Tugas Basis Data Sesi 6

Aldi Maulana Iqbal – 2021080122

Functional Dependency and Normalisation

Pengertian

Normalisasi adalah proses mengatur data dalam tabel agar sesuai dengan prinsip-prinsip normalisasi yang diterapkan. Prinsip-prinsip ini diterapkan untuk mengoptimalkan desain basis data dan menghindari masalah seperti data yang berulang, kesulitan dalam pembaruan data, dan masalah performansi. Normalisasi juga membantu dalam mengurangi risiko kesalahan data dan mempermudah pemeliharaan basis data.

Normalisasi dibagi menjadi beberapa level, yaitu normalisasi level satu (1NF), normalisasi level dua (2NF), dan normalisasi level tiga (3NF). Setiap level memiliki kriteria yang harus dipenuhi dalam desain tabel. Pada level satu, tabel harus memiliki kolom yang unik dan tidak boleh ada pengulangan data. Pada level dua, setiap kolom harus tergantung pada kunci utama tabel dan tidak boleh ada ketergantungan fungsional parcial. Pada level tiga, setiap kolom harus tergantung hanya pada kunci utama tabel dan tidak boleh ada ketergantungan fungsional transitive.

Normalisasi membantu dalam menjaga konsistensi data dan mempermudah proses pembaruan dan pengambilan data. Namun, normalisasi yang berlebihan dapat menyebabkan masalah performansi dan kesulitan dalam pengambilan data yang kompleks. Oleh karena itu, normalisasi harus diterapkan dengan hati-hati dan sesuai dengan kebutuhan aplikasi yang digunakan.

Pedoman Desain Informal untuk Skema Relasi

Ada empat ukuran kualitas informal untuk desain skema relasi. Langkah-langkah ini tidak selalu independen satu sama lain. Empat tindakan informal ini adalah:

1. Pedoman 1 menyatakan bahwa ketika atribut dikelompokkan untuk membentuk skema relasi, diasumsikan bahwa atribut yang termasuk dalam satu relasi memiliki arti kata-nyata tertentu dan interpretasi yang tepat terkait dengannya. Makna ini (Semantik) menentukan bagaimana nilai

atribut dalam tupel berhubungan satu sama lain. Rancangan konseptual harus memiliki makna yang jelas, jika dilakukan dengan hati-hati, diikuti dengan pemetaan sistematis ke dalam hubungan dan sebagian besar semantik akan dapat dipertanggungjawabkan. Oleh karena itu, desain skema relasi yang baik harus mudah dijelaskan maknanya dan atribut dari beberapa tipe entitas dan tipe relasi tidak boleh digabungkan menjadi satu relasi, sehingga skema relasi yang sesuai dengan satu tipe entitas atau satu tipe relasi memiliki arti langsung.

2. Salah satu tujuan utama desain skema dalam desain database adalah untuk meminimalkan ruang penyimpanan yang dibutuhkan oleh relasi dasar. Ini dilakukan dengan mengelompokkan atribut ke dalam skema relasi. Namun, masalah besar lainnya yang muncul saat menggunakan relasi sebagai relasi dasar adalah masalah pembaruan anomali. Pembaruan anomali terjadi ketika perubahan data pada satu bagian dari relasi menyebabkan masalah pada bagian lain dari relasi. Terdapat tiga jenis pembaruan anomali yang umum dalam suatu relasi, yaitu Anomali Penyisipan, anomali penghapusan, dan Anomali modifikasi. Untuk mengatasi masalah ini, PEDOMAN 2 menyarankan untuk merancang skema relasi dasar sedemikian rupa sehingga tidak ada anomali pembaruan yang ada dalam relasi. Jika masih ada anomali pembaruan, maka program yang memperbarui database harus diperiksa dan diperbaiki agar dapat beroperasi dengan benar.
3. Nilai nol dalam tupel adalah masalah yang sering terjadi ketika banyak atribut dikelompokkan bersama dalam relasi "gemuk" dan banyak atribut tidak berlaku untuk semua tupel dalam relasi. Hal ini menyebabkan banyak NULL dalam tupel tersebut, yang menghabiskan banyak ruang dan tidak mungkin untuk memahami arti atribut yang memiliki Nilai NULL. Masalah lain terjadi saat menentukan operasi gabungan. Salah satu masalah utama dan terpenting dengan Null adalah bagaimana menghitungnya ketika operasi agregat (yaitu, COUNT atau SUM) diterapkan. Null dapat memiliki banyak interpretasi, seperti atribut tidak berlaku untuk aturan ini, atribut tidak diketahui untuk tuple ini, atau nilai diketahui tetapi tidak ada yaitu, tidak dapat direkam. Untuk mengatasi masalah ini, perlu diingat PEDOMAN 3 yaitu cobalah untuk menghindari, menempatkan atribut dalam relasi dasar yang nilainya biasanya NULL. Jika Null tidak dapat dihindari, pastikan

bahwa hanya berlaku dalam kasus luar biasa dan sebagian besar tupel harus memiliki beberapa Nilai bukan NULL.

4. Generation of spurious tuples adalah masalah yang muncul ketika skema relasi dibagi menjadi dua relasi R1 dan R2 dan kemudian digabungkan kembali menggunakan NATURAL Join. Operasi gabungan ini dapat menghasilkan tupel palsu yang tidak valid dan tidak menyediakan informasi asli yang benar. Untuk mengatasi masalah ini, PEDOMAN 4 menyarankan agar skema relasi dirancang sedemikian rupa sehingga dapat digabungkan dengan kondisi persamaan pada atribut yang merupakan primary key atau foreign key. Ini akan menjamin bahwa tidak ada tupel palsu yang akan dihasilkan. Selain itu, mencocokkan atribut dalam relasi yang bukan kombinasi (kunci asing, kunci utama) harus dihindari karena dapat menghasilkan tupel palsu.

Functional Dependencies

Ketergantungan fungsional adalah hasil dari keterkaitan antar atribut atau di antara tupel dalam relasi apa pun.

Dalam relasi R, X dan Y adalah dua himpunan bagian dari himpunan atribut. Y dikatakan bergantung secara fungsional pada X jika nilai X yang diberikan (semua atribut dalam X) secara unik menentukan nilai Y (semua atribut dalam Y).

Ini dilambangkan dengan X → Y (Y tergantung pada X). X dalam ketergantungan ini dikenal sebagai penentu ketergantungan fungsional.

Functional Dependency Chart/Diagram

Grafik FD (Functional Dependency) adalah representasi visual dari dependensi fungsional di antara atribut dalam suatu relasi. Langkah-langkah untuk menggambar grafik FD adalah:

1. Cari tahu atribut primary key (atribut yang digunakan sebagai kunci utama dalam relasi tersebut).
2. Buat sebuah persegi panjang dan tuliskan semua atribut primary key di dalamnya.
3. Tulis semua atribut non-prime key (atribut yang bukan kunci utama) di luar persegi panjang.

4. Gunakan tanda panah untuk menunjukkan ketergantungan fungsional antar atribut. Panah ditarik dari atribut yang dipandang sebagai dependen ke atribut yang dipandang sebagai independen.

Jenis Functional Dependencies

There are four major types of FD's

1. Partial Dependency and Fully Functional Dependency
    - Partial dependency : situasi di mana atribut A tidak bergantung pada semua atribut kunci utama dalam relasi. Misalnya, jika relasi memiliki lebih dari satu atribut dalam kunci utama, dan atribut A tidak bergantung pada semua atribut kunci utama, maka ada ketergantungan parsial.
    - Fully functional dependency : situasi di mana atribut A bergantung pada semua atribut kunci utama dalam relasi. Misalnya, jika relasi memiliki atribut kunci utama (RollNo dan Game) dan atribut kunci non-prime (Nilai, Nama, dan Biaya), dan atribut A bergantung pada semua atribut kunci utama, maka A dikatakan dependen fungsional penuh.
2. Transitive Dependency and Non-transitive Dependency
    - Transitive dependency : ketergantungan antara atribut non-prime key dalam suatu relasi. Jika dalam relasi R, $X \rightarrow Y$ (Y tergantung pada X), $Y \rightarrow Z$ (Z tergantung pada Y), maka $X \rightarrow Z$ (Z tergantung pada X). Oleh karena itu, Z dikatakan bergantung secara transitif pada X. Ketergantungan ini dapat menyebabkan redundansi data dan masalah dalam desain database.
    - Non-transitive dependency : setiap ketergantungan fungsional yang tidak transitif. Ketergantungan ini tidak menyebabkan redundansi data dan tidak menimbulkan masalah dalam desain database. Ketergantungan non-transitif diidentifikasi dengan mengevaluasi ketergantungan antara atribut-atribut dalam relasi, dan mengecualikan atribut yang bergantung pada atribut lain secara transitif. Ketergantungan non-transitif merupakan kondisi yang diinginkan dalam desain database yang baik.

3. Single Valued Dependency and Multivalued Dependency
    - Single valued dependency : ketergantungan dalam setiap relasi R, dimana untuk nilai X tertentu, Y hanya memiliki satu nilai. Contohnya, dalam relasi "Pembeli" (BuyerID, BuyerName, BuyerAddress), ketergantungan bernilai tunggal dapat ditentukan antara BuyerID dan BuyerName, karena untuk setiap BuyerID tertentu, hanya ada satu BuyerName yang sesuai.
    - Multivalued dependency (MVD) : ketergantungan dalam setiap relasi R, dimana untuk nilai X tertentu, Y memiliki lebih dari satu nilai. Contohnya, dalam relasi "Pembelian" (BuyerID, ProductID, Quantity), ketergantungan multinilai dapat ditentukan antara BuyerID dan ProductID, karena untuk setiap BuyerID tertentu, dapat ada lebih dari satu ProductID yang sesuai. MVD dilambangkan dengan $X \rightarrow\rightarrow Y$.
4. Trival Dependency and Non-trival Dependency
    - Trival FD : Trival FD : kondisi di mana atribut kedua (Y) dalam dependensi fungsional ($X \rightarrow Y$) merupakan himpunan bagian dari atribut pertama (X). Dalam hal ini, atribut kedua tidak menambahkan informasi baru yang tidak dapat diperoleh dari atribut pertama, sehingga tidak diperlukan untuk dipertahankan dalam relasi.
    - Non-trival FD : kondisi di mana atribut kedua (Y) dalam dependensi fungsional ($X \rightarrow Y$) bukan himpunan bagian dari atribut pertama (X). Dalam hal ini, atribut kedua menambahkan informasi baru yang tidak dapat diperoleh dari atribut pertama, sehingga harus dipertahankan dalam relasi.

Anomali dalam Database Relasional

Anomali dalam Database Relasional adalah masalah yang muncul karena ketergantungan dalam database relasional. Anomali adalah hasil yang tidak diinginkan yang muncul dari modifikasi data.

Ada 3 jenis anomali yang umum terjadi dalam basis data relasional yaitu :

1. Insertion Anomaly

   Masalah yang muncul ketika ingin menambahkan informasi baru dalam relasi apa pun tetapi tidak dapat melakukannya karena beberapa kendala. Contohnya, jika relasi memiliki atribut yang harus diisi dengan data tetapi tidak dapat diisi karena tidak tersedia, ini dapat menyebabkan masalah Insertion Anomaly.

2. Deletion Anomaly

   Masalah yang muncul ketika mencoba menghapus informasi dari relasi apa pun dan ini menyebabkan penghapusan informasi lain yang tidak diinginkan. Contohnya, jika suatu relasi memiliki beberapa atribut yang saling terkait dan menghapus salah satu atribut, maka atribut lain yang terkait juga akan terhapus.

3. Updation Anomaly

   Masalah yang muncul ketika mencoba memperbarui informasi yang ada dalam relasi apa pun dan ini menyebabkan ketidakkonsistenan data. Contohnya, jika memperbarui atribut dalam relasi dan tidak melakukan perubahan yang sesuai pada atribut lain yang terkait, ini akan menyebabkan ketidakkonsistenan data.

Dependencies dan Logical Implications

Dalam konteks desain basis data, dependensi fungsional adalah kendala yang menunjukkan hubungan antara dua atribut atau dua set atribut dalam suatu relasi. Dalam skema relasional R dan sekumpulan dependensi fungsional F, sebuah dependensi fungsional X→Y dianggap tersirat secara logis oleh himpunan dependensi fungsional F jika untuk relasi R pada skema relasional itu, jika memenuhi F juga memenuhi X→Y.

Contohnya, dalam skema relasi R = (A, B, C, D) dan himpunan FD F = {A→B, B→C, C→D}, jika kita mengevaluasi dengan FD di dalam F, maka FD A→C, B→D dan A→D secara logis tersirat, karena dapat di deduksi dari ketergantungan yang ada dalam himpunan F.

Secara singkat, jika sebuah dependensi fungsional X→Y dapat diperoleh dari sekumpulan dependensi fungsional F yang ada, maka dependensi fungsional X→Y tersebut dapat dikatakan tersirat secara logis oleh himpunan dependensi fungsional F.

Armstrong's Axioms

Armstrong's Axioms adalah enam aturan yang digunakan untuk menemukan semua FD yang secara logis ditunjukkan oleh suatu set FD. X, Y, Z, dan W adalah subset atribut dari relasi R. Aturan-aturan ini adalah:

1. Reflektif. Jika Y adalah subset dari X, maka X → Y. Ini juga menyiratkan bahwa X → X selalu terjadi. FD jenis ini disebut FD trivial.

2. Augmentasi. Jika X → Y terjadi dan Z adalah suatu set atribut, maka ZX → ZY.

3. Transitivitas. Jika X → Y terjadi dan Y → Z terjadi, maka X → Z terjadi. Aturan-aturan ini dikatakan Sound dan Complete karena tidak menghasilkan FD yang tidak valid dan memungkinkan untuk menghasilkan F+ (penutupan dari F) dari set FD yang diberikan F. Namun, penggunaan Armstrong's Axioms secara langsung untuk perhitungan F+ cukup rumit dan kompleks. Oleh karena itu, beberapa aturan tambahan ditambahkan untuk mempermudah proses perhitungan F+. Aturan tambahan ini dapat

dibuktikan benar dengan menggunakan Armstrong's Axioms. Aturan tambahan ini adalah:

4. Additivitas atau Union. Jika X → Y dan X → Z, maka X → YZ terjadi.
5. Projectivitas atau Dekomposisi. Jika X → YZ terjadi, maka X → Y dan X → Z juga terjadi.
6. Pseudotransitivitas. Jika X → Y dan ZY → W terjadi, maka XZ → W terjadi.

Penutupan Satu Set Functional Dependencies

F+ adalah set dari semua dependensi fungsional yang diperoleh secara logis dari F. F+ adalah set FD yang dapat diperoleh dari F. F+ juga merupakan set FD yang paling kecil yang mengandung F dan tidak ada FD yang dapat diperoleh dari F dengan menggunakan aksioma yang tidak terkandung dalam F+.

Dengan mengetahui semua dependensi fungsional dalam suatu relasi, kita dapat dengan mudah mengidentifikasi superkey, candidate key, dan determinan lain yang diperlukan untuk normalisasi.

Algorithma ini digunakan untuk menghitung F+ yang merupakan penutup dari FD (Fungsional Dependency). Algorithma ini memiliki input yaitu relasi dengan himpunan FD F dan outputnya adalah penutup dari himpunan FD F. Algorithma ini mengikuti langkah-langkah sebagai berikut:

Step 1 : Inisialisasikan F+ = F, yang merupakan himpunan dari FD yang diberikan

Step 2 : Selama (perubahan pada F+), lakukan

Step 3 : Untuk setiap fungsi dependensi f dalam F+ Aplikasikan aturan reflektivitas dan pengembangan pada f dan tambahkan dependensi fungsional yang dihasilkan ke F+

Step 4 : Untuk setiap pasangan dependensi fungsional f1 dan f2 dalam F+ Aplikasikan aturan transivitas pada f1 dan f2. Jika f1 dan f2 dapat digabungkan, tambahkan FD yang dihasilkan ke F+.

Step 5 : Untuk setiap dependensi fungsional dalam F+ Aplikasikan aturan pengurangan dan pemisahan pada f dan tambahkan dependensi fungsional yang dihasilkan ke F+

Step 6 : untuk setiap pasangan dependensi fungsional f1 dan f2 dalam F+ Aplikasikan aturan Pseudotransitivity pada f1 dan f2. Jika f1 dan f2 dapat digabungkan, tambahkan FD yang dihasilkan ke F+.

Aturan tambahan ini membuat proses menghitung F+ lebih mudah dengan menyederhanakan proses yang digunakan pada langkah 3 dan 4. Jika kita ingin menghitung F+ hanya dengan menggunakan aturan Armstrong, maka hapus langkah 5 dan 6.

Covers

Perhatikan dua set FD F1 dan F2 dari skema relasi R. Kedua set F1 dan F2 dianggap sebanding jika penutup F1 (F1+) sama dengan penutup F2 (F2+). Penutup dari sebuah set FD adalah sekumpulan semua FD yang dapat diinfer dari set FD yang diberikan.

Set F1 mencakup F2 dan F2 mencakup F1 jika dan hanya jika F1 dan F2 sebanding. Ini berarti jika F1 sebanding dengan F2, maka F1 adalah sebuah penutup dari F2 dan F2 adalah sebuah penutup dari F1.

Pentingnya covers adalah untuk mengurangi kompleksitas dari perhitungan penutup dari sebuah set FD. Terkadang penutup dari sebuah set FD F+ dapat sangat besar dan sulit dihitung. Dalam kasus tersebut, penutup digunakan sebagai perwakilan dari penutup F. Dengan menggunakan penutup, lebih mudah untuk menghitung penutup dari sebuah set FD.

Jenis Cover

Redundant Cover: kondisi ketika sebuah himpunan atribut FD (dependensi fungsional) F1 dari suatu skema relasi R dapat digantikan oleh himpunan atribut yang lebih kecil F1'. Dalam hal ini, F1 dianggap sebagai redundant cover karena F1' dapat digunakan untuk mencakup semua atribut dari F1 tanpa ada atribut yang terlewatkan.

Contohnya, jika F1 = {A->B, B->C, A->C} , dan F1' = {A->C}, maka F1 adalah redundant cover karena F1' dapat digunakan untuk mencakup semua atribut dari F1, yaitu {A->B, B->C, A->C}.

Redundant cover dapat menyebabkan redundansi data dan kompleksitas dalam desain basis data, sehingga harus dihindari dengan cara menghilangkan atribut yang tidak diperlukan dari himpunan FD.

Non-redundant Cover: konsep dalam desain basis data yang mengacu pada dua set FD (fungsional dependensi) F1 dan F2 dari skema relasi R. Jika F1 mencakup F2 dan tidak ada subset F1' dari F1 yang mencakup F2, maka set F1 disebut sebagai non-redundant cover. Untuk mendapatkan non-redundant cover, beberapa FD harus dihapus dari F1. Namun, non-redundant cover yang diperoleh tidak unik dan dapat diperoleh lebih dari satu non-redundant cover. Non-redundant cover dengan jumlah FD yang minimum disebut sebagai non-redundant cover minimal.

Mengurangi jumlah fungsi dependensi yang digunakan dalam suatu skema relasi dapat membuat desain basis data lebih sederhana dan mudah dikelola. Namun, harus diingat bahwa menghapus fungsi dependensi dapat mengurangi akurasi dari informasi yang disimpan.

Canonical Cover: konsep yang digunakan dalam normalisasi basis data. Sebelum membahas tentang canonical cover Fc untuk himpunan FD (functional dependency) F, mari kita bahas beberapa istilah yang terkait dengan canonical cover. Simple FD adalah sebuah ketergantungan fungsional X→A1A2A3.....An yang dapat diganti dengan himpunan FD yang setara X→A1, X→A2, X→A3.....X→An dengan menggunakan aksioma additivity dan projectivity. Sebuah FD dalam bentuk X→Ai, di mana sisi kanan hanya memiliki satu atribut disebut sebagai simple FD. Setiap himpunan FD dapat diganti dengan himpunan simple FD yang setara.