IMPLEMENTASI METODE BITEMPORAL PADA SISTEM INFORMASI KEPEGAWAIAN INSTITUT TEKNOLOGI SEPULUH NOPEMBER SURABAYA

A. Tjahyanto dan Cahyono

Jurusan Teknik Informatika, Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember (ITS) - Surabaya
Kampus ITS, Jl. Raya ITS, Sukolilo – Surabaya 60111
Tel. + 62 31 5939214, Fax + 62 31 5939363
Email : arist@its-sby.edu

ABSTRAK

Pengelolaan sumber daya manusia merupakan salah satu hal yang harus dilakukan untuk mendukung kemajuan sebuah organisasi. Institut Teknologi Sepuluh Nopember sebagai salah satu institusi pendidikan mempunyai beberapa sumber daya yang harus dikelola. Salah satu sumber daya yang dikelola adalah data tenaga edukatif, dimana setiap bulan harus dapat menghasilkan informasi tenaga edukatif yang sudah berhak mendapatkan kenaikan gaji berkala, kenaikan pangkat atau kenaikan jabatan. Informasi itu disimpan di dalam kartu indek pegawai sebagai catatan mutasi dosen selama menjadi pegawai.

Dalam penelitian ini dibuat suatu perangkat lunak yang mampu digunakan untuk pengelolaan data dosen Institut Teknologi Sepuluh Nopember (ITS). Dengan memanfaatkan metode bitemporal, yaitu sebuah cara yang memanfaatkan waktu sebagai atribut untuk menyatakan status baris-baris dalam relasi. Status baris dalam sebuah relasi dikeompokkan sebagai status baris terhapus (delete record), status baris riwayat (history record) dan status baris sekarang (current record). Status tersebut mempermudah pencarian dan penelusuran data .

Aplikasi yang dibuat telah di uji coba secara lengkap pada dua kelompok pengguna yaitu administrator dan dosen. Pengguna administrator bisa melakukan transaksi menaikkan pangkat, jabatan dan kenaikan gaji berkala. Sedangkan pengguna dosen bisa melakukan update biodata. Dan dari uji coba yang telah dilakukan dapat disimpulkan bahwa aplikasi dapat berjalan sebagaimana mestinya sesuai dengan fungsinya. Dengan demikian sistem ini diharapkan dapat dimanfaatkan untuk mengelola data kepegawaian dosen yang menjadi aset ITS.

Kata Kunci : Metode Temporal, Riwayat, *Valid time, Transaction Time*, Metode Bitemporal.

1. PENDAHULUAN

Dunia pendidikan tumbuh berkembang dengan pesat. Sebuah pendidikan tinggi memiliki sumber daya manusia dosen yang perlu dipelihara. Semakin banyak dosen yang ada maka semakin banyak pula masalah yang harus diatur supaya pendidikan tinggi tersebut bisa berjalan dengan baik dalam mengalokasikan sumber daya maupun dalam usaha memberi penghargaan kepada mereka dalam mengajar yang diberikan dalam bentuk gaji. Pangkat dan jabatan dan gaji biasanya dicatat dalam kartu indeks pegawai untuk mendapatkan data riwayat setiap pegawai.

Salah satu informasi yang ditangani ITS adalah melakukan pengolahan data tenaga edukatif dimana setiap bulan harus bisa menghasilkan informasi tenaga edukatif yang sudah saatnya berhak mendapatkan kenaikan gaji berkala. Dengan dikeluarkan peraturan pemerintah tentang perubahan gaji pada waktu-waktu tertentu menyebabkan terjadinya penyesuaian (*impassing*) gaji yang harus diterima dosen tersebut. Hal ini membutuhkan perhatian yang khusus karena dibutuhkan sebuah perubahan total dari pembayaran yang harus diberikan kepada tenaga edukatif sesuai dengan kepangkatannya pada bulan yang akan datang.

Sistem yang ada di ITS sampai sekarang masih menggunakan perhitungan dengan menggunakan aplikasi lokal yang dijalankan pada beberapa komputer. Hal ini mengurangi kinerja bagian kepegawaian karena banyak diantara mereka yang tidak bisa secara optimal menyelesaikan pekerjaan karena aplikasi hanya berjalan pada komputer tertentu.

1.1. Permasalahan

1. Bagaimana menyimpan dan memelihara data-data kepangkatan dosen.
2. Bagaimana mengatasi impassing.
3. Bagaimana melakukan penelusuran riwayat kepangkatan dan penggajian.

1.2. Batasan Masalah

Penelitian yang dibahas akan dibatasi pada ruang lingkupnya sebagai berikut

1. Sistem kenaikan pangkat dan kenaikan gaji berkala tenaga edukatif dosen Institut Teknologi Sepuluh Nopember (ITS).
2. Penggunaaan metode *Bitemporal Baris Versioning* menurut Elmasri Navathe.
3. Riwayat yang dihasilkan adalah riwayat aktual menurut transaksi, bukan riwayat fakta secara real

2. METODE BASISDATA TEMPORAL

Metode Temporal secara umum meliputi semua aplikasi basisdata yang memerlukan aspek waktu saat mengatur informasi. Aplikasi basisdata temporal sudah dibangun sejak awal penggunaan basisdata itu sendiri. Dalam menciptakan aplikasi temporal tersebut menjadi tugas desainer dan developer aplikasi untuk mencari sendiri bentuk desain, memprogram dan menerapkan konsep temporal yang mereka butuhkan. Terdapat banyak contoh aplikasi dimana aspek waktu diperlukan untuk memelihara informasi dalam sebuah basisdata. Termasuk diantaranya: rumah sakit yang membutuhkan pemeliharaan riwayat penyakit pasien, asuransi yang membutuhkan pemeliharaan riwayat klaim dan riwayat kecelakaan, sistem pemesanan secara umum yang membutuhkan pemeliharaan riwayat pemesanan saat ingin mengetahui apakah informasi atas tanggal dengan waktu pemesanan masih berlaku, basisdata *science* membutuhkan riwayat berupa data ekperimen dan waktu saat data itu diperoleh. Basisdata perusahaan untuk menyimpan riwayat gaji, pekerjaan dan proyek pada setiap pegawai. Basisdata universitas menyimpan riwayat indeks prestasi mahasiswa. Fakta-fakta tersebut menunjukkan bahwa secara nyata sebagian aplikasi basisdata mempunyai beberapa informasi temporal.

Pada bagian ini akan diperkenalkan konsep basisdata temporal yang dibangun untuk mengatasi kompleksitas aplikasi basisdata temporal.

2.1. Insert Bitemporal

Tabel 1. Contoh Operasi Insert proaktif dan retroaktif PEGAWAI_BT.

	NIP	NAMA	ALM	FK_P	WM_VL	WS_VL	WM_TR	WS_TR
V1	n1	Budi	solo	p001	6/15/97	now	6/1/97 20:20	uc
V2	n2	Amir	jogja	p001	6/15/97	now	6/30/98 20:56	uc

Operasi *insert bitemporal* dijalankan dengan menciptakan versi pertama seperti ditunjukkan oleh V1 dan V2 pada PEGAWAI_BT Tabel 3. Baris V1 merupakan contoh operasi *insert proaktif* dan Baris V2 merupakan contoh operasi *insert retroaktif*. Operasi insert proaktif terjadi apabila saat memasukkan record baru ke dalam tabel, waktu mulai valid data tersebut secara fakta belum terjadi, tetapi karena sudah diketahui akan terjadi maka dimasukkan, sehingga waktu mulai transaksi lebih awal daripada waktu mulai valid data tersebut. Sedangkan operasi insert retroaktif terjadi apabila saat memasukkan record baru ke dalam tabel, waktu mulai valid data tersebut secara fakta sudah terjadi pada waktu lalu, sehingga waktu mulai valid data tersebut lebih dulu terjadi daripada waktu mulai transaksinya.

2.2. Delete Bitemporal

Tabel 2. Contoh Operasi Delete proaktif dan retroaktif PEGAWAI_BT.

	NIP	NAMA	ALM	FK_P	WM_VL	WS_VL	WM_TR	WS_TR
V3	n1	Budi	solo	p001	6/15/97	now	6/8/97 13:05	6/1/99 20:56
V4	n1	Budi	solo	p001	6/15/97	**6/15/99**	**6/1/99 20:56**	uc
V5	n2	Amir	jogja	p002	6/15/97	now	6/8/97 20:56	6/30/99 11:18
V6	n2	Amir	jogja	p002	6/15/97	**6/15/99**	**6/30/99 11:18**	uc

Operasi *delete bitemporal* dilakukan pada relasi PEGAWAI_BT dengan menjalankan V3 dan V4 seperti Tabel 2. Pegawai yang bernama Budi dan Amir meninggalkan instansinya pada tanggal 15 Juni 1999.

Operasi *delete proaktif* baris Budi secara logika dilakukan dengan sebuah transaksi T pada TS(T) = 6/1/99 20:56. Sebelumnya V3 adalah current version sehingga WS_TR memiliki nilai *uc*. Operasi *delete proaktif* secara logika dijalankan dengan mengeset V3.WS_TR ke 6/1/99 20:56 untuk membuat baris tersebut menjadi tidak valid lagi. Selanjutnya menciptakan tupel V4 dengan WS_VL sebagai 6/15/99 dan mengisi WM_TR sebagai 6/1/99 20:56.

Operasi *delete retroaktif* baris Amir secara logika dilakukan dengan sebuah transaksi T pada TS(T) = 6/30/99 11:18. Sebelumnya V5 adalah current version sehingga WS_TR memiliki nilai *uc*. Operasi *delete retroaktif* secara logika dijalankan dengan mengeset V5.WS_TR ke 6/30/99 11:18 untuk membuat baris tersebut menjadi tidak valid lagi. Selanjutnya menciptakan tupel V6 dengan WS_VL sebagai 6/15/99 dan mengisi WM_TR sebagai 6/30/99 11:18.

2.3. Update Bitemporal

Tabel 3. Contoh Operasi *Update* proaktif dan r*etroaktif* PEGAWAI_BT.

	NIP	NAMA	ALM	FK_P	WM_VL	WS_VL	WM_TR	WS_TR
V7	n1	Budi	solo	p001	6/15/97	now	6/8/97 13:05	**6/1/99 20:56**
V8	n1	Budi	solo	p001	6/15/97	6/14/99	**6/1/99 20:56**	uc
V9	n1	Budi	solo	p002	**6/15/99**	now	**6/1/99 20:56**	uc
V10	n2	Amir	jogja	p050	6/15/97	now	6/8/97 11:18	**6/30/99 14:33**
V11	n2	Amir	jogja	p050	6/15/97	6/14/99	**6/30/99 14:33**	uc
V12	n2	Amir	jogja	p051	**6/15/99**	now	**6/30/99 14:33**	uc

Operasi *update bitemporal* dilakukan pada basisdata bitemporal dengan tidak ada atribut yang secara fisik diubah pada suatu baris kecuali pada WS_TR (*transaction end time*) dari nilai *uc* ke nilai waktu tertentu. Semula versi sekarang V7 tabel PEGAWAI_BT memiliki *uc* pada atribut WS_TR. Jika beberapa atribut sebagai contoh fk_p di update maka transaksi T yang digunakan untuk mengupdate harus memiliki dua buah parameter, yaitu fk_p yang baru dan VT ketika fk_p tersebut menjadi efektif dalam dunia nyata. Bila diasumsikan VT_ adalah titik waktu sebelum VT dalam granularitas waktu, sedangkan transaksi T memiliki timestamp TS(T). Kemudian perubahan dibawah ini akan dijalankan pada tabel PEGAWAI_BT pada operasi update :

1. Menyalin V8 dari current version V7. Baris V8 adalah salinan current version sebelumnya setelah ditutup pada valid time VT_. Langkah-langkahnya adalah dengan mengisi nilai atribut sebagai berikut :
   - V8.WS_VL ke VT_
   - V8.WM_TR ke TS(T)
   - V8.WS_TR ke *uc*

   Insert V2 ke dalam tabel .
2. Menyalin V9 dari current version V7. Baris V9 adalah current version yang baru. Sebelum diinsert dilakukan modifikasi pengisian atribut dengan langkah sebagai berikut
   - V9.WM_VL ke VT
   - V9.WS_VL ke now
   - V9.fk_p ke nilai baru
   - V9.WM_TR ke TS(T)
   - V9.WS_TR ke uc

   Insert V9 ke tabel.
3. Megubah nilai atribut V7 dengan jalan mengisi :
   - V7.WS_TR ke TS(T)

Hal ini menujukkan record tersebut secara logika sudah dihapus, karena current version tidak lagi menunjukkan informasi yang benar.

Operasi *update retroaktif* pada Tabel 3. tiga baris pertama V7,V8 dan V9 dari PEGAWAI_BT meupakan langkah yang diperlukan untuk mengupdate Budi dari pangkat dengan kode p001 ke p002. Sebelumnya pada V7 tabel PEGAWAI_BT hanya ada p001 karena itu baris ini menjadi current version dengan WS_TR bernilai *uc*. Kemudian sebuah transaksi T yang terjadi saat TS(T) pada 6/1/99 20:56 mengubah fk_p menjadi p002 dengan waktu valid efektif 6/15/99. Pertama, baris V8 diciptakan, yang merupakan salinan V7 kecuali WS_VL diisi 6/14/99, satu hari sebelum waktu valid baru terjadi dan WM_TR adalah timestamp saat transaksi update. Kedua, tupel V9 diciptakan yang memiliki fk_p baru WM_VL diisi 6/1/99 dan WM_TR juga merupakan timestamp saat transaksi update. Akhirnya WS_TR dari V7 diupdate dengan timestamp saat transaksi update. Proses update ini adalah *update proaktif*, karena transaksi update dilakukan pada tanggal 1 Juni 1999, tetapi fk_p berubah secara efektif pada tanggal 15 Juni 1999.

Operasi *update retroaktif* pada Tabel 3 dilakukan dengan cara yang hampir sama ketika Amir berubah pangkatnya sehingga fk_p harus diubah. Timestamp update terjadi pada 6/30/99 14:33 dengan waktu valid yang efektif pada 6/15/99. Proses update ini disebut retroaktif karena waktu transaksi 6/30/99 14:33 sedangkan waktu valid 6/15/99. Jadi baris V10 di-replace dengan V11 dan V12.

2.4. Batasan Metode Bitemporal

Sama dengan relasi *non-temporal*, relasi basisdata bitemporal juga memiliki batasan atau konstrain yaitu aturan yang harus dipenuhi dalam basisdata bitemporal. Dalam basisdata temporal konstrain tersebut mengalami beberapa penyesuian. *Domain constraint* mengatur tipe data, format atau range nilai serta pengecekan nilai sebuah kolom. Pembahasan sebelumnya menunjukkan bahwa basisdata bitemporal adalah basisdata yang bersifat *append only*. Artinya setiap operasi baik itu insert delete ataupun update harus dimasukkan ke dalam basisdata. Hal ini berakibat basisdata tersebut tidak mengenal konstrain *entity integrity*. Sehingga pada *primary key* terjadi duplikasi saat melakukan operasi delete maupun update. Seandainya *primary key* merupakan gabungan antara *primary key* temporal dan *non-temporal* dengan waktu mulai valid dan waktu mulai transaksi serta primary key non-temporal maka untuk sementara bisa mengatasi entity entegrity database bitemporal. Permasalahan akan muncul bila suatu saat pengguna akan mengupdate record baris dengan waktu mulai valid yang sama, maka pengguna itu akan melanggar aturan entity entegrity tersebut. Lebih jelasnya bisa dilihat dalam tabel berikut ini.

Tabel 4. Contoh Operasi *Update* retroaktif PEGAWAI_BT.

	NIP	NAMA	ALM	FK_P	WM_VL	WS_VL	WM_TR	WS_TR
V13	n1	Budi	solo	p001	6/15/97	now	6/8/97 13:05	6/1/99 20:56
V14	n1	Budi	solo	p001	6/15/97	6/14/97	6/1/99 20:56	uc
V15	n1	Budi	solo	p002	6/15/97	now	6/1/99 20:56	uc

Saat pertama kali baris v13 menunjukkan baris Budi ditambahkan ke dalam tabel dengan NIP, WM_VL dan WM_TR sebagai *primary key*. Record tersebut menyatakan bahwa Budi dengan primary key [n1,6/15/97,6/8/97 13:05] berada pada pangkat p001. Ternyata pengguna salah dalam melakukan proses input yaitu record Budi harusnya memiliki pangkat p002 maka harus dilakukan proses update.

Proses update kemudian dilakukan sehingga menurut aturan metode bitemporal menghasilkan baris v14 dan v15 seperti pada Tabel 6. Baris v14 dan v15 menyalahi entity integrity sebuah tabel. Pemecahan yang bisa dilakukan adalah dengan menambahkan sebuah *field* yang bertipe *autoincrement number.*

Perbedaan lain antara bitemporal dan non-temporal basisdata terletak pada bagian *referential integrity*. Pada batasan *referential* ini yang menjadi *foreign key* adalah *primary key* yang bukan temporal.

Referential integrity disini juga harus bisa memastikan relasi antara *primary key* dan *foreign key* yaitu :

- Tabel bitemporal tidak bisa didelete secara fisik
- Jika menambahkan *foreign key* ke tabel detail harus berasal dari *primary key* tabel *master*.

Untuk memenuhi kebutuhan refferntial integrity ini bisa dibuat sebuah view tabel master untuk current version. Pada tersebut tidak diperbolehkan adanya duplikasi. Dari view tersebut barulah dipetakan primary key menjadi foreign key tabel detail. Misalnya ada kemungkinan terjadi baris current maka yang lebih dari satu maka harus dipilih baris dengan waktu update yang paling akhir atau dengan nilai autoincrement terbesar.view

Akibat penting update master yang bisa dihasilkan adalah tabel detail akan menunjuk master yang lebih dari satu. Riwayat tabel detail bisa diperoleh lewat pengecekan keadaan tabel master dan tabel detail tersebut dengan skrip SQL untuk menghasilkan view master versi current dan view detail versi riwayat .

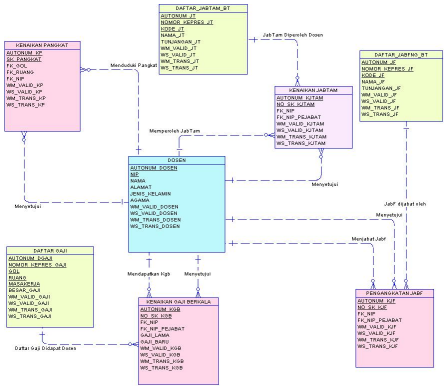
3. PERANCANGAN PERANGKAT LUNAK

Untuk membuat aplikasi pada penelitian ini terlebih dahulu dilakukan perancangan data, perancangan proses dan perancangan antar muka. Perancangan ini dimaksudkan untuk memudahkan dalam membuat aplikasi.

Perancangan data meliputi perancangan data masukan yang dibutuhkan oleh perangkat lunak, pemrosesan data, serta data keluaran yang dihasilkan oleh perangkat lunak.

Data masukan utama yang dibutuhkan oleh perangkat lunak yang dibuat disimpan dalam tabel. Tabel meupakan bentuk perwujudan dari entitas atau sekumpulan data yang bisa dibedakan. Untuk menunjukkan hubungan antara entitas di dalam sebuah sistem dibuat diagram relasi entitas. Diagram relasi entitas berguna untuk memberikan gambaran hubungan antara relasi sehingga dapat diimplementasikan. Entitas yang digunakan dalam sistem adalah adalah :

1. Entitas Dosen
2. Entitas Daftar Gaji
3. Entitas Daftar Jabatan Fungsional
4. Entitas Daftar Jabatan Tugas Tambahan
5. Entitas Kenaikan Pangkat
6. Entitas Kenaikan Gaji Berkala
7. Entitas Kenaikan Jabatan Fungsional
8. Entitas Kenaikan Jabatan Tugas Tambahan



Gambar 4. Diagram relasi entitas kepegawaian.

4. UJI COBA DAN ANALISIS

Uji coba yang dilakukan secara berulang berdasarkan skenario yang sudah di rancang perlu dilakukan analisis terhadap hasil uji coba, sehingga nantinya dapat diketahui kelebihan dan kekurangan yang terdapat dalam metode pada kasus sistem informasi kepegawaian ini. Sistem Informasi ini akan bisa dijadikan bahan referensi Biro Administrasi Umum dan Keuangan Institut Teknologi Sepuluh Nopember untuk pengembangan aplikasi selanjutnya.

4.1. Hasil Uji Coba Tingkat Administrator

Dalam uji coba tingkat administrator yang dilaksanakan berdasarkan skenario uji coba sebelumnya diperoleh suatu hasil bahwa penyediaan informasi riwayat bisa diperoleh dengan menggunakan metode ini. Sebagaimana pelaksanaan uji coba yang telah dilakukan. Sebuah proses telah dijalankan untuk menyediakan data *current* dari masing masing entitas sebelum direlasikan. Proses yang lain juga dilakukan untuk mendapatkan data versi terakhir. Transaksi pengubahan atau update dilakukan dengan ketentuan data yang bisa diupdate adalah data terakhir. sedangkan usaha untuk mendapatkan data riwayat diperoleh dari data riwayat sejak pertama sampai waktu sekarang. Operasi penghapusan tidak dilaksanakan secara langsung artinya penghapusan dilakukan dengan menjalankan pengubahan. Proses pencarian data

current ini melibatkan intruksi yang panjang. Pemasukkan data transaksi secara *retroaktif* maupun *proaktif* tetap akan menghasilkan data riwayat yang konsisten seiring dengan berjalannya waktu. Kemungkinan yang terjadi adalah adanya data yang tumpang tindih. Tetapi karena waktu yang dianggap valid adalah waktu saat ini maka kejadian yang waktu mulai transaksinya paling mendekati batas kurang dari waktu sekarang adalah informasi yang paling valid selain itu bisa juga dimanfaatkan atribut *autoincrement*.

Saat melakukan kenaikan pangkat dosen ke jenjang berikutnya maka sistem akan mencari data dosen yang *versinya current* dan data kenaikan pangkat yang dengan *versi terakhir*. Aksi yang melibatkan data dosen *versi current* dan data pangkat versi terakhir ini juga diwajibkan memasukkan waktu valid baris. Waktu valid masukkan menyebabkan data diterima secara *retroaktif* atau *proaktif*. Akibat yang ditimbulkan dari inputan transaksi ini adalah ditemukannya adanya data-data yang secara logika dihapus, data riwayat, data sekarang, dan data yang sekarang belum berlaku. Jadi pada suatu relasi bisa ditemuakan daftar data yang terhapus, daftar data riwayat dan daftar data current, serta daftar data calon current dan atau calon histori.

4.2. Hasil Uji Coba Tingkat Dosen

Pengujian tingkat dosen juga dilaksanakan berdasarkan skenario uji coba yang sudah dipersiapkan. Sesuai dengan uji coba pengubahan biodata dosen di atas diperoleh data dosen dalam sebuah relasi yang secara logika bisa dibagi menjadi daftar baris terhapus, daftar baris riwayat, dan daftar baris current. Sebuah proses telah dilakukan untuk menyediakan baris terakhir kali dosen mengubah biodatanya. Pengubahan biodata dosen dilakukan dengan mengubah data terakhir tersebut. Jika data terakhir tersebut kurang dari waktu sekarang maka akan diperoleh sebuah histori. Tetapi bila data terakhir tersebut lebih dari waktu sekarang maka akan diperoleh calon baris riwayat. Pemasukkan data transaksi secara *retroaktif* maupun *proaktif* tetap akan menghasilkan data riwayat yang konsisten seiring dengan berjalannya waktu.

5. KESIMPULAN

Dari pengamatan dan beberapa uji coba yang telah dilakukan, disimpulkan bahwa:

- Untuk menerapkan metoda bitemporal, harus dilakukan sedikit perubahan skema basisdata, antara lain untuk menampung catatan yang diiperlukan oleh bitemporal. Kolom tambahan berjumlah empat buah, antara lain waktu mulai valid, waktu selesai valid, waktu mulai transaksi, dan waktu selesai transaksi.
- Dibutuhkan ruang penyimpanan yang lebih besar bila dibandingkan basisdata aslinya.
- Diperlukannya *view valid current version* untuk setiap tabel.

DAFTAR PUSTAKA

[BUR-87] Burhannudin, Tayibnapos. *Administrasi Kepegawaian Suatu Tinjauan Analitik*. Praditya Paramita,Jakarta. 1987.

[ELM-99] Elmasri Ramez, Navathe. *Fundamental of Database System*. Addison Wesley, 3rd Edition, 1999.

[HOM-97] Homer Alex, Andrew Enfield. *Profesional Active Server Pages*. Wrox Press Ltd. 30 Lincold Road, Oltan, Birmingham, B27 6 PA USA. 1997.

[MIC-02] Michael Minock. *Temporal Database Concept*. Umea Universited. http://www.cs.umu.se/kurser/TDBD15/VT01/4temporal.pdf.2002

[RAM-99] Ramalho Jose. *SQL Server 7.0*. Wordware Publishing. 1999.

[WAY-99] Waymire Richard, Rick Sawtell. *Teach Your Self Microsoft SQL Server 7.0 in 21 Days*. Sam Publishing.1999.