OBJECT DATABASE SYSTEMS

HTCSDL QH chỉ cho phép lưu 1 số loại dữ liệu: (kiểu số, ngày, chuỗi), những kiểu dũ liệu này chỉ phù hợp cho các loại ứng dụng cũ ( vd các loại phần mềm xử lý dữ liệu hành chính) . tuy nhiên có nhiều ứng dụng hiện nay phải lưu trữ các loại dữ liệu phúc tạp để xử lý. Điển hình như dữ liệu được lưu trong HDH hoặc những cấu trúc dữ liệu đặc biệt, sâu hơn nữa là dữ liệu trong DBMS. Ví dụ: hệ thống máy tính hỗ trợ cho việc thiết kế và chế tạo( CAD/CAM), đa truyền thông, và quản lý tài liệu, hồ sơ…

Khi lượng dữ liêu tăng lên thì, nhiều tính năng được đưa ra bởi các hqt là rất cần thiết - vd: giảm thời gian khai triển ứng dụng, đồng thời kiểm soát và phục hồi, cơ chí chỉ mục, khả năng truy vấn. trong đó, khái niệm hướng đối tượng có ảnh hưởng lớn phục vu cho dữ liệu phức tạp và chỉ hướng phát triển các hệ thống hướng đối tượng, là vấn để mà chương này ta bàn luận.

ODS được phát triển theo 2 hướng sau:

* Object-oriented database systems: OODS dc đề xuất như 1 thay thế cho hệ thống quan hệ và nhằm vào các loại ứng dụng mà ở đó các đối tượng phức tạp đóng vai trò trung tâm.
* Object-relational (liên quan) database systems: ORDS có thể đc xem là loại mở rộng của Relational db sys với các hàm cần thiết phục vụ cho 1 lớp rộng của ứng dụng và cung cấp nhiều cách chuyển đổi giữa 2 loại này.

Chúng ta dùng các lược dành sau: RDBMS là relational database management systems, object-oriented database management systems là OODBMS, và object-relational database management systems(ORDBMS). Trong chương này chi chỉ bàn về ORDBMSs và nhấn mạnh làm sao có thể xem chúng như 1 phát triển của RDBMSs, sâu hơn nữa là xem như 1 hệ biến hoá khác nhau một cách hoàn toàn.

Phiên bản chuẩn SQL:1999 dựa trên mô hình ORDBMS, hơn nữa OODBMS

Orient hỗ trợ các loại dữ liệu mới phức tạp cho các ứng dụng hiện nay. Các loại dữ liệu phức tạp này điển chỉnh được lưu trong file systems OS hoặc lưu trong các cấu trúc dữ liệu, sâu hơn nữa là ở trong DBMS.

Object-database systems dc phát triển theo 2 hướng khác nhau:

* Object-oriented database systems: dc đề xuất như 1 sự thay thế cho relational systems và nhằm vào các loại ứng dụng mà các đối tượng đóng vai trò trung tâm. Cái phương pháp này bị ảnh hượng nhiều từ ngôn ngôn ngữ lập trình hướng đối tượng và có thể dc hiểu như 1 sự nỗ lực để làm tăng các hàm cho một môi trường ngôn ngữ lập trình.
* Object-relational database systems: là hệ mở rộng của relational db sys với các hàm cần thiết để phục vụ cho 1 lớp rộng của các ứng dụng và cug cấp cầu trung gian giữa hệ relational và hệ object-oriented.

Trong chương này ta bàn về ORDBMS và nhấn mạnh làm thế nào chúng có thể dc xem như 1 sự phát triển của RDBMS, chứ không phải là 1 hệ khác biệt hoàn toàn.

Chuẩn sql:1999 dựa trên mô hình ordbms chứ không pahỉ trên OODBMS, chuẩn chứa các tính năng phục vụ cho nhiều loại dữ liệu phức tập sẽ dc bàn trong chương này. Chúng ta tập trung khai thác khái niệm cơ bản chứ khong phải trình bày về SQL:1999. Có 1 số tính năng ta bàn có thể không có trong sql:1999. Chúng ta sẽ cố gắng phụ hợp với ký hiệu trong sql:1999.

Khái niệm mà ta bạn luận dưới đây thì nó dùng chung cho cả ORDBMS và OODBMS,

Đa số các hãng quản trị RDBMS đều tích hợp cả ORDBMS, vậy sự tồn tại kiến thức về thiết kế và thực thi của Relational db có thể dc thừa hưởng để giải quyết các mở rọng ORDBMS. Việc hiệu các thách thức và cơ hội mà các mở rộng này trình bày cho ng dùng db, ng thiết kế, ng thực thi là điều quan trọng.

Trong chương này, phần 25.1 đén 25.5 làm động cơ thúc đẩy và giới thiệu khái niệm object-oriented. Các khái niệm bàn trong các phần này là chung cho cả OODBMS và ORDMS, thậm chí cú pháp cũng giống với SQL:1999. Ta bắt đầu ví dụ ở 25.1 minh hoạ vì sao các phần mở rộng mô hình quan hệ là cần thiết đề đối phó với 1 số lĩnh vực ứng dụng mới. ví dụ này dc dùng xuyên suốt chương. Ta bàn luận làm thế nào các kiểu dữ liệu trừu tượng dc dịnh nghĩa và vận hành như thế nào trong 25.2 và các type có thể dc tạo thành các kiểu structured ntn trong 25.3. bàn về objects và object identity trong 25.4 và kế thùa và laoị kế thừa trong phần 25.5.

Bàn về làm thế nào để lấy lợi ích của khái niệm mới object-orientd vào thiết kết dữ liệu ordbms trong 25.6. trong 25.7, ta bàn 1 vài sự giải quết các thách thức dc đề ra bởi object-relational sys. Ta bàn ODL và OQL là các chuẩn cho OODBMS, trong 25.8 và so sánh ngắn gọn ORDBMS và OODBMS trong 25.9.

**25.1 ví dụ tạo động lực:**

Ngày nay các công ty như ngành giải trí tài sản của họ k phải là vật hữu hình mà vd như là video và audio.

Ví dụ công ty Dinky giải trí , 1 tập đoàn lớn của Hollywood mà tài sản chính của cty là tập các nvật hoạt hình, đặt biết là nv nổi tiếng Herbert the worm. Dinky có số lượng lớn phim Herbert the worm, chứa nhiều hình âm thanh, video, hànhdđộng. csdl của Dynky dc dùng đ36 quản lý việc bán và cho thuê các dữ liệu liênquan đến Herbert, vd: video, audio.

**25.1.1 kiểu dữ liệu mới**

* User-defined abstract data types: kiểu dữ liệu mới như video, audio, và ta cần lưu chúng vào csdl. Và cần các hàm đặc biệt để thac tác với các đối tượng này. Ví dụ tạo ra phiên bản nén hình.
* Structured types: trong ứng dụng này, ta cần nhiều xây dựng nhiều loại dữ liệu mới từ các loại nguyên tố sử dụng constructors để tạo tập bộ, bộ, mảng, …(25.3)
* Inheritance: số lượng loại dữ liệu tăng, rất khó để nhận ra sự tương đồng giữa các loại khác nhau và để tận dụng lợi ích của nó. Ví dụ: ảnh nén và ảnh có độ phân giải thấp một mức đọ nào đó nó là những tấm ảnh. Vì thế kế thừa các tính năng từ đối tượng ảnh trong quá trình định nghĩa compressed image object và lower-resolution image objects.(25.5)

Làm sao ta có thể định vị dc các vấn đề này trong RDBMS? Trong relational sys Ta có thể lưu images, videos,… bằng BLOBs, nó là 1 chuỗi các byte và hqt hỗ trợ kiểu này giúp cho ng dùng k cần phải bạn tâm về kích thước của BLOBs. BLOB có thể chiếm nhiều trang, k giống như những thuộc tính thường. các quá trình sâu hơn của BLOB phải dc hoàn thành bởi chương trinh ứng dụng của ng dùng, trong ngôn ngữ mà SQL code nhúng vào (thuộc tính của loai LOB trong sql server: text, ntext(unicode), image(binarydata)), nvarchar(max) unicode, giá trị lob là con trỏ trỏ tới dữliệu lob. Nhược điểm khi truy cập nó lấy ra toàn bộ giá trị mà mình k dc chọn lọc lấy những giá trị nào. Điều này nó không thể hiện tính nhất quán dữ liệu, vì ngữ nghĩa của data này nó phụ thuộc nhiều vào host language của ứng dụng và không đc quản lý bởi dbms

25.1.2 thao tác với các loại dữ liệu mới

Ta sẽ diễn tả các lại dử liệu mới mà cần lưu trong Dinky db. Ta không nói bất cứ điều gi vể cách dùng các loại dữ liệu này trong queries ntn, ta học 2 câu queries mà db Dinky cần có. Cú pháp của nó thì chưa phê bình; nó đủ để hiểu diễn tả ý nghĩa. Ta quay trờ lại.

Yêu cầu từ công ti ngũ cốc clog cần thuê hình Herbert the worm đứng ở trước mặt trời. câu query diễn tả lấy ra 1 tập các images và giá của mỗi images đucợ cú pháp sql như hình 25.2. dinky có các phương thức dạng ngôn ngữ java và dc ghi vào db sys. NHỬNG phương thức này dc dùng trong truy vấn bằng cách. Phương thức thumbnail tạo ra 1 mẫu image kích thước nhỏ với image của nó.

Yêu cầu thứ 2 của ng điều hành Dinky muốn nhiều phim Herbert đc chiếu trong vùng Andorra khi ngũ cốc dc đem ra bán ở đây. Để kiểm tra tình trạng vấn đề, ng điều hành muốn tìm tên của tất cả các rạp phim chiếu phim Herbert trong vồng 100K ở Andorra. Câu query như sau:

25.3

ảnh (ảnh, muc)

categores(muc, gia, )

type theater)\_t AS (tno integer, ten, dia chi, dt)

theaters of theather\_t;

nowshowing (pjim, theater ref(theater\_t) with scope theaters, ngày bắt đầu, kết thức)

phim(stt, nhan de, dien vien, dao dien, budget)

countries(

diễn giải cho hình 25.3: thuoc tinh theater trong nowshowing là 1 tham chiếu đến 1 đối tượng của bảng khác. Đối tượng tham chiếu này cho phép ký hiệu viết tặt: N.theater🡪name và N.theater🡪address, mỗi ký hiệu trỏ này chỉ đến thuộc tính của đối tượng theater\_t mà dc tham chiếu trong bộ N trong Nowshowing

start: diễn viên chính . phương thức radus trả về 1 vòng trong có tâm là đối số đầu tiên và bán kính là đối số thứ 2. Overlaps: kiem tra phạm vi không gian(boundary) của Andorra có chồng cắt với vòng tròn mà radius tạo ra không.

* Có equijoin: N.film = F.filmno
* Có phuong thuc overlap ket 2 bảng theater voi vung chieu phim

**Vấn đề:**2 câu lệnh thuộc ob relational vừa rồi có vài tính năng vượt trội:

* Ng dùng định nghĩa phương thức: loại dữ liệu trừu tượng ng dùng định nghỉa dc thao tác qua các phương thức, vd phương thức is\_herbert.
* Các toán tử cho các loại cấu trúc: ngoài toán tử như dữ liệu thường có sẳn, ordbms còn cung cấp các toán tử cho các loại đó: vd: loại setof có các toán tử: thuộc, con, giao…).
* Toán tử cho các loại tham chiếu: 🡪

🡪tóm lại: relational sys hạn chế sự linh hoạt các loại dữ liệu có sẵn. loại dữ liệu trong relational: interger, string…) nhưng trong ob re sys có thêm : loại dữ liệu trừu tượng do ng dùng định nghĩa (jpeg, polygon, ..), loại cấu trúc (row (n1, t1, nn, tn), listof(base)…), loại tham chiếu ????????????

**25.2 user-defined abstract data types**

**25.2.1**

**25.3 structured types.**

**25.3.1 vận dụng data of structured types.**

Objects and oids: mỗi bộ trong 1 bảng có thể dc đặt 1 oid bằng cách định nghĩa bảng như 1 loại cấu trúc, ví dụ Trong định nghĩa bảng Theaters trong. Ngược lại với bảng countries, các bộ trong bảng k có các oid. REF giá trị của nó là oids, REF phải có giá trị thuộc 1 loại structured type. Theater of type ref(theater\_t) -🡪 cột theater tham chiếu đến đối tượng có loại theater\_t, đối tượng đó là các cột trong bảng Theaters

Đầu tiên phại tạo loại cấu trúc bảng: theater\_t, trong đó có thể có các thuộc tính tham chiếu (chỉ rõ là tham chiếu này loại gì).

Dựa vào cấu trúc bảng đó tạo bảng theaters (bảng theaters này se có các bộ dc gọi là các đối tượng), và bảng theaters này phải có thuộc tính oid để xác định duy nhất đối tượng (hàng) trong bảng. khi tạo bảng cũng có thể tạo thuộc tính tham chiếu dc ( khi đó chỉ rõ luôn là tham chiếu vô bảng nào)

**25.4 objects, object identity, reference types**

Trong hqt csdl hướng đối tươgn5, đối tượng dữ liệu có thể dc gán 1 giá trị object identifier (oid), xác định duy nhất trong dữ liệu trong suốt thời gian sống của nó. DBMS chịu trách nhiệm tạo ra giá trị này và phải đảm bảo nó luôn xd dc 1 đối tượng duy nhất trong suốt thời gian sống. ở 1 số hqt tất cả các bộ dc lưu trong bất cứ bảng nào thì cũng đều là đối tượng và he qt sẽ tự đống đánh cho chúng 1 giá trị oid. ; 1 số hqt khách, ng dùng có thể chỉ ra các bảng nào có oid.

1 oid của đối tượng dc dùng để chỉ tới đối tượng đó mà có dữ liệu dc lưu ở nơi khác. Nó như 1 giá con trỏ tham chiếu có kiểu là loại mà nó tham chiếu đến. ref(base) là 1 hàm dựng kiểu là con trỏ trỏ đến object kiểu base. Ref có thể dc chèn vào trong hàm dựng khác.

**So sánh url với oid:**

|  |  |
| --- | --- |
| Oid | url |
| Xác định duy nhất 1 đối tượng trong suốt thời gian | Tài nguyên web dc trỏ bởi 1 URL có thể thay đổi theo thời gian |
| đơn giản, nó không mang ý nghĩa vật lý ( không chứa thông tin vật lý của đối tượng mà nó trỏ đến)- dẫn đến nếu thay đổi địa chỉ lưu trữ của đối tượng thì ta khộng phải sửa oid này. | Chứa cả địa chỉ network và tên file system, khi tài nguyên mà url đó trỏ đến thay đổi địa chỉ lưu hoặc địa chỉ network, đường dẫn url đến tài nguyên đó bị sai hoặc phải yêu cầu cơ chế forwarding. |
| Oid dc phát sinh tự động bởi DBMS | Ng dùng tạo ra |

**25.4.1 notions of equality (khái niệm đẳng thức).**

Sự khác biệt giửa các loại tham chiếu với các loại cấu trúc tham chiếu tự do (reference-free structured types) làm nảy sinh 1 vấn đề khác: định nghĩa đẳng thức. 2 object có cùng loại được xem là deep equal khi và chỉ khi:

* Các Đối tượng là kiểu nguyên tử và có cùng giá trị hoặc
* Các đối tượng là kiểu tham chiếu và 2 đối tượng mà chúng tham chiếu đến phải thoả deep equal
* Đối tượng là loại cấu trúc và các thành phần tương ứng trong các đ1ôi tượng phải thoả deep equal.

2 đối tượng có cùng loại tham chiếu dc định nghĩa là shallow equal nếu cả 2 chỉ đến cùng đối tượng. định nghĩa shallow equality có thể dc mở rộng cho đối tượng có loại tuỳ ý khác bằng cách lấy định nghĩa của deep equality và thay deep equals thành shallow equals.

2 đối tượng thoả shallow equal thì thoả deep equal, ngược lại không đúng

**25.4.2 dereferencing reference types**

Loại tham chiếu ref(foo) không giống như loại foo nó trỏ đến. để truy cập foo dc tham chiếu, có phương thức deref() cung cấp song song với hàm dựng loại ref. giống như trong ng lập trình def() = \*a. ví dụ cần lấy 1 bộ trong bảng Nowhowing, lấy giá trị name của theater: Noshowing.deref(theater).name. vì tham chiếu đến các loại bộ thì giống nhau, nên 1 số hqt cung cập 1 toán tử mũi tên kết hợp với hậu tố của toán tử dereference operator . đó là: Nowshowing.theater🡪name.

**25.5 kế thừa**

**25.5.1 định nghĩa các loại kế thừa**

Trong csdl Dinky, ta mô hình hoá rạp chiếu phim thành kiểu theater\_t. nhưng Dinky muốn kinh doanh thêm loại rạp chiếu-café, rapchiếu-café sẽ cung cấp pizza và các laoị thức ăn uống khi xem phim. Vì vậy trong db cần hỗ trợ lưu trữ các thông tin này, theater-café giống như theater nhưng có thêm các loại thuộc tính thực đơn. T dùng kế thừa, đoạn lệnh DDL sau diễn tả điều đó:

Create type theatercafe\_t under theater\_t (menu text):

Trong lệnh này tạo ra theatcafe và tất cả các thuộc tính và phương thức của theater\_t đều dc theatercafe dùng lại nhưng điểu ngược lại thì k. ta nói theatercafe\_t kế thừa các thuộc tính của theater\_t.

**Lưu ý** cơ chế kế thừa k phải là macro làm ngắn gọn lệnh create. Nó tạo ra quan hệ rõ ràng trong dữ liệu giữa subtype (theatercafe\_t) và supertype (theater\_t): 1 object của subtype ccũng dc xét là 1 object của supertype.

The substitution Principle: cho 1 supertype A và 1 subtype B, luôn luôn có thể thay thế 1 đối tượng loại B thành 1 biểu diễn hợp lý được viết bởi object laoị A, ngoại trừ việc tạo ra type errors.

Note: kế thừa cũng có thể dc dùng cho kiểu nguyên tử, cho cả kiểu row.

**25.5.2 cớ chế binding phương thức**

Ta có thể viết lại phương thức của supertype trong subtype. Viết 1 phương thức mới với phương thức cũ gọi là overloading tên phương thức.

Do việc overloading này, hệ thống phải hiểu phương thức nào dc dành cho 1 biểu diễn cụ thể. Vd, khi sys cần gọi pthức display() trong object kiểu jpeg\_image\_t, nó dùng phương thức display riêng biệt. khi cần vọi display cảu image\_t (k phải loại subtyped) thì n1o gọi pthuc display chuẩn. quá trình (process) quyết định phương thức nào dc gọi lên dc gọi là liên kết phương thức với đối tượng (binding to the object). Binding dc thực hiện khi khi đang phân tích ngữ pháp lệnh (early binding), binding dc thực hiện lúc chạy chương trình(late binding)

Early binding : Khi quá trình binding được thực thi trước khi chương trình chạy(by compiler & linker). Vấn đề chính của early binding là compiler không biết chính xác method nào để gọi khi chỉ có một tham khảo tới đối tượng Instrument. C compilers chỉ có một lời gọi method là early binding.

- late binding : ra đời để giải quyết vấn đề của early binding, nó sẽ binding ở thời điểm chạy, nó có cơ chế xác định đối tượng nào ngay tại thời điểm gọi và gọi method của đối tượng đó.

- Late binding là cái core của Tính đa hình (Polymorphism). Nhờ late binding mà tính đa hình trở thành công cụ mạnh mẽ trong Lập trình hướng đối tượng.

**25.5.3 tập phân cấp, phạm vi kiểu, và query**

Có 1 chút khác biệt giữa kế thừa ta bàn vói ngngữ lt hdt.

tuy nhiên vì hqt object-rela cung ấp ngôn gnữ truy vấn trên tập dữ liệu dạng bảng,nên các cơ chế từ các ngôn gnữ lập trình dc tăng cường trong csdl đối tượng để xử lý với bảng và truy vấn tốt. một cách riêng, trong ob-relational sys ta ca thể định nghĩa 1 bảng chứa cá các đối tượng có kiểu cụ thể, như bảng theater trong lược đồ Dinky. Cho 1 subtype như theater-café, nếu mốn tạo bảng theater-café khác để lưu thong tin về theater cafes. Nhưng khi viết 1 truy vấn trên bảng theaters, đôi khi muốn 1 câu truy vấn tượng tự trong bảng theate-cafes, sau hết, nếu ta chiếu các cột, 1 ví dụ của bảng theater-café có thể dc quan tam như 1 ví dụ của bảng.

không phải yêu cầu ng dùng chỉ rõ 1 câu query riêng lẽ cho mỗi bảng, ta có thể thông báo cho hệ thống là 1 bảng mới thuộc subtype dc xem như 1 phần của bảng supertype:

create table theater\_cafes of type theater\_cafe\_t under theaters;

truy vấn này nói cho sys biế các lệnh query trên bảng theaters phải dc chạy trong các bộ trên cả bảng theaters và theater\_cafes. Nếu subtype definition gọi phương thức overlaoding, thì late-binding dc dùng để chắc ràng phước thức đó dc gọi cho các bộ hợp lý.

Nhìn chung, mệnh để under dc dùng để tạo 1 cây tuỳ ý của bảng gọi 1 tập phân cấp. lệnh query trên 1 bảng riêng biệt T trong phân cấp dc chạy trên tất cả bộ trong T và các node con của nó. Đôi khi, ng dùng có thể muốn truy vấn chạy chỉ trên T, và k trên các con, thì thêm cú pháp only.

1 số sys tự động tạo các bảng đạc biệt cho mỗi kiểu, mà chứa tham chiếu đến các instance của kiểu dữ liệu đó mà tồn tại trong db ( ở bất kỳ bảng nào thuộc kiểu dữ liệu đó). Các bảng này dc gọi là **type extents** và cho phép truy vấn trên tất cả đối tượng của kiểu đó, k kể đến nơi các đối tượng cư trú trong db. Type extents hình thành 1 collection hierarchy mà song song với kiểu phân cấp.