Лекція №13 ПОіП « КОНСТРУЮВАННЯ ОСНОВНИХ БЛОКІВ ПАКЕТІВ»

(Модуль 2 -)

План лекції.

Загальні питання конструювання пакетів

Конструювання пакета прикладних програм (ППП) ніяким чином не порушує загальних правил технології проектування прикладного програмного забезпечення, але за своєю специфікою універсального програмного виробу має певні правила побудови, які й вивчаються в межах цієї теми. Нагадаємо, що конструювання пакета здійснюється на стадії технічного та робочого проектування, початковими даними для виконання яких є вимоги, викладені у технічному завданні щодо задач предметної області, апаратно-програмного середовища майбутнього пакета, категорії потенційних користувачів пакета, умов використання пакета тощо, а також результати зовнішнього проектування, що визначають зміст та характер інформаційного середовища пакета.

Основні правила побудови пакетів стосуються базових засобів системного наповнення пакета, які можуть настроюватися на конкретні засоби зовнішнього керування та конкретні моделі предметної області. Для настроювання ППП на конкретну предметну область необхідно додати до оболонки пакета опис інформаційної бази, опис функціональних зв’язків, а також підключити оброблювальні модулі.

Інтерфейс користувача реалізує одну з важливих функцій керуючої програми, а саме: забезпечення взаємодії між людиною та комп’ютером. Розрізняють два типи двостороннього обміну інформацією між людиною та комп’ютером: діалог, що керується системою (ініційоване спілкування), діалог, що керується користувачем (директивне спілкування), а також мішане спілкування (поєднання двох типів діалогу), яке є найбільш поширеною формою організації взаємодії.

Залежно від побудови пакета він може забезпечувати за одне звернення до нього розв’язання тільки однієї задачі або потоку задач предметної сфери. І хоча реалізація пакетів, що не забезпечують обробку потоку задач, безумовно, простіша, проте не можна забувати, що в цьому разі їх використання буде не дуже зручним. З цією особливістю пов’язане питання про організацію вхідного потоку пакета. Сукупність початкових даних, що входять до складу вхідного потоку пакета, складається з керуючих директив користувача щодо послідовності задач, керуючих параметрів для

виконання кожної із задач, а також даних, що подаються на обробку. Саме тому за характером даних, що вводяться, та інформації, що формується пакетом, можна виділити чотири групи функцій інтерфейсу: функції щодо керування, функції за даними, довідкові функції; інформаційні функції. Ці групи функцій можуть бути реалізовані послідовно або паралельно.

Для реалізації вимог користувача щодо задачі, яка розв'язується за допомогою пакета, незалежно від форми подання керуючих директив і параметрів (рівня вхідної мови), у структурі керуючої програми повинні бути модулі формування поточного стану та управління викликом оброблювальних модулів. За призначенням вони дістали назву керуючих модулів.

Засоби зовнішнього інтерфейсу мають забезпечити використання даних, що зберігаються на зовнішніх носіях, або зберігання даних, що є результатом обробки, тобто забезпечити реалізацію функції керуючої програми пакета щодо організації обміну даними. Зауважимо, що конкретні вимоги до зовнішнього інтерфейсу визначаються на основі аналізу передбачених умов використання пакета та обсягів даних, що обробляються. Слід пам'ятати про необхідність створення зручних умов роботи користувача під час ведення інформаційної бази. Для цього потрібно передбачувати можливість перегляду оперативної бази даних, нагромаджених даних, що використовуються в розрахунках, проміжних та остаточних результатів, а також можливість їх зміни.

Комунікативні і мовні можливості засобів спілкування

Загальний опис засобів спілкування

Розв'язання проблеми спілкування кінцевих користувачів із СОД припускає розробку засобів спілкування, що задовольняють потреби масових користувачів на всіх стадіях існування СОД. У термінах розповсюдженої класифікації кінцевих користувачів (КК), масовим користувачем СОД є не програмуючий кінцевий користувач. Таким чином, рішення проблеми спілкування КК із СОД вимагає створення для не програмуючого КК таких засобів (систем), що дозволять йому здійснювати ефективну взаємодію із СОД на стадіях розробки, використання і розвитку.

Ефективність процесу взаємодії користувачів з пакетами вимагає його наближення до процесу взаємодії, що існує між людьми. Тому багато особливостей засобів спілкування запозичені з аналізу діалогів між людьми.

Засоби спілкування складаються з наступних компонентів: діалогового (комунікативного), мовного (розуміння і генерація) і обробного. Обробні можливості поділяються так:

- непроцедурні, тобто можливості з автоматичного формування рішення вхідної задачі;
- можливості набуття знань;
- пояснювальні можливості;
- можливості, передбачені задачами, збереженими в бібліотеці прикладних програм.

Засоби спілкування найбільш точно відповідають узагальненій схемі, що описує взаємодія користувача із СОД.

У залежності від поточного стану діалогу, відображуваного контекстом, і від типу завдання діалоговий компонент формує порядок роботи системи, тобто визначає порядок виклику компонентів.

Комунікативні можливості засобів характеризуються тим, що взаємодія з користувачем здійснюється відповідно до гнучкої діалогової структури з використанням безлічі стратегій, що забезпечують надійне одержання і правильну обробку завдань користувача (тобто обробку, що відповідає цілям користувача). Розвинуті комунікативні можливості засобів спілкування забезпечують гнучкий розподіл функцій між учасниками взаємодії, що гарантує виявлення і задоволення інформаційних потреб кінцевих користувачів.

Мовні можливості припускають, що спілкування здійснюється звичною для користувача мовою, наприклад природною мовою, у якому допускаються лексичні і граматичні помилки, можливе використання не тільки повних, але і фрагментарних висловлень (наприклад, окремих іменних груп). Надання користувачу можливостей використовувати звичні способи взаємодії забезпечує природність і ефективність процесу взаємодії. Користувач уникає необхідності чіткого дотримання синтаксису мови спілкування (що призводило до великої кількості невдач), і це дозволяє йому приділяти основну увагу семантиці висловлень.

Непроцедурні можливості припускають, що користувач при спілкуванні із системою визначає тільки, яку задачу він хоче розв’язати, не визначаючи способу розв’язання даної задачі. Введення непроцедурності дає можливість не кінцевому користувачеві, який програмує, взаємодіяти із СОД на стадіях розробки і розвитку, тобто брати участь не тільки у використанні, але й у розробці програм. Тлумачні можливості засобів спілкування припускають здатність до пояснення того, що система чи має чи не має робити, що вона робить, і чому. Тлумачення використовуються для пояснення того, як було здобуто результат, і для виявлення причин, що перешкоджають розв’язанню поставленої задачі. У засобах спілкування першого і другого поколінь пояснювальні можливості обмежувалися видачею стандартних повідомлень про помилки, передбачених при розробці системи, і статичними поясненнями можливостей системи і використовуваного нею способу подання знань (у рамках можливостей словника-довідника даних). Розвиток тлумачних властивостей засобів спілкування підвищує довіра кінцевих користувачів до системи, сприяє виявленню й усуненню непорозумінь, що виникають у процесі взаємодії.

Можливості зі здобуття знань припускають набуття нових і модифікацію старих знань у режимі безпосередньої взаємодії з кінцевим користувачем. Введення в засоби спілкування можливостей із набуття знань дозволяє системі налаштовуватися на зміни як проблемної сфери, так і інформаційних потреб користувачів.

Ближче інших до вимог, пропонованих до систем спілкування, підходять експертні системи і системи, що взаємодіють з базами даних (БД) природною мовою (ПМ). В останніх найбільш детально досліджені питання створення діалогового і мовного компонентів, однак їх не можна віднести до систем з таких причин:

1) вони є експериментальними, а не промисловими;

2) вони приділяють основну увагу питанням діалогу і мови на стадії використання системи, зневажаючи питанням обробки повідомлень, надзвичайно важливим для стадій розробки і розвитку.

У свою чергу, експертні системи подібні засобам спілкування в наступному:

1) вони є єдиними з існуючих систем, що містять всі основні компоненти засобів спілкування (діалоговий, мовний, що вирішує, що здобуває, і пояснювальний);

2) вони розробляються, використовуються і розвиваються кінцевими користувачами, що не знають програмування. Більш того, багато експертних систем є промисловими і перебувають в експлуатації протягом багатьох років.

Комунікативні можливості засобів спілкування

У процесі спілкування користувачів із СОД неминуче виникають різного роду локальні утруднення (помилки, нерозуміння, недовіра тощо). Більш ранні засоби спілкування, по суті, ігнорували наявність локальних утруднень, що істотно обмежувало сфери прикладних цих програм, чи прагнули передбачити всі можливі локальні утруднення. Виникнення непередбачених ситуацій, як правило, призводить до глобальних невдач.

Діалог можна розглядати, принаймні, на трьох рівнях:

1) загальна (глобальна) структура діалогу, що характеризує його тип;

2) тематична структура, що відбиває структуру задачі, розв'язуваної в поточному діалозі даного типу;

3) структура кроку діалогу (локальна структура), що відбиває взаємодію учасників в елементарному акті діалогу.

Основні відмінності сучасних засобів спілкування від більш ранніх:

1) використання для завдання тематичної структури діалогу незмінних сценаріїв і сценаріїв, що генерує система в процесі формування рішення задачі;

2) забезпечення великої гнучкості і лаконічності спілкування за рахунок більшої волі (перехоплення ініціативи, перевірка правильності інтерпретації висловлень користувача), що дозволяє, не перериваючи поточний сценарій, повідомити користувачеві про свої утруднення.

Можливості засобів спілкування через інтерпретацію повідомлень

На етапі інтерпретації повідомлень користувача розв'язуються дві основні задачі:

1) буквальна інтерпретація повідомлення (висловлення) у контексті діалогу;

2) інтерпретація повідомлення за метою учасників спілкування.

Завдання буквальної інтерпретації полягає в тому, щоб, з огляду на контекст, ідентифікувати сутність висловлення і його предметну сферу. Друге завдання інтерпретації полягає в тім, щоб визначити, як повідомлення погодиться з цілями учасників. Обидві задачі (особливо друга) є складними і до кінця не вивченими. Одним з підходів розв'язування таких задач є використання проблемних сфер.

Проблемна сфера — це сукупність предметної сфери і безлічі задач, здійснених над сутностями з предметної сфери. До простих проблемних сфер відносяться різні предметні сфери, сутність яких становлять прості задачі інформаційного обслуговування: довідкові задачі про погоду, про товари, про

літературу і т. ін.; задачі резервування місць, квитків, товарів тощо. Необхідність розв'язання простих задач інформаційного обслуговування виникає практично в будь-якої людини як на виробництві, так і в побуті.

В даний час розроблені системи, що взаємодіють з кінцевими користувачами природною мовою при розв'язуванні простих задач інформаційного обслуговування. Однак це не привело до масового створення подібних систем, хоча потреба в таких системах загальновідома. Засоби спілкування зазначених систем мають недоліки, що утруднюють їх широке практичне використання. У них відсутні:

- можливість обробки неграматичних конструкцій;
- пояснювальні властивості;
- здатності здобувати нові знання.

Вище було відзначено, що у випадку простих задач інформаційного обслуговування проблема спілкування кінцевих користувачів із СОД значно спрощується.

Так, при замовленні залізничних квитків параметри обслуговування приблизно такі: станція відправлення, станція призначення, дата відправлення, кількість місць, час відправлення (при наявності декількох потягів), час прибуття і т.п. У запобігання непорозумінь підкреслимо, що повсюдно використовувані системи резервування квитків не є в нашій термінології системами, що здійснюють безпосередню взаємодію з кінцевим користувачем, тому що в цих системах кінцевий користувач взаємодіє з касиром (який спілкується із системою незрозумілою кінцевому користувачу мовою), а не із системою.

«Прозорість» структури простих задач інформаційного обслуговування дозволяє використовувати для їх опису фрейми. ***Фрейм*** є поданням деякої сутності в термінах її компонентів (складових). Так, фізичний об'єкт можна подати в термінах його частин, явище може бути представлене в термінах його чи учасників більш детально в термінах під'явищ, що складають вихідну подію. Компоненти фрейму називаються ***слотами***. Структура простих задач інформаційного обслуговування природним способом відповідає структурі фрейму. При цьому параметри обслуговування відповідають слотам. Робота системи під час обслуговування користувача полягає в заповненні слотів фрейму і можливо у виконанні деяких маніпуляцій над фреймом із заповненими слотами.

Таким чином, використання фреймів для опису знань про прості задачі інформаційного обслуговування забезпечує природний спосіб подання як апріорних знань про проблемну сферу, так і знань про процес взаємодії системи з користувачем.

Ідентифікація сутностей по опису (буквальна інтерпретація). При взаємодії людей слухач має здатність по опису співрозмовника ідентифікувати ті з відомих йому об'єктів чи подій, що співрозмовник мав на увазі. При цьому опис може бути докладним (за відсутності контексту) і зовсім коротким (за наявності контексту). Коли співрозмовник описує деяку сутність, то він хоче однозначно ідентифікувати її для слухача. Для цього співрозмовник використовує свої уявлення про знання слухаючого. Він конструює такий опис, що, на його думку, дозволить слухачу однозначно ідентифікувати описувану сутність. При спілкуванні між людьми співрозмовнику часто вдається домогтися однозначності, однак неминучі і невдачі.

Відповідно до принципу неявного підтвердження співрозмовник припускає успішність комунікації і, зокрема те, що слухаючий по описах однозначно ідентифікує згадані в повідомленні сутності, якщо слухаючий не вказує зворотнього. Тоді той, хто слухає, не перериває природного ходу діалогу, якщо повідомлення однозначне, однак в разі наявності багатозначності чи незадовільності якогось опису слухач ініціює піддіалог, у ході якого даний опис уточнюється.

При розв'язуванні простих задач інформаційного обслуговування система з описів користувача звичайно має виділяти сутності, що є значеннями параметрів обслуговування, хоча навіть у таких простих додатках сутності не можуть бути зведені до скінченної множини сутностей, відомих системі. Дійсно, при розв'язуванні задачі резервування звичайно потрібно знати прізвище клієнта. Очевидно, що системі не можуть бути заздалегідь відомі прізвища всіх можливих клієнтів. Досвід показує, що навіть при розв'язуванні простих задач інформаційного обслуговування описи, що складаються користувачем, у загальному випадку не є однозначними, тобто неминучі багатозначні інтерпретації.

Пояснювальні можливості засобів спілкування

Поняття «пояснення» вивчається логікою і теорією пізнання і трактується дуже широко. Можна говорити про наукове і повсякденне пояснення. Єдиного, загальноприйнятого визначення даного терміна не існує. З цієї причини було дано непряме визначення терміна «пояснення», тобто визначення через сукупність параметрів, що його характеризують . У літературі стосовно штучного інтелекту використовується таке визначення: «Пояснити — зробити ясним, відповісти на запитання ЯК, ЧОМУ».

Класифікація пояснювальних можливостей. У загальному випадку один учасник спілкування може зажадати від іншого будь-які пояснення. У системах спілкування це не завжди можливо. Пояснювальні здатності систем спілкування повинні включати, принаймні, що з цього випливають види пояснень:

1) пояснення здатностей системи;
2) пояснення фактичних подій;
3) пояснення гіпотетичних подій;
4) пояснення критичних ситуацій.

Пояснення першого виду видаються у відповідь на питання користувача про здатності системи. Прикладами зазначених запитань є такі: «Чи можете ви повідомити номер телефону в Києві?»; «Чи можете ви прийняти замовлення квитків на літак?»; «Що ви вмієте робити?». Незважаючи на те, що багато запитань про здатності вимагають під час відповіді на них виконати відповідні дії, деякі запитання про здатності вимагають буквальної відповіді. Для відповіді на такі запитання система спілкування повинна мати наочну модель того, що вона вміє робити, і того, що вона не вміє.

Пояснення другого виду видаються у відповідь на запитання користувача щодо дій системи, виконуваних у минулому, сьогоденні і майбутньому. Приклади таких запитань: «Що ви щойно сказали?»; «Ви замовили квитки на Київ?». Для генерації пояснень другого виду необхідно зберігати трек дій системи і знати структуру цілей діалогу.

Пояснення третього виду видаються у відповідь на запитання користувача про гіпотетичні дії, тобто про дії системи не у фактичних, а в передбачуваних ситуаціях. Приведемо приклади таких питань: «Ви можете повідомити номер телефону, якщо я повідомлю вам адресу?»; «Ви зможете задовольнити замовлення, якщо перенести день відправлення на завтра?». Відповіді на гіпотетичні питання прагнуть від системи здатностей конструювати гіпотетичні ситуації, що мають існувати одночасно і незалежно як від поточної ситуації системи, так і від інших ситуацій, попередньо визначених користувачем.

Пояснення четвертого виду генеруються системою самостійно, а не за запитом користувача, як це відбувається в усіх інших розглянутих нами видах пояснень. Пояснення такого виду призначені для того, щоб у випадку повного нерозуміння системою користувача спробувати уникнути глобальної невдачі в діалозі.

Зауважимо, що в ряді систем до пояснень відносять і генерацію відповідей на питання про зміст предметної сфери. Ці відповіді не слід відносити до пояснювальних здатностей, тому що вони реалізуються за допомогою іншого механізму.

Цей набір пояснювальних можливостей достатній для ефективної взаємодії з користувачем під час розв'язування простих задач інформаційного обслуговування. Розглянуті види пояснень мають більш широкі можливості, чим це прийнято в традиційних СКБД і питально-відповідних системах.

Пояснення здібностей системи. У спілкуванні між людьми той, хто слухає, може інтерпретувати запитання про його здібності одним із двох способів: чи буквально, чи як вимога виконати дію, вкладена в дане питання. Прикладом запитання, що передбачає буквальну інтерпретацію, є такий: «Ви вмієте грати на скрипці?».

Слід зазначити, що питання про здібності звичайно можуть бути виражені як від другої особи, так і від першої особи. Наприклад, користувач може поставити запитання як у вигляді «Ви можете повідомити номер телефону?», так і у вигляді «Я можу з'ясувати номер телефону?». Загалом, питання від першої особи можуть розглядатися так само, як питання другій особі.

Пояснення фактичних подій. Тут будуть розглянуті питання про фактичні події (у минулому, чи сьогоденні, майбутньому), тобто про події, що сталися чи мають статися, а не про події, що могли б відбутися, якби навколишній світ був якось змінений. Імовірно, системи найближчого майбутнього будуть здатні відповідати тільки на наступні питання про події:

1) питання про власні дії системи і про результати цих дій;

2) питання про ті дії користувача, що безпосередньо спостерігалися системою.

Питання про події, що знаходяться поза безпосереднім спостереженням системи, будуть нею не сприйняті.

Приведемо приклади типів питань, на які повинні уміти відповідати системи спілкування:

1. Що ви щойно сказали?
2. Чому ви хочете знати моє ім'я?
3. Повідомлення, що я відправив учора, доставлено?

4. Моє повідомлення буде відправлено негайно?

Перший приклад містить явну вказівку про нерозуміння. Такі запитання про те, що було сказано, служать забезпеченню ясності комунікації і є приватним видом питань про дії. Для відповідей на запитання про події, як правило, потрібно запам'ятовувати всі події, що мали місце в поточному сеансі взаємодії (у випадку дій системи необхідно запам'ятовувати і їхні причини).

Питання про події можуть посилатися на події з попередніх сеансів взаємодії. Як правило, недоцільно зберігати детальну історію всіх попередніх сеансів. Імовірно, досить пам'ятати тільки «головні» події. Питання про майбутні дії вимагають, щоб система могла «уявити», які події будуть мати місце в майбутньому, виходячи з поточних умов.

Так само, як і питання про здатності, запитання про події можуть використовувати непрямі мовні обороти. Для більшості «так/ні» запитань про події видача просто негативної відповіді є неприйнятною тому, що такі запитання звичайно необхідно розглядати як непрямий мовний зворот, що містить вимогу видати пояснення при негативній відповіді. Так, на запитання «Моє вчорашнє повідомлення доставлене адресатові?» недостатньо відповісти «Ні». Необхідно пояснити причини, що перешкодили доставити повідомлення. Аналогічно запитання «Моє повідомлення буде відправлено негайно?» не є тільки «так/ні» запитання. Цілком природно інтерпретувати такий запит як бажання користувача відправити його повідомлення негайно.

Варто підкреслити, що деякі запитання про події не вимагають відповіді. Наприклад, при замовленні квитків запитання користувача «Я вже говорив вам, що мені треба п'ять квитків?» по суті є не запитом, а повідомленням інформації. Тут користувач хоче переконатися в тім, що система знає про кількість квитків, що їх він замовляє. Розпізнавання таких мовних зворотів системами спілкування є важливої, але невирішеної поки проблемою.

Пояснення гіпотетичних подій. Системи спілкування, крім відповідей на запитання про події в контексті поточної ситуації, повинні бути здатні відповідати на запитання про події в контексті ситуації, запропонованої користувачем. Обробка гіпотетичних питань є надзвичайно складною задачею, однак при роботі з простими сферами задача значно спрощується, оскільки в простих сферах суттєво обмежені число і взаємозв'язки перемінних, щодо яких користувач може висунути певні гіпотези.

У випадках простих сфер інформаційного обслуговування найбільш важливий клас гіпотез стосується значень одного з параметрів чи обслуговування значень його підслотів. Користувач може поставити запитання щодо здатності, спираючись на гіпотезу, що певний слот має визначене значення чи знаючи, що значення (невизначене) цього слота відомо системі.

Для відповіді на гіпотетичні питання система повинна сконструювати уявлення щодо гіпотетичної ситуації і потім у контексті цієї ситуації вирахувати відповідь на питання. При конструюванні гіпотетичної ситуації система повинна подбати про те, щоб дана ситуація не входила в конфлікт із фактичною ситуацією чи з іншими ситуаціями, що їх користувач визначив раніше. Справа в тім, що користувач у

ході діалогу може повернутися до попередньо визначеної ситуації, може ввести іншу ситуацію і може переходити від однієї ситуації до інших неодноразово (типовим прикладом є резервування квитка з пункту А в пункт В при наявності декількох маршрутів з А в В і наявності декількох рейсів). Тобто система повинна мати можливість незалежного подання альтернативних ситуацій, що мають загальну інформацію, і можливість вибору альтернатив.

Структура діалогу типу запитання-відповідь

Основою для класифікації інтерфейсу людина-комп'ютер є структура діалогу. Вона визначає ступінь взаємодії між системою і користувачем; вона також задає основні форми керування цією взаємодією. Традиційно виділяють чотири основні структури типів:

- запитання і відповідь;
- меню;
- екранних форм;
- на базі команд,

кожна з який ґрунтується на аналогії з визначеним типом взаємодії між людьми. Розглянемо, як перші дві з цих структур:

- надають користувачеві й одержують від нього інформацію;
- відповідають психологічним особливостям різних користувачів, що виконують різні завдання;
- можуть бути реалізовані за допомогою процесів вводу-виводу.

Оскільки під час побудови діалогу людина — комп'ютер машина не може зіграти цілком роль людини, розроблювачу не варто занадто категорично орієнтуватися на моделі спілкування людей один з одним. Однак, як правило, придатність структури діалогу можна оцінити шляхом аналізу того, яка структура діалогу людина-людина використовується в аналогічній ситуації. Щоб зробити цю оцінку більш точною, ми скористаємося в першу чергу п'ятьма основними критеріями — природністю, послідовністю, стислістю, підтримкою користувача і гнучкістю. Керуючись цими критеріями, можна оцінити ступінь придатності будь-якого діалогу.

Особливості

Структура діалогу типу запитання і відповідь (З/В) ґрунтується на аналогії зі звичайним інтерв'ю. Система бере на себе роль інтерв'юера й одержує інформацію від користувача у вигляді відповідей на запитання. Це найбільш відома структура діалогу, оскільки для її реалізації потрібні процеси вводу і виводу. Спочатку цю структуру називали ***діалоговим режимом***.

Термін «питання і відповідь» використовується не зовсім точно, оскільки всі діалоги, керовані комп'ютером, складаються так чи інакше з питань, на які користувач відповідає. Однак у структурі З/В цей процес виражений явно. У кожному пункті діалогу система виводить як підказку одне запитання, на яке користувач дає одну відповідь. Залежно від отриманої відповіді система може вирішити, яке наступне запитання ставити.

Якщо той, хто відповідає на запитання інтерв'юера, дає неточну відповідь, комп'ютер зазвичай повторює запитання, повідомляючи, що відповідь не підходить. Аналогічно, якщо помилкова відповідь вводиться в структуру З/В, система видає повідомлення про помилку і знову виводить підказку; цей процес повторюється доти, поки не буде отримано прийнятну відповідь. Відповідь звичайно вводиться користувачем у вигляді текстового рядка, що набирається на клавіатурі (рис. 8.1). Цей рядок може являти собою або об'єкт зі списку можливих відповідей (обраний об'єкт), або довільні дані.

Який гросбух	?	**постачання**
Команда	?	відіслати
Тип об'єкта	?	рахунок
Номер рахунку	?	123456
Клієнт	?	с123

Рис. 8.1. Приклад сеансу діалогу в режимі запитання-відповідь

Критерії розробки

Структура З/В надає природний механізм уведення як керуючих повідомлень (команд), так і даних; інтерв'юер може визначити, що хоче зробити той, у кого беруть інтерв'ю, і стосовно чого. Ніяких обмежень на діапазон чи тип вхідних даних, що можуть оброблятися, не накладається. Існують системи, відповіді в яких даються природною мовою, але частіше використовуються пропозиції з одного слова з обмеженою граматикою.

Природність діалогу значною мірою залежить від характеру запитань, що задаються в процесі інтерв'ю. Якщо інтерв'юер задає занадто довгі питання, той, хто відповідає, до закінчення запитання забуде, про що його запитували на початку. Він також має легко визначити, де є запитання, а де немає. Усе сказане залишається справедливим і для структури З/В.

Для полегшення сприйняття довжину повідомлення варто обмежити приблизно 40 символами, що виводяться в лівій частині екрана, що складає 2/3 ширини екрана. Навідні запитання системи мають чітко відрізнятися від відповідей користувача; цього можна досягти, змінивши чи регістр, чи колір відповідей, підвищивши їхню контрастність, увівши розділові знаки, наприклад двокрапку.

Структура діалогу типу питань і відповідей достатнім чиномзабезпечує підтримку користувача, тому що навіть коротке навідне запитання за розумної побудови може бути зроблене саме так, що все пояснює. Якщо необхідно, у навідне запитання можна включити додаткову інформацію, наприклад формат введення:

рахунки від (дд/мм/рр):

Ступінь підтримки користувача не збільшується, якщо питання стає багатослівним чи надто ввічливим. Ви невдовзі дуже втомитесь від інтерв'ю, якщо інтерв'юер буде часто вдаватися до вишуканого стилю чи запопадливо випереджати кожне запитання чемним «будь ласка». Однак структуру З/В можна оснастити гарною підтримкою користувача за його бажанням. Якщо користувачеві для відповіді на

яке-небудь запитання потрібна довідка, система може надати йому докладну інформацію, яка потрібна для відповіді на поставлене запитання, наприклад:

Команда?: help

Припустимі команди:

post — переслати дані у програму фінансового обліку;

report — вивести розрахунок балансу, аналіз чи пропозицій людей за віком;

monthend — продовжити обробку гроссбуха в наступному місяці.

Структура З/В не гарантує мінімального обсягу введення, оцінюваного в кількості натискань клавіш, однак при придатному підборі скорочень можна зменшити будь-яку надмірність. Навіть якщо уведення відбувається досить швидко, для людини, що вже знає, які запитання ставить система і які відповіді потрібно на них давати, відповідати на всю серію запитань досить обтяжливо. Потрібен механізм, що дає можливість користувачеві відповісти відразу на всі питання. Такий механізм називають випереджувальним введенням, і його можна легко реалізувати. Він підвищує гнучкість структури З/В.

Висновки

Хоча структура З/В може здатися дещо застарілою з появою таких могутніх пристроїв відображення інформації, як процесори введення-виводу, у неї проте є свої переваги. Це проміжна структура, що може задовольнити вимоги різних користувачів і типів даних. Вона досить гнучка й у достатньому ступеню забезпечує підтримку користувача. Зокрема, така структура особливо доречна при реалізації діалогу з безліччю «відгалужень», тобто в тих випадках, коли на кожне запитання передбачається велика кількість відповідей, кожна з який впливає на те, яке запитання буде поставлено наступним. З цієї причини структура З/В часто використовується як діалогова структура в експертних та навчаючих системах.

Структура діалогу типу меню

Особливості

Меню є, очевидно, відомим механізмом організації запитів на введення даних під час діалогу, керованого комп’ютером. Меню було неодмінною приналежністю ранніх дешевих пакетів прикладних програм для мікрокомп’ютерів, і звичайно саме його наявність малася на увазі продавцем, коли він описував такий пакет як «дружній». Надалі популярність меню зросла завдяки появі інтерфейсу, керованого за допомогою маніпулятора типу «миша».

Сутність структури меню полягає в тім, що в користувача є список можливих варіантів даних для введення, з якого треба вибрати те, що потрібно. Існують різні формати представлення меню на екрані, деякі з яких наведено на рис. 8.2.

Меню у вигляді блоку даних на екрані — це традиційний формат. Інші формати стали популярними з появою «віконної» техніки виводу даних, в даний час повсюдно використовуваної на ПК.

Рис. 8.2. Приклади різних форматів меню
а — меню у вигляді блока даних;
б — меню у вигляді піктограмм;
в — меню у вигляді рядка даних;
г — список об’єктів, вибираних вказівкою цифрових кодів;
д — список об’єктів, вибираних вказівкою мнемонічних кодів.

Меню у вигляді рядка даних може з’являтися чи угорі, чи внизу екрана і часто залишається в цій позиції протягом усього діалогу. Таким чином, за допомогою меню зручно відображати можливі варіанти даних для введення, доступних у будь-який час роботи в діалозі. Ще один поширений спосіб подання використовується для відображення додаткових меню у вигляді блоків даних, що «вискакують» або «випадають» на екран у поточному положенні курсора або «витягаються» безпосередньо з рядка меню верхнього рівня. Ці меню зникають після вибору варіанта. Меню у вигляді піктограм являє собою безліч блоків об’єктів вибору, розкиданих по всьому екрану; часто об’єкти вибору містять графічне подання варіантів роботи.

В діалозі з використанням меню не треба виводити на екран уточнюючі питання. Користувач діалогового меню може вибрати потрібний пункт, уводячи текстовий рядок, що ідентифікує цей пункт, указуючи на нього чи безпосередньо переглядаючи список у режимі ролика і вибираючи з нього.

Система може виводити пункти меню послідовно, при цьому користувач вибирає потрібний йому пункт натисканням клавіші.

Контроль правильності даних, що вводяться, у системі, що використовує меню, полягає в перевірці відповідності введеної інформації списку об’єктів вибору з меню. При наборі ідентифікатора на клавіатурі, введена інформація перевіряється на відповідність вмісту об’єкта вибору меню; при безпосередній указівці дані, що вводяться, перевіряються на розташування усередині поля відображення об’єкта вибору на екрані. Якщо дані введено неправильно, сформувати повідомлення про помилку досить просто: оскільки прийнятні варіанти відповідей відображаються на

екрані, помилка може відбутися тільки при наборі інформації і користувач легко її знайде.

Цей процес повторюється доти, поки не буде отриманий прийнятний варіант введення. Зазначимо, що меню не виводиться заново щораз.

Як і у випадку з З/В структурою, у діалозі на базі меню має сенс або значення обраного пункту меню (вміст об'єкта вибору), або номер обраного пункту меню (перший, другий і т.д.).

Критерії розробки

Меню можна з рівним успіхом застосовувати для введення як керуючих повідомлень, так і даних. Прийнятна структура меню залежить від розміру й організації меню, від способу вибору пунктів меню і реальної потреби користувача в підтримці з боку меню.

У випадку великого списку на екрані ситуація ускладнюється тим, що читати текст на екрані значно складніше, ніж текст на папері. В ідеалі екранне меню має містити 5—6 пунктів; варто уникати списку з 10 чи більше пунктів.

Деякі розроблювачі реалізують меню у вигляді блока даних, для введення яких потрібно набирати лише слова «Так» чи «Ні».

При значній кількості можливих варіантів вибору їх варто згрупувати в ієрархію невеликих меню. На рис. 8.3 показано меню, що може з'явитися після вибору користувачем пункту «Автомобіль» з меню «Види страхування», приведеного на рис. 8.2. Якщо не вдається легко побудувати ієрархію, із упевненістю можна припустити, що структура діалогу на базі меню не підходить. З цієї причини меню рідко використовується для введення даних (наприклад, коду клієнта чи імені файла), коли діапазон їхніх значень широкий.

Рис. 8.3. Підменю в ієрахічній структурі

Деякі автори продемонстрували виняток з цього правила. При роботі на низькошвидкісних терміналах у складі мережі доцільніше використовувати лінійну структуру з великих (багатостовпцевих) меню, ніж ієрархічну з малих меню; таким чином, можна подолати тимчасові затримки, пов'язані з повторним викликом наступних один за одним меню. Якщо всі дії, пов'язані з вибором, реалізуються послідовним переглядом, то це також приводить до роботи з меню, що містить значну кількість можливих варіантів, наприклад імена файлів у каталозі, який не можна розділити; у цьому випадку ми також приходимо до меню типу «Так — Ні».

Усередині кожного меню з ієрархічною структурою велике значення має порядок проходження пунктів. У діалоговому меню пункти мають йти в природному порядку, якщо такий можна установити. Якщо існують якісь сумніви з приводу «природності» чи порядку і якщо список порівняно великий, то пункти меню варто розташовувати за абеткою. Слід подбати про те, щоб установлювати порядок там, де

його не було, чи там, де упорядкування неправильне. Люди, як правило, вважають, що виконання п. 1 у списку має передувати п. 2. Тому пункт меню, пов'язаний із завершенням роботи, часто вміщують наприкінці меню.

Структура типу меню є найбільш питомим механізмом для роботи з пристроями вказівки і вибору: меню являє собою зображення тих об'єктів, що вибираються користувачем. Якщо діалог складається винятково з меню, можна реалізувати послідовний інтерфейс, при якому користувач застосовує тільки пристрої для вказівки; однак така сталість рідко досяжна на практиці. Варто також зауважити, що, хоча робота з цими пристроями не вимагає майстерного володіння клавіатурою, для підготовленого користувача — це не найшвидший спосіб вибору з меню.

Структура діалогу на основі екраних форм

Введення

Розглянемо дві структури, що обробляють одночасно цілу серію відповідей. Ми побачимо, що для обробки кожної введеної відповіді з такої серії можна використовувати такі ж абстракції, як в діалозі типу З/В та меню.

Всі чотири типи структур всебічно охоплюють кожен окремий крок діалогу. Кожна з них добре підходить для визначеного кола користувачів чи типу вхідних повідомлень. Однак у більшості прикладних програм подібна організація діалогу повинна підтримувати різні типи вхідних повідомлень і різні рівні підготовки користувача; не існує єдиної структури діалогу, що однаково добре підходила б для усіх варіантів діалогу. Розглянемо, як ці основні структури можна комбінувати одну з одною, щоб задовольнити різноманітні вимоги різних сфері застосування. Такі комбінації називають ***змішаною структурою діалогу***.

Властивості

Один зі способів одержання інформації від інших людей полягає в тім, щоб ставити їм запитання і вислухувати їхні відповіді. Такий підхід знайшов відображення в структурі типу запитання — відповідь. Інший підхід — одержати від них відразу всю інформацію заповненням форми, наприклад, поданої на рис. 8.4.

Рис. 8.4. Діалог на основі екранної форми

Подібні форми широко використовуються при замовленні товарів, резервуванні місць у готелях, розрахунках за товари і послуги, висновку страхових договорів, складанні анкет і т.д. Службовці більшості фірм працюють зі стандартними формами, наприклад рахунками, видатковими і платіжними відомостями. На практиці форми

використовуються в основному там, де облік якої-небудь діяльності (трансакція) вимагає введення досить стандартного набору даних.

Цей принцип збору інформації лежить в основі структури діалогу типу екранної форми. На відміну від структури типу питання — відповідь, у якій одночасно ставиться тільки одне питання, у випадку екранної форми перед користувачем ставиться відразу кілька запитань. Ця множинність запитань досить постійна в тім сенсі, що відповідь на попередні запитання не впливає на те, яке запитання буде поставлене наступним.

Бухгалтерська форма звичайно заповнюється власноруч згори вниз. Той, хто заповнює форму, може вибирати послідовність відповідей, тимчасово пропускати якесь запитання, повертатися назад для корекції попереднього чи відповіді, навіть розірвати бланк і почати заповнювати новий. Він працює з формою доти, доки не заповнить її цілком і передасть одержувачу.

Діалог типу екранної форми звичайно має такі ж можливості. Користувач може відредагувати якусь відповідь перед введенням. Він може також тимчасово пропускати запитання і повертатися до відповіді на попередні питання. Користувач продовжує працювати з формою доти, доки він, удоволений результатами роботи, не натисне визначену клавішу, що означає кінець уведення, або не відповість ствердно на задане запитання типу

Переходити до обробки (т/н)

Якщо одержувач цієї форми присутній при її заповненні, він може вказати на помилки в міру їхньої появи. Однак при цьому відволікається увага заповнюючого форму, що може призвести до збільшення кількості помилок. Краще почекати закінчення заповнення форми і перевірити її цілком. Потім позначити помилки і попросити їх виправити.

Комп'ютерна система може перевіряти кожну відповідь безпосередньо після введення, або ж вона може виждати і вивести список помилок тільки після заповнення форми цілком. У деяких системах інформація, що вводиться користувачем, стає доступною тільки після натискання клавіші «уведення» звичайно по закінченні заповнення форми. Питання в тім, чи треба перевіряти відповідь безпосередньо чи відкласти до кінця уведення усіх відповідей, вирішити непросто; повідомлення про помилки, виведені безпосередньо після відповіді, можуть відвернути увагу, але можуть справити і позитивний вплив. Взагалі в тих випадках, коли інформація для уведення вибирається з якогось вихідного документа, перевірку краще відкласти до кінця заповнення форми, щоб не переривати процес уведення; якщо ж такого вихідного документа немає, то перевірку слід виконувати відразу після уведення відповіді.

Якщо зустрілася якась помилка, діалог не повинен заново виводити порожню форму; виводиться форма з попередніми відповідями і допущеними помилками. Як і у випадку заповнення звичайної форми, ви одержуєте новий бланк тільки в тому разі, якщо попередній приведений до такого стану, що простіше почати з початку, чим продовжувати старий заповнювати бланк.

Так само як і в структурах типу питання — відповідь і меню, окрема відповідь може вибиратися зі списку можливих варіантів або вона вводиться у вигляді довільних значень. Це, як правило, символьний рядок, але вибір можна здійснити переглядом

схованого меню. У традиційних формах це відповідає рекомендації «непотрібне викреслити».

Критерії розробки

Форми є питомим способом уведення вмісту трансакції, оскільки трансакція за визначенням містить порівняно стандартний набір значень даних. Таким чином, цю структуру доречно застосовувати там, де джерелом даних служить існуюча канцелярська форма. У прикладі на рис. 8.4 діалогу заздалегідь відомо, які одиниці даних вимагаються, оскільки кожне поле задається вмістом, розташуванням на екрані (слотом) і атрибутами.

Критерії, використовувані для структури типу запитання — відповідь, застосовні до побудови і відображення окремих питань екранної форми. Поля відповідей повинні знаходитися в межах невеликих прямокутних областей екрана, виділених атрибутів, чи обмежуються за допомогою спеціальних символів, наприклад квадратних дужок, як у прикладі на рис. 8.4. Інша частина екрана повинна бути захищена, щоб у режимі луна-друку дані, що вводяться, відображалися лише у визначених областях.

Той факт, що на екрані міститься цілий ряд запитань, уводить додаткові обмеження, що ускладнюють розміщення даних на екрані.

Важливо, щоб форма, що відображена на екрані, була схожа на ту форму на папері, що є джерелом інформації. Не обов'язково, щоб зовнішній вигляд цих форм збігався, це навіть може погіршити сприйняття даних на екрані, але всі елементи даних, що вводяться, повинні розташовуватися в тому ж відносному порядку і мати такий самий формат, що й у вихідному документі. Зміст цієї вимоги очевидний, якщо взяти до уваги процес уведення.

Часто всі необхідні одиниці не можна відобразити одночасно в межах одного екрана і їх необхідно поділити на групи, що відображаються на послідовності екранів. Важливо, щоб ця розбивка зберігала логічні зв'язки і не спричинялася до рознесення зв'язних частин документа на різні екрани. Наявні канцелярські форми допомагають виконати такий поділ.

Структура діалогу типу екранної форми забезпечує високий рівень підтримки користувача. У тих випадках, коли фізична форма служить джерелом даних, від діалогу потрібна дуже незначна додаткова підтримка. Оскільки користувачеві потрібно просто внести в неї інформацію, весь екран практично можна заповнити запитаннями у вигляді коротких заголовків. Єдина вимога полягає в тому, щоб користувач міг прочитати введену ним інформацію з метою контролю.

У разі відсутності відповідної фізичної форми, користувачу необхідно читати зміст екрана, щоб зрозуміти, що потрібно вводити. У цьому випадку повинна бути заповнена менша частина екрана, а питання повинні бути більш докладними. Для кожного питання повідомлення про помилки і довідкова інформація повинні бути передбачені, і користувач повинен мати можливість одержати довідку з будь-якого питання форми. Користувачеві можна надати підтримку, уключивши деякі елементи формату відповіді в питання (поле відповіді, наприклад):

Дата Рахунка (dd/mm/yy) [ ] або Дата Рахунка[//]

При цьому важливо не перестаратися; наприклад, формат адреси

Будинок і вулиця [ ]

Район [ ]
Місто [ ]
Округ [ ]

припускає тільки приміські адреси, тоді як спосіб завдання адреси залежить від місцевості. У деяких автоматизованих системах при такому форматі майже неможливо зробити замовлення, якщо ви живете на фермі: для оформлення замовлення за допомогою ЕОМ треба вказати номер будинку, інакше вона не перейде до введення наступних даних!

Проблема виникає і тоді, коли форма на папері містить поля, що не потрібно вводити. Такі області документа повинні бути чітко відділені від полів, уміст яких треба вводити. Форми можуть містити також необов'язкові поля чи поля, що заповнюються, якщо якесь значення вводиться в попереднє поле; такі поля найкраще розташовувати наприкінці форми. Якщо це неможливо, система має автоматично пропускати будь-яке запитання, пов'язане з цим необов'язковим полем, якщо введення в нього не потрібне, і виводити відповідне пояснення.

Реалізація

Подібно структурі типу меню, структура типу екранної форми будується в два етапи: по-перше, форма відображається цілком, і, по-друге, питання повторюються доти, доки не завершиться заповнення форми. Заповнення форми закінчується введенням ознаки завершення, або (якщо така ознака не визначена) відповідями на всі запитання.

Форму можна визначити як набір вихідних полів. Однак на відміну від меню цей набір неможливо однозначно поділити на допоміжні структури, що будуть входити в будь-яку форму. Це можуть бути поля, що у деяких формах виконують роль «заголовка форми» і «заключного тексту форми». Інші вихідні поля задають заголовки; хоча вони являють собою запитання користувачеві, вони не мають відношення до процесу діалогу. Можливі також і інші поля, такі як підзаголовки і «обмежники», що задають межі форми. Ми можемо вважати всі ці поля неосновними «фоновими» полями форми.

Структура діалогу на основі командної мови

Властивості

Структура діалогу на основі мови команд настільки ж поширена, як і структура діалогу типу меню, насамперед тому, що вона дуже часто використовується в операційних системах. Вона розташовується на іншому кінці спектра структур діалогу стосовно структури типу меню. Термін «команда» відбиває подібність діалогу такого типу з діалогом, що відбувається під час військових занять: користувач виконує роль командира, наказуючи солдату-комп`ютеру. Командир-користувач вкладає в команду всю інформацію, необхідну для виконання поставленої задачі солдатом. Команда містить у собі постановку задачі, можливі пояснення і дані, необхідні для її розв'язання. Наприклад:

«Стій» — задачі не вимагаються дані

або

«Очисти мені апельсин» — задача і її дані.

Система нічого не виводить, крім постійної підказки (наприклад, підказка «ім'я пристрою», прийнята для більшості мікрокомп'ютерних операційних систем), що означає готовність системи до роботи. Кожну команду вводять з нового рядка і звичайно закінчують натисканням клавіш ВК чи Введення

A>dir

A > pip b := a:*.com

A > mode com1:9600,e,7,l,p

Мова команди ґрунтується на принципі "необговорення" команди. Відповідальність за правильність подаваних команд лягає на користувача.

Якщо команду неможливо виконати, наприклад, коли подальша обробка введеного рядка неможлива, структура на базі мови команд відмовляється від її виконання причому здебільшого без жодного повідомлення про помилку, і усе введення необхідно повторити, наприклад:

A > drgi a.*.com

Невірна команда чи ім'я файла (що невірно?)

Критерії розробки

Подібно до структури типу меню, структура на базі мови команд зручна для уведення вихідних керуючих повідомлень, однак вона забезпечує більш широкі можливості вибору в будь-якій крапці діалогу і не вимагає ієрархічної організації фонових завдань. Тому ця структура зручна для реалізації діалогу з операційною системою, у якому фонові завдання являють собою плоску структуру рівноправних і самостійних завдань.

Хоча розглянута структура може підтримувати досить велику кількість команд, на практиці їх кількість звичайно обмежується, щоб не перевантажувати пам'ять користувача. Структура на базі мови команд не відрізняється гарною підтримкою користувача і придатна для підготовлених і постійних користувачів. Перед використанням такої системи користувач має пройти початковий курс навчання, а надалі він вивчить усі тонкощі роботи із системою по документації, а не на практиці.

Більш того, оскільки системі невідомо, що має намір робити користувачеві, важко придумати яку-небудь реальну допомогу користувачу в процесі роботи, крім як досить загального характеру.

Оскільки дана структура припускає великий обсяг матеріалу, що запам'ятовується, імена команд варто вибирати так, щоб вони несли значенневе навантаження і легко запам'ятовувалися. Допомогою в запам'ятовуванні може стати не тільки вибір імен команд, але навіть підбор клавіш для команд керування курсором.

Розроблювач повинен намагатися уникати зайвої функціональності, яка виникає при створенні власного командного рядка для кожної функції, задачі, що виконується, тобто не створювати безліч різноманітних команд з функціями, що перекриваються. Такі «благі» наміри нерідко призводять до появи великої кількості ключових слів для позначення команд і синтаксичних правил, багато з яких рідко використовуються і викликають розгубленість більшості користувачів.

Позиційні параметри зменшують обсяг інформації, що вводиться, але їх очевидним істотним недоліком є те, що значення, що вводяться, повинні вказуватися у

визначеному порядку, оскільки, переставивши місцями вхідний і вихідний файли, можна одержати вкрай небажаний результат. Завдання позиційних параметрів ускладнюється, якщо список параметрів досить великий; деякі команди операційних систем мають десяток і більше параметрів. Цей недолік трохи нівелюється, якщо дозволено опускати параметри, уводячи два роздільники один за одним.

Ключові параметри зменшують навантаження на пам'ять користувача в тому розумінні, що не треба запам'ятовувати порядок проходження параметрів, крім того, можна опускати необов'язкові параметри. З іншого боку, користувачеві потрібно запам'ятати безліч ключових слів, а розроблювачу — придумати для них «змістовні» імена. Цей підхід також вимагає більшого часу роботи системи, щоб розпізнати ключові слова, задані в довільному порядку.

У багатьох мовах команд список параметрів може містити ключі, що уточнюють спосіб інтерпретації команди. Звичайно ключі позначаються спеціальними ідентифікаторами, і їх можна розташовувати в будь-якому місці списку параметрів.

Багато командних мов підтримують макрози, що розширюють функціональні можливості діалогу без збільшення кількості команд. Макроз містить кілька окремих командних рядків, розташованих у текстовому файлі. Коли вводиться ім'я файла, окремі рядки команд макрозу виконуються одна за одною, як якби вони вводилися з клавіатури. У командних рядках макрозу можна використовувати формальні параметри, що будуть замінені фактичними параметрами, що задаються при виклику макросу.

Реалізація

Існує очевидна подібність між уведенням команди і введенням даних у готову форму. Уведення команди можна розглядати як уведення ряду відповідей на ряд поставлених питань. Таким чином, введення

A > corp FromFileName ToFileName

можна трактувати як відповіді, на які мається на увазі питання:

Команда corp

Відкіля FromFileName

Куди ToFileName

Як і при роботі з екранною формою, тут має бути визначена ознака того, що користувачем завершено введення; як такий обмежник для команди використовується символ ВК. Однак на відміну від форми набір питань заздалегідь невідомий. Невідомий доти, доки не пізнано конкретну команду (у деяких випадках) чи конкретний різновид команди.

Висновки

Структура на основі мови команд згідно зі своїми можливостями найшвидша і гнучкіша з усіх структур діалогу, і більшість діалогів на базі «природної мови» реалізується за допомогою мов команд із дуже великим набором ключових слів. Підготовлений користувач зазнає задоволення від відчуття того, що він керує системою, а не навпаки. Однак ця структура не забезпечує користувача підтримкою, тому навіть підготовлені користувачі вважають, що важко використовувати всі закладені в ній можливості до кінця. Більшість же користувачів добре знайомі тільки з дуже обмеженими засобами, якими вони користуються регулярно.

На практиці багато з описаних аспектів мов команд можна імітувати структурами типу питання — відповідь і процесами, описаними в попередніх розділах.

Інструментальні засоби розробки пакетів

Застосування будь-яких інструментальних засобів розробки ППЗ спрямоване на підвищення продуктивності праці розробників на різних стадіях створення програмного виробу. До них належать засоби автоматизації розробки програмних модулів, безпосередньо пов'язаних з формуванням вихідних повідомлень; засоби проектування вікон користувача; засоби проектування взаємодії користувача з програмною системою та ін.

Зосередимо увагу на засобах реалізації запитів користувача мовою структурованих запитів SQL, яка у поєднанні з можливостями генератора звітів дає змогу отримати вихідні повідомлення різної складності.

SQL поєднує в собі можливості мови, визначення даних, мови маніпулювання даними і мови запитів. Крім того, вона є основним компонентом в таких популярних СКБД, як Oracle, DB2, SQL Server, InterBase, SQL/DS, Ingress, OS/2 Extended Edition, Informix, Sybase, SQLBase, VAX SQL.

SQL є найбільш могутньою і популярною мовою маніпулювання даними реляційних систем керування базами даних (СКБД). Основні можливості мови, доступні в SQL-орієнтованій СКБД, визначено в рамках спеціального стандарту ANSI/ISO. SQL-стандарт визнано з 1982 року, а формально визначено як ANSI-стандарт X3.135 в 1986 році і як ISO-стандарт — в 1987. Стандартизація можливостей SQL продовжується і зараз. На зміну стандарту SQL2, реалізованому в більшості комерційних СКБД, приходить новий варіант — SQL3.

Назва «SQL» є абревіатурою від Structured Query Language (структурована мова запитів). Раніше використовувалася й інша назва — SEQUEL, як вимова «S.Q.L.». SQL поєднує в собі можливості мови визначення даних, мови маніпулювання даними і мови запитів. При цьому він реалізовує і основні функції реляційних СКБД:

Визначення даних

SQL дозволяє визначити структуру даних, що підтримуються, й організацію використаних реляційних відношень.

Доступ до даних

SQL забезпечує доступ до даних, що зберігаються, з прикладних програм.

Маніпулювання даними

SQL дозволяє користувачеві або прикладній програмі змінювати вміст бази даних вставлянням нових даних, видаленням або модифікацією існуючих даних.

Управління доступом

SQL забезпечує синхронізацію обробки бази даних різними прикладними програмами, захист даних від несанкціонованого доступу.

Розподіл даних

SQL може координувати роботу конкуруючих користувачів, забезпечуючи їхню одночасну роботу з базою даних.

Забезпечення цілісності даних

З використанням SQL можна визначити досить складні обмеження цілісності, задоволення яких буде перевірятися за всіх модифікацій бази даних.

SQL в стандарті ANSI/ISO є реляційно-повною мовою, і СКБД, що підтримують його, задовольняють всі правила Кодда.

Структура основного виразу

Основною функцією, що використовується при розробці додатків, є вибір необхідних даних з бази даних до прикладної програми. У загальному випадку таку операцію називають ***запитом***. Вираз запиту наведено на рис. 8.5.

Рис. 8.5. SQL-запит

Результатом запиту завжди є таблиця необхідної структури з необхідними даними. У окремому випадку таблиця може містити одну колонку і один рядок, тобто являти собою просте значення. Результуюча таблиця може мати рядки, що повторюються, видалити які можна, указавши DESTINCT в позиції SELECT. Розглянемо призначення позицій в структурі речення запиту.

SELECT визначає, які колонки повинна мати результуюча таблиця. Символ «*» означає, що результуюча таблиця має включати в себе всі колонки всіх таблиць, на основі яких будується запит. Якщо кілька початкових таблиць мають однойменні колонки, то ідентифікувати потрібну колонку можна, указавши її повне ім’я, складене з імені таблиці і імені колонки. Як колонку результуючого відношення може бути вказано будь-який вираз, що використовує набір стандартних функцій, наприклад, COUNT(), AVG(), SUM() та інші. Таким чином, можна отримувати колонки, значення в яких будуть обчислюватися певним зазначеним чином, наприклад (колонка1 + колонка2 * avg(колонка3) — 30).

FROM визначає список таблиць, на основі яких будується запит. Оскільки SQL є структурною мовою, то він допускає використання вкладених SELECT-конструкцій. Базова таблиця може бути вказана якимось SELECT-виразом.

WHERE дозволяє вказати умову, якій повинні задовольняти значення в рядках результуючого відношення. Наприклад, (колонка1 < колонка2 and колонка2 > avg(колонка3)). Умова може використовувати перевірку порожніх (невизначених) значень, перевірку входження значення в безліч значень іншої таблиці.

GROUP BY визначає список імен колонок, по однакових значеннях яких буде виконуватися згрупування рядків. Тобто рядки з однаковими значеннями у вказаних колонках будуть представлені в результуючому відношенні одним рядком. Але якщо в SELECT вказано, наприклад, функцію AVG() або SUM(), то вона застосовується до всіх рядків групи.

HAVING дозволяє відібрати з безлічі згрупованих рядків тільки ті, які задовольняють вказаній умові. Наприклад, для того, щоб відібрати найменування студентських груп, що містять більше за 25 студентів, необхідно визначити умову (COUNT(група) > 25).

ORDER BY дозволяє визначити умову впорядкованості рядків в результуючому відношенні. Така умова являє собою список колонок, по яких необхідно упорядити рядки. Для кожного з атрибутів можна вибрати убутну або зростаючу послідовність. Два або більше SELECT-виразів з однаковою структурою результуючих таблиць можуть утворювати загальну результуючу таблицю з використанням реляційних операцій з'єднання (UNION), перетину (INTERSECT) і розподілу (DIFFERENCE).

Модифікація даних

Для зміни вмісту бази даних SQL передбачає три операції INSERT, DELETE і UPDATE (вставка рядків в таблиці, видалення рядків з таблиць і зміна значень в існуючих рядках таблиць).

Операція вставки може бути одиничною або груповою. Для одиничної вставки необхідно конкретно визначити значення колонок нового рядка.

Рис. 8.6. Операція вставки

Якщо вказано неповний перелік колонок таблиці, то колонки, що залишилися, набувають значення, що встановлене як замовчуване, або невизначеного значення (NULL) у випадку, якщо замовчуваного значення не встановлено.

Операція групової вставки передбачає додання рядків в таблицю з якоїсь іншої таблиці, вказаної явно або за допомогою SELECT-виразу.

Синтаксис виразу групової вставки є аналогічним одиничній, але замість позиції VALUES необхідно вказати певну SELECT-конструкцію.

Операція вилучення є груповою, тобто застосовується до всіх рядків таблиці, що задовольняють необхідну умову. На відміну від запис-орієнтованих мов маніпулювання даними, SQL не використовує поняття поточного рядка в таблиці, передбачаючи, що будь-який рядок може бути однозначно ідентифіковано в таблиці за допомогою значення первинного ключа.

Рис. 8.7. Операція вилучення

Якщо позиція WHERE відсутня, то вилучаються всі рядки таблиці.

Умова може використати функцію перевірки входження значення в якусь іншу таблицю, задану за допомогою SELECT-конструкції.

Рис. 8.8. Операція відновлення

Операція відновлення значень в рядках таблиці також є груповою.

Позиція WHERE визначає, до яких рядків потрібно застосувати операцію відновлення. За умов, аналогічно DELETE, можна використати SELECT-конструкції.

У SQL велику увагу приділяють забезпеченню цілісності даних при виконанні операцій оновлення даних, передбачено можливість обліку спеціальних обмежень цілісності. Будь-які операції, що порушують такі обмеження, відхиляються.

Найчастішим прикладом обмеження цілісності є обмеження на діапазон припустимих значень в таблицях. Дуже часто значення в таблиці є коректними тільки в тому випадку, коли вони присутні в одній або більше таблицях, логічно пов’язаних між собою.

При виконанні вилучення або оновлення рядків для забезпечення цілісності даних іноді необхідно виконувати певні супутні операції в інших логічно пов’язаних таблицях. Наприклад, вилучення рядків в одній таблиці може супроводитися вилученням пов’язаних рядків в одній або більше таблицях. Може також виникнути необхідність замінити певні значення пов’язаних рядків іншої таблиці на невизначені. При цьому такі дії можуть виконуватися рекурсивно для доволі складних багатотабличних структур.

Такого роду обмеження цілісності визначаються при створенні окремих таблиць і визначенні структури бази даних.

Групу операцій модифікації даних, що мають логічно закінчене значення і після повного виконання яких база даних залишиться коректною, називають трансакцією. У SQL передбачені засоби управління трансакціями, що дозволяють відстежувати виконання трансакцій, обробляти виникаючі помилки і координувати обробку бази даних декількома додатками або користувачами в паралельному режимі.

Твердження COMMIT означає вдале закінчення поточної трансакції і початок нової. Твердження ROLLBACK вказує на необхідність виконання зворотного відкату, тобто автоматичного відновлення стану даних, що змінювалися, на момент початку трансакції.

У більшості випадків координація роботи в багатокористувацькому режимі виконується за допомогою механізму блокування (монопольного захоплення деякої частини бази даних). Виконувати блокування можна автоматично, блокуючи дані деякою трансакцією, як тільки до них відбувається звертання, і звільняти їх при обробці COMMIT і ROLLBACK.

В SQL можна блокувати таблиці в монопольному режимі (читання і запис з боку інших трансакцій відкладаються до моменту закінчення трансакції — песимістичне блокування) або в режимі розділення (відкладаються тільки оновлення з боку інших трансакцій — оптимістичне блокування).

Рис. 8.9. Блокування таблиці в монопольному режимі

Створення баз даних

Мова SQL поєднує в собі можливості мови маніпулювання даними і мови визначення даних. Мова визначення даних дозволяє:

- описувати і створювати нові таблиці;
- вилучати визначення таблиць й самі таблиці, якщо вони більше не потрібні;
- змінювати структуру існуючих таблиць без втрат даних в цих таблицях;
- визначати віртуальні таблиці, що будуються і постійно підтримуються на основі раніше визначених таблиць;
- визначати правила забезпечення секретності даних;
- створювати індекси за ключами для швидкого доступу до даних, що зберігаються;
- управляти фізичним зберіганням даних.

Для визначення нового об’єкта бази даних використовується ключове слово CREATE, для вилучення — DROP, для зміни — ALTER.

Для створення таблиці вираз має такий вигляд:

Рис. 8.10. Створення таблиці

Фрагмент визначення колонок:

Рис. 8.11. Визначення колонок

Позиція NOT NULL вказує на необхідність контролю за тим, щоб в кожному рядку таблиці значення за даною колонкою було визначене. Якщо позиції NOT NULL і WITH DEFAULT відсутні, то значенням за замовчуванням є невизначене (NULL) значення. Якщо указати WITH DEFAULT, то для значення за умовчанням будуть використані стандартні константи, відповідні типу значення. Специфічне значення за замовчуванням можна визначити за допомогою ключового слова DEFAULT, за яким указати необхідну константу.

Фрагмент визначення первинного ключа визначає список колонок, створюючих ключ.

Рис. 8.12. Визначення первинного ключа

Зовнішні ключі використовуються для логічного поєднання декількох таблиць в певне уявлення. Визначаючи зовнішній ключ, можна встановити відповідність рядків між різними таблицями. Логічне поєднання виконується для рядків, що мають значення, які збігаються, у вказаних колонках.

Рис. 8.13. Визначення зовнішнього ключа

Позиція ON DELETE визначає операції, супутні вилученню рядка в таблиці. При цьому може постати необхідність у вилученні відповідного рядка в пов'язаній таблиці (RESTRICT), виконати рекурсивне вилучення в безпосередньо пов'язаній таблиці і всіх непрямо пов'язаних (CASCADE), очистити значення відповідних колонок у пов'язаних рядках таблиці (SET NULL).

Обмеження на унікальність рядків у таблиці вказується аналогічно визначенню первинного ключа.

Рис. 8.14. Обмеження на унікальність рядків

Створена таблиця може зберігатися на диску різними способами, якими можна управляти. Можна розподіляти таблиці бази даних в декількох спеціальних областях або кількох файлах, підтримувати декілька паралельних копій таблиць на різних носіях і т. ін.

Вилучення таблиці є найбільш простою операцією.

Рис. 8.15. Вилучення таблиці

Реструктуризація таблиці полягає в зміні складу колонок, ключів (рис. 8.16).

Рис. 8.16. Реструктуризація таблиці

Повне ім'я таблиці в базі даних включає в себе префікс — ім'я власника. Для зручності ідентифікації невласних таблиць в SQL передбачено можливість визначення синонімів.

Рис. 8.17. Визначення синоніма

Використовувати синоніми можна в усіх виразах SQL.

Вилучити визначення синоніма можна за допомогою DROP.

Рис. 8.18. Вилучення визначення синоніма

Для швидкої обробки SELECT-конструкцій можна визначити індекси — спеціальні пошукові структури, що зберігають значення деяких колонок з посиланнями на відповідні рядки таблиці. Аналізуючи структуру, що використовується в додатку SELECT-конструкцій, програміст досить просто може виділити необхідні індекси.

Рис. 8.19. Створення індексу

ASC/DESC вказує на вигляд впорядкованості значень в індексі (за зростанням або спаданням).

Вилучити індекс можна за допомогою простого DROP-виразу.

У ANSI/ISO-стандарт структура бази даних має ієрархію — база даних (схема), підсхема користувача (безліч таблиць у поданні користувача), таблиця. Таким чином, повне ім’я таблиці складається з трьох імен, розділених крапками.

Погляди на базу даних, або, інакше, віртуальні відносини, визначаються на рівні розробки прикладних застосувань баз даних і являють собою зручний засіб для незалежної розробки прикладних застосувань різними програмістами, реструктуризації баз даних без зміни прикладних програм, простого забезпечення таємності даних і надійності додатків.

Віртуальна таблиця є результатом неявного виконання якоїсь SELECT-конструкції. Рядки такої таблиці утворяться або в момент звернення до них, або, одного разу сформувавшись, постійно змінюються при зміні відносин, що входять у відповідну SELECT-конструкцію.

Рис. 8.20. Створення віртуальної таблиці

Позиція CHECK OPTION вказує на необхідність контролю за вмістом віртуальної таблиці під час оновлення базових відношень.

Важливим моментом при створенні баз даних є визначення привілеїв користувачів (переліку дозволених операцій при використанні бази даних). При роботі з базою даних користувач проходить реєстрацію, вказуючи своє ім’я і пароль. На основі імені користувача СКБД контролює дії користувача і відхиляє несанкціоновані операції.

Встановити привілеї користувачеві можна виразом GRANT (рис. 8.21), а зняти — виразом REVOKE (рис. 8.22).

Рис. 8.21. Надання привілеїв користувачеві

Рис. 8.22. Зняття привілеїв з користувача

Привілеї виконання операцій можна призначити конкретному користувачеві або всім користувачам, указавши PUBLIC. Присутність WITH GRANT OPTION означає, що користувачеві разом з правами виконання операцій дозволяється передавати частину своїх прав іншим користувачам.

Системний каталог

Використання SQL-виразів для визначення даних тільки в рамках прикладних програм ускладнює розробку додатка кількома програмістами і унеможливлює практично повноцінний супровід інформаційних комплексів. ANSI/ISO-стандарт передбачає використання системного каталогу, в якому фіксуються всі визначення й зміни структури бази даних. У мінімальному обсязі системний каталог має містити:

- опис таблиць (перелік усіх таблиць бази даних з вказівкою імені таблиці і її власника, числа колонок таблиці і т. ін.);
- опис колонок (перелік всіх колонок, що використовуються в базі даних, з вказівкою імені таблиці-власника, типу значення, дозволу зберігання невизначених значень і т. ін.);
- опис користувачів-власників (перелік власників з вказівкою їх імен і паролів);
- опис віртуальних таблиць (аналогічно визначенню реальних з визначенням відповідного SQL-виразу);
- опис привілеїв (перелік усіх активних GRANT-конструкції).

Системний каталог підтримується в таблицях із зарезервованими іменами. Ці таблиці можуть бути доступні для використання в SELECT-конструкціях.

Для кожної таблиці й колонки можна визначити змістовні мітки і розширені коментарі, які зручно використати в додатках, що розробляються. У SQL для цієї мети передбачені конструкції COMMENT і LABEL.

Програмування на SQL

SQL використовується в діалоговому і пакетному режимах. Діалоговий режим передбачає інтерпретацію кожної конструкції і її виконання. Пакетний режим передбачає написання програм, що реалізовують обробку бази даних в прикладних інтересах. Дворежимне використання SQL дуже зручно для попереднього інтерактивного відлагодження програм і реалізації остаточної версії додатка у вигляді більш швидкодіючої програми. Структура мови й її можливості в основному ідентичні в обох режимах.

Для ефективної розробки прикладних додатків з використанням SQL існує декілька варіантів реалізації мови і, відповідно, СКБД. Кожний з варіантів передбачає використання своєї техніки програмування. Найбільш поширеним є вбудований SQL, що реалізовується в статичному і динамічному варіантах. Альтернативою вбудованому SQL, що заявила про себе останнім часом, є зовнішній SQL, що передбачає взаємодію додатків з базою даних у режимі клієнт/сервер.

Вбудований SQL передбачає використання якоїсь основної мови програмування (наприклад, C, PASCAL, ADA, COBOL, FORTRAN, ASSEMBLER) і SQL-препроцесора. Реалізація вбудованого SQL регламентована IBM-стандартом SQL-продуктів.

Текст програми складається з довільної суміші команд основної мови і команд SQL зі спеціальним префіксом, наприклад, EXEC SQL. Структуру SQL-виразів розширено для використання змінних основної мови в SQL-конструкціях.

SQL-препроцесор видозмінює текст програми відповідно до вимог компілятора основної мови програмування. При цьому в передбаченій формі розставляються звернення до процедур і функцій бібліотеки, що реалізовує обробку SQL і основних функції СКБД. Після компіляції програми й її компонування додаток являє собою самостійний модуль.

Для роботи препроцесора необхідно в кожній програмі визначати структуру таблиць, що використовуються, наприклад, за допомогою конструкції CREATE TABLE. Вводиться додаткова інструкція DECLARE, що надає можливість посилатися на опис таблиці з декількох SQL-виразів в програмах, що розробляються. Структура такої інструкції аналогічна до CREATE, але містить тільки інформацію, необхідну для роботи SQL-препроцесора.

Рис. 8.23. Надання можливості посилатися на опис таблиці з кількох SQL-виразів

Обробка помилок у вбудованому SQL виконується на кроці компіляції і при виконанні. На кроці компіляції програми помилки в SQL-виразах виявляються препроцесором. При виконанні додатка код завершення кожного SQL-виразу міститься в спеціальній змінній основної мови програмування SQLCODE.

Реалізація SQL

Статичний варіант вбудованого SQL передбачає виконання наступних дій з боку СКБД при обробці кожного здійснюваного SQL-виразу:

- синтаксичний розбір речення;
- ратифікація виразу, визначення семантики;
- оптимізація вираження запиту;
- генерація плану реалізації запиту;
- виконання плану реалізації запиту.

Якщо певний запит виконується циклічно, то більшість операцій можна виконати заздалегідь, а в циклі здійснювати тільки виконання плану реалізації запиту. У попередній підготовці реалізації запитів і полягає основна ідея концепції динамічного вбудованого SQL.

Динамічний SQL дозволяє виконати всі етапи SQL-виразу за допомогою префікса EXECUTE IMMEDIATE, тобто включає в себе статичну версію виконання.

Двоетапне виконання SQL-виразу полягає в попередньому створенні плану реалізації запиту і виконанні побудованого плану. Префікс PREPARE в SQL-виразі використовується для побудови плану реалізації запиту, який формується препроцесором під час компіляції програми. Пропозиція EXECUTE виконує план реалізації запиту під час виконання додатка.

Статичний і динамічний варіанти SQL передбачають надання програмісту спеціальної бібліотеки процедур і функції виконуючої функції СКБД. У цьому випадку кожен з додатків містить в собі практично повний набір процедур СКБД. Якщо безліч додатків працюють із загальною базою даних в мережі, то кожний запит виконується на тому комп’ютері, на якому функціонує додаток-власник. Але більш ефективно виконувати запит не там, де працює додаток, а там, де зберігаються дані для запиту. Виграш у часі досягається скороченням обсягу проміжних даних, що передаються по мережі.

Концепція SQL API (application programming interface, інтерфейс програмування додатків) передбачає взаємодію додатка (клієнта) з сервером бази даних. Додаток передає серверу бази даних SQL-вираз, а сервер повертає додатку результати виконання запиту. До складу додатка в цьому випадку входить тільки комунікаційний модуль, а СКБД міститься тільки у програмі серверу.

Такий підхід не вимагає використання препроцесора, а бібліотека СКБД для клієнта представлена у вигляді процедур, що виконують чотири основні групи дій:

- з’єднання з сервером і відключення від нього;
- передавання повідомлення (символьного рядка SQL-команди) серверу;
- обробка помилок виконання;
- прийом таблиці-результату (порядковий скролінг).

Концепція клієнт/сервер, аналогічно вбудованому SQL, використовує ідею попередньої підготовки плану реалізації запиту; тут план будується під час виконання, але заздалегідь, тобто одноразово перед виконанням запиту. В даному підході є один нюанс: будувати план виконання запиту (перший раз виконувати запит) слід в момент найменшого завантаження серверу, інакше є загроза, що побудований план не буде оптимальним.

Перспективи мови SQL

Мова SQL має певні недоліки, які мають намір усунути в новому стандарті SQL3. Незважаючи на компактність і повноту мови, SQL є досить складним для розуміння. Більшість програмістів звикли розробляти додатки, обробляти дані на рівні записів. Для програмування на SQL, що оперує таблицями, необхідна певна перебудова мислення. Проблема полягає швидше не в нестачах ядра SQL, у відсутності виразів і сервісу, що спрощують процес програмування.

Існує ряд типів запитів, які дуже складно реалізувати на SQL. Наприклад, це побудова таблиць на основі зовнішнього з'єднання, які досить просто утворюються з використанням set relation в мовах xBase. Досить складно виконувати рекурсивну побудову результуючих таблиць, наприклад, для отримання переліку деталей, що входять в деякий багаторівневий виріб.

Передбачається розвиток SQL у напрямі концепції об'єктно-орієнтованого програмування. Однак, за деякими прогнозами, реалізація таких планів буде мати немало проблем, пов'язаних з узгодженням принципу клієнт/сервер в SQL і принципу взаємодії об'єктів в об'єктно-орієнтованому програмуванні.

Швидше за все, вказані недоліки не знизять популярності SQL. Адже сьогодні SQL — єдиний претендент на стандарт мови прямого обміну запитами в неоднорідних інформаційних системах.

Приклади запитів та програм з використанням мови SQL наведено в дод. 1.

Контрольні питання

1. *Наведіть перелік основних типів структур діалогу.*
2. *Визначте суть діалогу типу питання — відповідь.*
3. *Охарактеризуйте структуру діалогу типу меню.*
4. *Який порядок побудови діалогу типу екранної форми?*
5. *Перелічіть переваги та недоліки діалогу на основі мови команд.*
6. *Які основні функції сучасних СКБД реалізовує SQL?*
7. *Для чого і як застосовується оператор SELECT?*
8. *Які існують операції модифікації даних?*
9. *Як можна створити таблицю та змінити її структуру мовою SQL?*
10. *Для чого застосовуються курсори в SQL? Їх синтаксис.*