مفاهيم قواعد البيانات

Database Concepts and Design

المستوى : الرابع

رمزالمقرر : ٢٢٣ حسب

المتطلبات السابقة : ١٢١ حسب

طبيعة المقرر : ساعتين نظري +ساعتين عملي

المرجع : أصول نظم قواعد البيانات –الجزء الأول

تأليف : أ.د رامز المصري / أ.د شامكانت نافاث

ترجمة د.م خالد ناصر السيد

استاذة المادة

م/ليندا البدري

المحاضرة الحادية عشر

مفاهيم قواعد البيانات المبنية على التقنيات الموجهة الأهداف (الشيئية)

Concepts for Object Database

م/ ليندا البدري

المفاهيم المبنية على التقنيات الموجهة الأهداف(الشيئية)
Overview of Object-Oriented Concepts

- يتكون الكائن (Object) من مكونين : الحالة أو القيمة (state) والسلوك (Behavior) .
- الكائنات في لغات البرمجة ذات التقنيات الموجهة الاهداف (Object-Oriented Programming Languages(OOPL)) تتواجد فقط أثناء تنفيذ البرنامج (كائنات عابرة) ، أما OODB فيمكنها مد تواجد الكائنات عن طريق تخزينها بشكل دائم بحيث يمكن استخراجها بعد ذلك وتسمح بمشاركة البرامج والتطبيقات في تلك الكائنات .

- أحد أهداف قواعد البيانات الموجهة الأهداف هو معالجة التطابق المباشر بين كائنات كل من العالم الواقعي وقواعد البيانات بحيث لا تفقد تكاملها وسلامتها وهويتها مع أمكانية تعريفها و تشغيلها .
- ميزة أخرى في قواعد البيانات الموجهة الاهداف هي أن الكائنات يمكن أن يكون لها بنية ذات تركيب إجباري لازم لكي يحتوي على كافة المعلومات الضرورية التي تصف الكائن . في المقابل في قواعد البيانات التقليدية فإن المعلومات عن الكائن تكون منثورة خلال عدة علاقات .

- تُصر بعض نماذج OO على أن كافة العمليات التي يستطيع المستخدم تطبيقها على كائن ما يجب أن تكون معرفة مسبقاً .مما يفرض عزلاً كاملاً للكائنات . وقد تم تخفيف هذا المنهج الصارم في معظم نماذج البيانات بغرض تسهيل تحديد الإستعلامات .
- لتشجيع عملية العزل يتم تعريف العملية على جزأين : الجزء الأول يسمى إتصال العملية (Interface of operation) ، أو بصمة العملية (Signature of operation) ويحدد أسم العملية والمعاملات (المتغيرات) المستخدمة . أما الجزء الثاني فيسمى الطريقة أو الجسم ويحدد أداة تنفيذ العملية .

- مفهوم اساسي آخر في نظم OO وهو نوع وهرمية الطبقة وكذلك الوراثة .
- يوجد مفهوم آخر في OO وهو التحميل الزائد للمعامل (Operator overloading) الذي يشير الى القدرة على تطبيق المعامل في نماذج مختلفة للكائنات . في هذا الوضع يمكن أن يشير اسم عملية الى أكثر من تنفيذ مختلف أعتماداً على نماذج الكائنات التي تطبق عليها . تسمى هذه الخاصية تعدد أشكال المعامل (Operator polymorphism) .

هوية الكائن Object identity

- يوفر نظام قاعدة بيانات OO هوية وحيدة (unique identity) لكل كائن مستقل مخزن في قاعدة البيانات .
- هذا المحدد الوحيد ينفذ من خلال محدد كائن وحيد ينتجه النظام ((Object identifier(OID) .
- قيمة OID لاتكون مرئية للمستخدم الخارجي ولكن يستخدمها النظام داخلياً لتحديد احاية الكائن ولإنشاء وإدارة روابط بين الكائنات .
- الصفة الرئيسية اللازمة في OID هي الثبات . بمعنى عدم تغير قيمة OID لكائن معين .

- صفة أخرى ل OID هي أن يتم استخدام كل OID مرة واحدة فقط . حتى في حالة حذف كائن من قاعدة البيانات فلا يمنح OID الخاص به لكائن آخر .
- هاتان الصفتان تدلان على ان OID يجب أن لا يعتمد على أي قيم لخصائص الكائن لأن خصائص الكائنات تتغير .
- يعتبر من غير المناسب أن نبني OID على العنوان الفعلي للكائن في المخزن ، مع ذلك بعض الانظمة تستتخدم العنوان الفعلي كـ OID لزيادة كفأة استخراج الكائن .
- في الغالب يستعمل العدد الصحيح الطويل لتمثيل ال OID ، ثم استخدام ال(hash table) للتحويل بين قيمة OID والعنوان الفعلي للكائن .

بنية (تركيب) الكائن Object structure

- في قواعد البيانات OO يمكن إنشاء حالة الكائن المركب(القيمة الحالية) من كائنات أخرى (أو من قيم أخرى) باستخدام بناء نوع معين (type constructor) .
- أحد الاشكال النظامية لتمثيل كائن بأن نرى كل كائن مكون من ثلاثة أشياء هي (i,c,v) حيث i يمثل OID ، و c هو بناء نوع (إشارة لكيفية إنشاء حالة كائن) ، و v هي حالة الكائن الحالية (القيمة الحالية) .
- بناء النوع (type constructor) يمكن أن يكون أحد الانواع الاساسية وهي : العنصر الذرة (atom) ، والصف العلائقي (tuple) ، والمجوعة (set) .

- توجد أنواع أخرى مثل قائمة متصلة (list) ، ومصفوفة (array) ، وحافظة (bag) وتعرف بنماذج التجميع ويحتوي كل منها على مجموعة من الكائنات وتكون مرتبة في ال list ، و ال array ، وغيرمرتبة في الbag .
- البناء الذري (atom) يستخدم لتمثيل العناصر الذرية البسيطة مثل الاعداد الصحيحة و الكسور والعبارات واي عناصر بيانات بسيطة أخرى .

- حالة الكائن v يتم تفسيرها على حسب نوع البناء c . فإذا كانت c=atom فإن v تكون قيمة ذرية من مجال القيم الاساسية .
- وإن كانت c=set فإن v هي مجموعة من محددات الكائنات $\{i_1, i_2, i_3, \ldots\ldots, i_n\}$ وهي مجوعة OIDs لمجوعة من الكائنات من نفس النوع .
- وإن كانت c=tuple فإن v صفاً علاقياً في الشكل $\{a_1: i_1, a_2: i_2, \ldots\ldots, a_n: i_n\}$ حيث إن كل a_j هو اسم خاصية ، وكل i_j بمثابة OIDs .
- وإن كانت c=list فإن v قائمة مرتبة $[i_1, i_2, \ldots\ldots, i_n]$ من OIDs لكائنات من نفس النوع .
- في حال كون c=array فإن v مصفوفة ذات بعد واحد من OIDs.

- الفرق الجوهري بين المصفوفة والقائمة هو أن المصفوفة لها حد أقصى من محددات الكائنات ، اما القائمة فلا حدود لها .
- الفرق الرئيسي بين المجموعة والحافظة هو أن عناصر المجموعة يجب ان تكون متميزة عن بعضها اما الحافظة فيمكن أن تحتوي على عناصر متكررة .
- يُطلق على بناءات النوع set ، و list ، و array و bag اسم انواع مجموعة (انواع كتلة) لتميزها عن الانواع البسيطة وانواع الصف العلائقي .
- الصفة الرئيسية لنوع المجموعة هو ان حالة الكائن تكون مجموعة من الكائنات التي من الممكن ان تكون مرتبة أو غير مرتبة .

أمثلة

تأملي الكائنات التالية :

$O_1 = (i_1, atom, 'Houston')$

$O_2 = (i_2, atom, 'Bellaire')$

$O_3 = (i_3, atom, 'Sugarland')$

$O_4 = (i_4, atom, 5)$

$O_5 = (i_5, atom, 'Research')$

$O_6 = (i_6, atom, '22\text{-}5\text{-}1988')$

$O_7 = (i_7, set, \{i_1, i_2, i_3\})$

$O_8 = (i_8, tuple, <Dname: i_5, Dnumber: i_4, MGR: i_9, Location: i_7, Employees: i_{10}, Project: i_{11} >)$

$O_9 = (i_9, \text{tuple}, \langle \text{Manager}: i_{12}, \text{Manager\_Start\_date}: i_6 \rangle)$

$O_{10} = (i_{10}, \text{set}, \{i_{12}, i_{13}, i_{14}\})$

$O_{11} = (i_{11}, \text{set}, \{i_{15}, i_{16}, i_{17}\})$

$O_{12} = (i_{12}, \text{tuple}, \langle \text{Fname}: i_{18}, \text{Minit}: i_{19}, \text{Lname}: i_{20}, \text{SSN}: i_{21},$

$\ldots\ldots\ldots\ldots\ldots, \text{Salary}: i_{26}, \text{Supervisor}: i_{27}, \text{Dept}: i_8 \rangle)$

تطابق الكائنات مقابل تساويها

- تأملي الكائنات التالية : O_1 ، و O_2 ، و O_3 ، O_4 ، و O_5 ، و O_6 :

$O_1 = (i_1, tuple, < a_1 : i_4, a_2 : i_6 >)$

$O_2 = (i_2, tuple, < a_1 : i_5, a_2 : i_6 >)$

$O_3 = (i_3, tuple, < a_1 : i_4, a_2 : i_6 >)$

$O_4 = (i_4, atom, 10)$

$O_5 = (i_5, atom, 10)$

$O_6 = (i_6, atom, 20)$

- الكائنات O_1 و O_2 لهما حالات متساوية ، نظراً لأن حالاتهما عند المستوى الذري هي نفسها ولكن القيم يتم الوصول اليها خلال كائنات متباينة O_4 و O_5
- حالتا الكائنات O_1 و O_3 متطابقان ، بالرغم من أن الكائنات نفسها غير متطابقة لأن لهما OIDs مختلفة .
- بالرغم من أن حالات الكائنات O_4 و O_5 متساوين ، فإن الكائنين الفعلين O_4 و O_5 غير متطابقين لأن لهما OIDs مختلفين .

بناء النوع Type Constructors

- يمكن باستخدام لغة تعريف الكائنات (object definition language(ODL)) تعريف انواع الكائن لتطبيق قاعدة بيانات معين . كما في الشكل التالي :

```
define type EMPLOYEE
    tuple ( Fname:        string;
            Minit:        char;
            Lname:        string;
            Ssn:          string;
            Birth_date:   DATE;
            Address:      string;
            Sex:          char;
            Salary:       float;
            Supervisor:   EMPLOYEE;
            Dept:         DEPARTMENT;

define type DATE
    tuple ( Year:         integer;
            Month:        integer;
            Day:          integer; );

define type DEPARTMENT
    tuple ( Dname:        string;
            Dnumber:      integer;
            Mgr:          tuple ( Manager:    EMPLOYEE;
                                  Start_date: DATE;     );
            Locations:    set(string);
            Employees:    set(EMPLOYEE);
            Projects      set(PROJECT);   );
```

Specifying the object types EMPLOYEE, DATE, and DEPARTMENT using type constructors.

تحديد سلوك الكائن عبر عمليات الطبقة
specifying object behavior via class operations

- يمكن تطبيق مفاهيم إخفاء المعلومات وعزلها على كائنات قاعدة البيانات .
- الفكرة الرئيسية هي أن نعرف سلوك (behavior)نوع الكائن إعتماداً على العمليات (operations) التي يمكن تطبيقها على كائنات من ذلك النوع .
- يمكن إستخدام بعض العمليات لإنشاء (بناء) أو هدم (حذف) كائنات وربما يستخدم البعض الآخر لاستخراج حالة الكائن أو تطبيق بعض الحسابات .
- الشكل التالي يوضح اضافة عمليات لتعريف كائنات داخل طبقات Classes .

```
define class EMPLOYEE
    type tuple (    Fname:          string;
                    Minit:          char;
                    Lname:          string;
                    Ssn:            string;
                    Birth_date:     DATE;
                    Address:        string;
                    Sex:            char;
                    Salary:         float;
                    Supervisor:     EMPLOYEE;
                    Dept:           DEPARTMENT;   );
    operations      age:            integer;
                    create_emp:     EMPLOYEE;
                    destroy_emp:    boolean;
end EMPLOYEE;

define class DEPARTMENT
    type tuple (    Dname:          string;
                    Dnumber:        integer;
                    Mgr:            tuple (  Manager:      EMPLOYEE;
                                             Start_date:   DATE;    );
                    Locations:      set(string);
                    Employees:      set(EMPLOYEE);
                    Projects        set(PROJECT);    );
    operations      no_of_emps:     integer;
                    create_dept:    DEPARTMENT;
                    destroy_dept:   boolean;
                    assign_emp(e:  EMPLOYEE): boolean;
                    (* adds an employee to the department *)
                    remove_emp(e: EMPLOYEE): boolean;
                    (* removes an employee from the department *)
end DEPARTMENT;
```

Adding operations to the definitions of EMPLOYEE and DEPARTMENT.

- بالنسبة لتطبيقات قاعدة البيانات فأن المتطلب بعزل الكائنات بالكامل يبدو صارماً جداً . أحد الطرق لتخفيف هذا المتطلب هو أن نقسم تركيب كائن الى خصائص مرئية (Visible) ، واخرى مخفية (hidden) .
- يستخدم المصطلح طبقة (class) للأشارة الى تعريف نوع كيان ما ، مع تعاريف العمليات لذلك النوع .
- يتم تطبيق العملية على كائن ما باستخدام ترميز النقطة (dot notation) . على سبيل المثال ، إذا كانت d تشير الى كائن القسم department ، يمكننا وضع عملية ما مثل no_of_emps موضع التنفيذ بكتابة d.no_of_emps

THE END