipe data seperti DATETIME. Pada tataran ini tidak melakukan perubahan apapun pada mesin basis data-nya (*database engine*). Pilihan strategi ini memang cukup logis, karena biaya yang tinggi.

Strategi implementasi pada tataran bawah (*low level*) adalah melakukan modifikasi pada *database engine* agar lebih sesuai dengan karakteristik data temporal. Bagi sebuah dbms yang telah mapan maka biaya modifikasi menjadi sangat mahal, karena sama dengan membuat baru dari awal (*scratch*).

Strategi implementasi pada *low level* inilah yang diusulkan dalam tulisan ini, Sutrisno (1997), Sutrisno (2002), Sutrisno (2003), Sutrisno (2004). Pertimbangannya adalah memberikan kesempatan yang luas untuk proses belajar membangun sebuah dbms dari awal (*scratch*). Memberikan keluwesan yang lebih besar dalam pengembangannya. Konsekuensi dari pemilihan strategi ini adalah melakukan banyak pekerjaan dari mulai merancang model data, arsitektur, struktur data, hingga *coding*.

Basis data temporal dapat dikembangkan pada model relasional, tentunya dengan beberapa penyesuaian dalam kaidahnya, Ling, et.al (1992), Jensen, et.al (1994), Snodgrass (1992), Donghui, et.al (200), Khatri, et.al (2001), Skyt, et.al (2001), Sutrisno (2003), Sutrisno (2004). Basis data temporal dapat pula dikembangan pada model berorientasi obyek, Sutrisno (1997), Dumas, et.al (2004), Pissinou, et.al (1994), Griffiths, et.al (2001), Griffiths, et.al (2001+), Sutrisno (2002).

5.1 Model Relasional

Membuat *database engine* dari awal mulai dari ide, konsep, rancangan hingga implementasi. Mendefinisikan model data, tipe data yang akan didukung, arsitektur, operasi-operasi dasar (primitif) yang berhubungan dengan pendefinisan data, juga manipulasi data, organisasi berkas (*file*), rancangan logikal, hingga rancangan fisik, Sutrisno (2003).

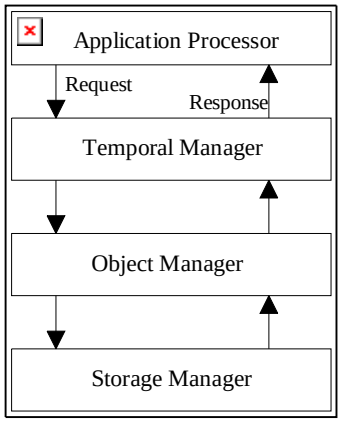
Prototip *database engine* dibangun berdasarkan rancangan tersebut, dikembangkan, dan diuji coba. Tujuan dari prototip tersebut untuk memverifikasi bahwa rancangan benar-benar dapat diimplementasi. Prototip telah berhasil dibangun menggunakan bahasa pemrograman Java, diuji coba, dan dapat diakses dengan aplikasi yang ditulis dalam bahasa Java, Sutrisno (2004).

5.2. Model Berorientasi Obyek

Aktifitas yang sama seperti pada 5.1 dilakukan juga pada pembuatan *database engine* berdasarkan model berorientasi obyek, Sutrisno (1997). Prototip dibangun menggunakan Borland C++, serta memanfaatkan beberapa pustaka (*library*) untuk pengolahan *stream*. Prototip tersebut dinamakan *Temporal Object Manager* (TOM).

Prototip diujicoba pada program aplikasi yang ditulis dalam bahasa C++ secara orientasi obyek. Setiap *object class* temporal pada program aplikasi diturunkan dari *class* temporal yang telah disediakan oleh prototip, dan berperan sebagai *superobject* temporal, Sutrisno (1997).

Arsitektur prototip TOM terdiri lapisan (layer) *Storage Manager*, *Object Manager*, *Temporal Manager*, dan *Application Processor*.



Gambar 2. Arsitektur Temporal Object Manager (TOM)

6. Kesimpulan

Basis data temporal merupakan sebuah perluasan aplikasi yang menyertakan aspek waktu pada basis data. Karakteristik alamiah sebuah aplikasi yang hidup dan berkembang ditandai dengan data yang mengalami pembaharuan status seiring dengan waktu.

Pilihan strategi implementasi pada tataran bawah (*low level*) yaitu membuat *database engine* memberikan peluang dan kesempatan yang luas untuk proses belajar membangun sebuah dbms dari awal (*scratch*). Memberikan keluwesan yang lebih besar dalam pengembangannya. Basis data temporal dapat dikembangkan pada model relasional, maupun pada model berorientasi obyek.

7. Pengembangan Lanjutan

Prototip-prototip yang dikembangkan baru pada *database engine*, dan hal ini perlu ditambahkan dengan fasilitas seperti concurrency control, security control, integrity, dan recovery. Penambahan bahasa pengolah query (query language processor), antar-muka yang lebih ramah pemakai, serta basis data dapat diakses dalam lingkungan *client-server*.

Pengembangan prototip basis data temporal diaplikasikan bersama dengan aplikasi data spasial. Hal ini tentu saja memerlukan studi lebih lanjut.

Daftar Pustaka

Chakravarthy, Sharma, and Seung-Kyum Kim. (1994). Resolution of Time Concept in Temporal Databases. *Information Sciences* 80:91-125.

Connolly, Thomas N., and Carolyn Begg, and Anne Strachan. (2003). *Database System A Practical Approach to Design, Implementation, and Management*. Addison Wesley, 3rd edition.

Donghui, Zhang, Vassilis J. Tsotras, and Bernhard Seeger. (2000). A Comparison of Indexed Temporal Join. TR-50, August. *A TIMECENTER Technical Report*.

Dumas, M., Marie-Christine Fauvet, Pierre-Claude Scholl. (2004). TEMPOS: A Platform for Developing Temporal Applications on Top of Object DBMS. *IEEE Transaction on Knowledge And Data Engineering*, March, 16(3):354-374.

Elmasri, Ramez, and Shamkant B. Navathe. (2000). *Fundamental Database Systems*. Benjamin Cummings, 3rd edition.

Griffiths, Tony, Alvaro A.A. Fernandez, Norman W. Paton, Bo Huang, Mike Worboys, Chris Johnson, Keith T. Mason, John Stell. (2001). Tipod: A Comprehensive System for the Management of Spatial and Aspatial Historical Objects. *Tripod Project*.

Griffiths, Tony, Alvaro A.A. Fernandez, Norman W. Paton, Bo Huang, Mike Worboys, Chris Johnson, Keith T. Mason. (2001+) Tripod: A Comprehensive Model for Spatial and Aspatial Historical Objects. *Tripod Project*.

Jensen, Christian S., and Richard Snodgrass. (1994). Temporal Specialization and Generalization. *IEEE Transactions on Knowledge and Data Engineering*, December 6(6):954-974.

Khatri, Vijay and Sudha Ram and Richard T. Snodgrass, and Grady O'Brien. (2001). Supporting User-defined Granularities and Indeterminacy in a Spatiotemporal Conceptual Model. TR-55, April. *A TIMECENTER Technical Report.*

Kim, Won. (1992). *Introduction to Object-Oriented Database*. The MIT Press Cambridge, Masschussetts, London, England, third printing.

Kim, Seung-Kyum, and Sharma Chakravarthy. (1994). Temporal Databases with Two-Dimensional Time: Modeling and Implementation of Multihistory. *Information Sciences* 80:43-89.

Ling, D.H.O., and D.A. Bell. (1992). Modelling and Managing Time in Database Systems. *The Computer Journal* 30(4):332-341.

Pissinou, Niki, and Kia Maki. (1994). A Coherent Architecture for a Temporal Object Data Base Management System. *Journal Systems Software* 27:195-205.

Siau, Keng (editor). (2003). *Advanced Topics in Database Research*. Idea Group Publishing, Vol. 2.

Silberschatz, Abraham, Henry F. Korth, S. Sudarshan. (1997). *Database Systems Concepts*. McGraw-Hill, 3rd edition.

Skyt, Janne, and Christian S. Jensen. (2001). Persistent Views–A Mechanism for Managing Aging Data. TR-65, July. *A TIMECENTER Technical Report.*

Snodgrass, Richard. (1992). Temporal Query Language TQuel. *ACM Transaction on Database Systems* 12(2):247-298.

Sutrisno. (1997). *Basis Data Temporal Berorientasi Obyek: Implementasi Pengelolaan Versi Data*. Tesis Magister, Universitas Indonesia.

Sutrisno. (2002). Object-Oriented Temporal Database: Implementation of Version Management. *The Journal of The Computer Society of India,* June 32(2):19-25.

Sutrisno. (2003). *Attribute Temporal Database Management System: Technical Specification. Working Material*, Universitas Pelita Harapan.

Sutrisno, Lazarusli, Irene A, Phineas, Martin. (2004). Fasilitas Data Manipulation Language Pada Attribute Temporal Database Management System. *Jurnal Ilmiah Ilmu Komputer*, April, 2(2):103-112.

Worboys, Michael F. (1994). A Unified Model for Spatial and Temporal Information. *The Computer Journal* 37(1):26-34.