Міністерство освіти і науки України

Відокремлений структурний підрозділ
«Ковельський промислово-економічний фаховий коледж
Луцького національного технічного університету»

ОРГАНІЗАЦІЯ БАЗ ДАНИХ ТА ЗНАНЬ

Методичні вказівки до самостійної роботи
для здобувачів освітньо-кваліфікаційного ступеня фаховий молодший бакалавр спеціальності 122 Комп'ютерні науки
денної форми здобуття освіти

Ковель 2025

УДК 004.451.9
О-60

Затверджено до видання методичною радою ВСП «КПЕФК ЛНТУ»
протокол №_____ від « ___» _____ 2025 року
Голова методичної ради ВСП «КПЕФК ЛНТУ» _____ Ігор ІЛЮШИК

Розглянуто і схвалено на засіданні циклової комісії з комп'ютерних наук
ВСП «Ковельський промислово-економічний фаховий коледж ЛНТУ»,
Протокол № 4 від «20» січня 2025 року.
Голова випускної комісії_____Олександр ПРИСАДА

Електронна копія друкованого видання передана до книжкового фонду бібліотеки ВСП «КПЕФК ЛНТУ»
Завідувачка бібліотеки_____Неля ТЕЛЮЧИК

Укладач:_____ Олександр ПРИСАДА, викладач ВСП «КПЕФК ЛНТУ»

Рецензент: _____Степан БАБАРИКА, к.т.н., викладач ВСП «КПЕФК ЛНТУ»

Відповідальний за випуск:_____Леся ПРОКОПЧУК, методист ВСП «КПЕФК ЛНТУ»
О-60
Організація баз даних та знань»[Текст] Методичні вказівки до самостійної роботи для здобувачів освітньо-кваліфікаційного ступеня фаховий молодший бакалавр спеціальності 122 Комп'ютерні науки денної форми здобуття освіти
/уклад. Олександр ПРИСАДА. – Ковель: ВСП «КПЕФК ЛНТУ», 2025. – 115с.
Видання містить методичні вказівки до самостійної роботи з освітнього компонента «Організація баз даних та знань» для здобувачів освітньо-кваліфікаційного рівня фаховий молодший бакалавр спеціальності 122 Комп'ютерні науки денної форми здобуття освіти. Навчальний матеріал дозволяє отримати поглиблені знання та розши рити професійні компетенції при вивченні освітньої компоненти «Організація баз даних та знань»

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Поняття інформаційної технології

2. Навчальна мета. Студент повинен:

знати: визначення інформаційної технології, етапи розвитку, с кладові і інструментарій ІТ.

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Що таке інформаційна технологія
2) Етапи розвитку інформаційних технологій
3) Складові інформаційної технології
4) Інструментарій інформаційної технології

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Основні ознаки етапів розвирку ІТ? В чому різниця
2) Який інструментарів використовується в ІТ?

5.Контроль засвоєння теми

Різнорівневі питання

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

1 ПОНЯТТЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

1.1 Що таке інформаційна технологія

Технологія - це комплекс наукових і інженерних знань, реалізованих у прийомах праці, наборах матеріальних, технічних, енергетичних, трудових факторів виробництва, способах їх з'єднання для створення продукту або послуги, що відповідають певним вимогам. Тому технологія нерозривно пов'язана з машинізацією виробничого або невиробничого, насамперед управлінського процесу. Управлінські технології грунтуються на застосуванні комп'ютерів і телекомунікаційної техніки.

Згідно з визначенням, прийнятим ЮНЕСКО, інформаційна технологія - це комплекс взаємозалежних, наукових, технологічних, інженерних дисциплін, що вивчають методи ефективної організації праці людей, зайнятих обробкою і зберіганням інформації; обчислювальну техніку і методи організації і взаємодії з людьми і виробничим устаткуванням, практичні додатки, а також пов'язані з усім цим соціальні, економічні і культурні проблеми. Самі інформаційні технології вимагають складної підготовки, великих початкових витрат і наукомісткої техніки. Їхнє введення повинно починатися зі створення математичного забезпечення, формування інформаційних потоків у системах підготовки фахівців.

1.2 Етапи розвитку інформаційних технологій

Існує кілька точок зору на розвиток інформаційних технологій з використанням комп'ютерів, що визначаються різноманітними ознаками поділу.

Загальним для всіх викладених нижче підходів є те, що з появою персонального комп'ютера почався новий етап розвитку інформаційної технології. Основною метою стає задоволення персональних інформаційних потреб людини як для професійної сфери, так і для побутової.

Ознака поділу - вид задач і процесів обробки інформації

1-й етап (60 - 70-ті рр.) - Опрацювання даних в обчислювальних центрах у режимі колективного користування. Основним напрямком розвитку інформаційної технології була автоматизація операційних рутинних дій людини.

2-й етап (з 80-х рр.) - Створення інформаційних технологій, спрямованих на вирішення стратегічних завдань.

Ознака поділу - проблеми, які стоять на шляху інформатизації суспільства

1-й етап (до кінця 60-х рр.) Характеризується проблемою опрацювання великих обсягів даних в умовах обмежених можливостей апаратних засобів.

2-й етап (до кінця 70-х рр.) Пов'язаний з поширенням ЕОМ серії 1ВМ / 360. Проблема цього етапу - відставання програмного забезпечення від рівня розвитку апаратних засобів.

3-й етап (з початку 80-х рр.) - Комп'ютер стає інструментом непрофесійного користувача, а інформаційні системи - засобом підтримки прийняття його рішень. Проблеми-максимальне задоволення потреб користувача і створення відповідного інтерфейсу для роботи в комп'ютерному середовищі.

4-й етап (з початку 90-х рр.) - Створення сучасної технології міжустановчих зв'язків та інформаційних систем. Проблеми цього етапу вельми багато. Найбільш істотними з них є:

- укладання угод і встановлення стандартів, протоколів для комп'ютерного зв'язку;
- організація доступу до стратегічної інформації;
- організація захисту і безпеки інформації.

Ознака поділу - перевага, яку надає комп'ютерна технологія

1 -й етап (з початку 60-х р.р ..) характеризується досить ефективним опрацюванням інформації при виконанні рутинних операцій з орієнтацією на централізоване колективне використання ресурсів обчислювальних центрів. Основним критерієм оцінки ефективності інформаційних систем була різниця між витраченими на розробку і зекономленими в результаті впровадження коштами. Основною проблемою на цьому етапі була психологічна - погана взаємодія користувачів, для яких створювалися інформаційні системи, і розроблювачів через розходження їхніх поглядів і поні-манія розв'язуваних проблем. Як наслідок цієї проблеми, створювалися системи, які користувачі погано сприймали і, незважаючи на їх досить великі можливості, не використовували повною мірою.

2-й етап (з середини 70-х рр.) Пов'язаний з появою персональних комп'ютерів. Змінився підхід до створення інформаційних систем-орієнтація зміщується в бік індивідуального користувача для підтримки прийнятих ним рішень. Користувач зацікавлений у проведеній розробці, налагоджується

контакт із розроблювачем, виникає порозуміння між обома групами спеціалістів. На цьому етапі використовується як централізована опрацювання даних, характерне для першого етапу ,. так і децентралізована, що базується на вирішенні локальних задач і роботі з локальними базами даних на робочому місці користувача.

3-й етап (з початку 90-х рр.) Пов'язаний з поняттям аналізу стратегічних переваг у бізнесі і заснований на досягненнях телекомунікаційної технології розподіленого опрацювання інформації. Інформаційні системи мають своєю метою не просто збільшення ефективності опрацювання даних і допомога керівнику. Відповідні інформаційні технології повинні допомогти організації вистояти в конкурентній боротьбі й одержати перевагу.

Ознака поділу - види інструментарію технології

1-й етап (до другої половини XIX ст.) - "Ручна" інформаційна технологія інструментарії якої складали: перо. чорнильниця, книга Комунікації здійснювалися ручним способом шляхом виправлення через пошту листів, пакетів, депеш. Основна мета технології - представлення інформації в потрібній формі.

2-й етап (з кінця XIX в.) - "Механічна" технологія, інструментарій якої складали: друкарська машинка, телефон, диктофон, оснащена більш досконалими засобами доставки пошта. Основна мета технології - представлення інформації в потрібній формі більш зручними засобами,

3-й етап (40 - 60-ті рр. XX ст.) - "Електрична" технологія, інструментарій якої становили: великі ЕОМ і відповідне програмне забезпечення, електричні друкарські машинки, ксерокси, портативні диктофони.

Етапі відбувається зміна мети технології. Акцент в інформаційній технології починає переміщуватись з форми представлення інформації на формування її змісту.

4-й етап (з початку 70-х рр.) - "Електронна" технологія, основним інструментарієм якої стають великі ЕОМ і створені на їхній базі автоматизовані системи управління (АСУ) і інформаційно-пошукові системи (ІПС) , оснащені широким спектром базових і спеціалізованих програмних комплексів. Центр ваги технології ще більш зміщується на формування змістовної сторони інформації для управлінського середовища різних сфер суспільного життя, особливо на організацію аналітичної роботи. Безліч об'єктивних і суб’єкгивних факторів не дозволили вирішити поставлені перед новою концепцією інформаційної технології завдання, Проте був здобутий досвід формування змістовної сторони управлінської інформації і підготовлена фахова, психологічна і соціальна база для переходу на новий етап розвитку технології,

5-й етап (з середини 80-х рр.) - "Комп'ютерна" ("нова") технологія, основним інструментарієм якої є персональний комп'ютер із широким спектром стандартних програмних продуктів різного призначення. На цьому етапі відбувається процес персоналізації АСК, що проявляється у створенні систем підтримки прийняття рішень певними фахівцями. Подібні системи мають вбудовані елементи аналізу і інтелекту для різних рівнів управління, реалізуються на персональному комп'ютері і використовують телекомунікації. У зв'язку з переходом на мікропроцесорну базу суттєвим змінам піддаються і технічні засоби побутового, культурного та іншого призначень.

Починають широко використовуватися в різних областях глобальні і локальні комп'ютерні мережі.

1.3 Складові інформаційної технології

Використовувані у виробничій сфері такі технологічні поняття, як норма, нормативів, технологічний процес, технологічна операція і т.п., можуть застосовуватися і в інформаційній технології. Перш ніж розробляти ці поняття в будь-який технології, в тому числі і в інформаційній, завжди слід починати з визначення мети. Потім плтрібно спробувати провести структурування всіх передбачуваних дій, що призводять до наміченої мети, і вибрати необхідний програмний інструментарій.

Необхідно розуміти, що освоєння інформаційної технології і подальше її використання повинні бути зведені до того, що потрібно спочатку добре оволодіти набором елементарних операцій, кількість яких обмежена. З цього обмеженого числа елементарних операцій у різних комбінаціях складається дія, а з дій, також у різних комбінаціях, складаються операції, що визначають той

або інший технологічний етап. Сукупність технологічних етапів утворює технологічний процес (технологію). Він може починатися з будь-якого рівня і не включати, наприклад, етапи або операції, а складатися тільки з дій. Для реалізації етапів технологічного процесу можуть використовуватися раз-ні програмні середовища.

Інформаційна технологія, як і будь-яка інша, повинна відповідати таким вимогам:

• забезпечувати високий рівень розчленування всього процесу обробки інформації на етапи (фази), операції, дії;

• включати весь набір елементів, необхідних для досягнення поставленої мети;

• мати регулярний характер. Етапи, дії, операції технологічного процесу можуть бути стандартизовані й уніфіковані, що дозволить більш ефективно здійснювати цілеспрямоване управління інформаційними процесами.

1.4 Інструментарій інформаційної технології

Реалізація технологічного процесу матеріального виробництва здійснюється з по-міццю різних технічних засобів, до яких відносяться: устаткування, верстати, інструменти, конвеєрні лінії і т.п.

За аналогією і для інформаційної технології повинно бути щось подібне. Такими технічними засобами виробництва інформації буде апаратне, програмне і математичне забезпечення цього процесу. З їх допомогою проводиться переробка первинної інформації в інформацію нової якості. Виділимо окремо з цих засобів програмні продукти і назвемо їх інструментарієм, а для більшої чіткості можна його конкретизувати, назвавши програмним інструментарієм інформаційної технології.

Визначимо це поняття:

Інструментарій інформаційної технології - один або кілька взаємопов'язаних програмних продуктів для певного типу комп’ютера, технологія роботи в яких дозволяє досягти поставленої користувачем мети.

В якості інструментарію можна використовувати такі поширені види програмних продуктів для персонального комп'ютера: текстовий процесор (редактор), настільні видавницькі системи, електронні таблиці, системи управління базами даних, електронні записники, електронні календарі, інформаційні системи функціонального призначення (фінансові, бухгалтерські, для маркетингу та ін.). експертні системи і т.д.

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Види сучасних інформаційних технологій

2. Навчальна мета. Студент повинен:

знати: види сучасних інформаційних технологій і їх зарактеристику

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Інформаційна технологія обробки даних
2) Інформаційна технологія управління
3) Інформаційна технологія прийняття рішень
4) Інформаційна технологія експертних систем

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Основні характеристики технології обробки даних
2) Відміни технології експертних систем і технології прийняття рішень
3) Використання технології управління

5.Контроль засвоєння теми

Різнорівневі питання

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)

2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X

3. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

2 ВИДИ СУЧАСНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

2.1 Інформаційна технологія обробки даних

Характеристика і призначення

Інформаційна технологія обробки даних призначена для вирішення добре структурованих завдань, по яких є необхідні вхідні дані і відомі алгоритми і інші стандартні процедури їх обробки. Ця технологія застосовується на рівні операційної (виконавчої) діяльності персоналу невисокої кваліфікації з метою автоматизації деяких рутинних постійно повторюваних операцій управлінської праці. Тому впровадження інформаційних технологій і систем на цьому рівні істотно підвищить продуктивність праці персоналу, звільнить його від рутинних операцій, можливо, навіть призведе до необхідності скорочення чисельності працівників.

На рівні операційної діяльності вирішуються такі завдання:

• обробка даних про операції, які здійснює фірма;

• створення періодичних контрольних звітів про стан справ у фірмі;

• отримання відповідей на всілякі поточні запити й оформлення їх у вигляді паперових документів або звітів.

Прикладом може послужити щоденний звіт про надходження і видачу готівки коштів банком, що формується з метою контролю балансу готівки, або ж запит до бази даних по кадрах, який дозволить отримати дані про вимоги, що висуваються до кандидатів на певну посаду.

Існує кілька особливостей, пов'язаних з обробкою даних, що відрізняють дану технологію від усіх інших:

• виконання необхідних фірмі задач по обробці даних. Від кожної фірми передбачено законом наявності та збереження даних про свою діяльність, які можна використовувати як засіб забезпечення і підтримки контролю на фірмі. Тому в будь-якій фірмі обов'язково повинна бути інформаційна система обробки даних і розроблена відповідна інформаційна технологія;

• вирішення тільки добре структурованих задач, для яких можна розробити алгоритм;

• виконання стандартних процедур обробки. Існуючі стандарти визначають типові процедури опрацювання даних і регламентують їхнє дотримання організаціями усіх видів;

• виконання основного обсягу робіт в автоматичному режимі з мінімальною участю людини;

• використання деталізованих даних. Записи про діяльність фірми мають детальний (докладний) характер, що допускає проведення ревізій. У процесі ревізії діяльність фірми перевіряється хронологічно від початку періоду до його кінця і від кінця до початку;

• акцент на хронологію подій;

• вимога мінімальної допомоги у вирішенні проблем з боку спеціалістів інших рівнів.

Збереження даних . Багато дані на рівні операційної діяльності необхідно зберігати для подальшого використання або тут же, або на іншому рівні. Для їх зберігання створюються бази даних.

Створення звітів (документів). В інформаційній технології обробки даних необхідно створювати документи для керівництва і працівників фірми, а також для зовнішніх партнерів. При цьому документи можуть створюватися як за запитом або в зв'язку з проведеною фірмою операцією, так і періодично наприкінці кожного місяця, кварталу або року.

2.2 Інформаційна технологія управління

Характеристика і призначення:

Метою інформаційної технології керування є задоволення інформаційних потреб усіх без винятку співробітників фірми, що мають справу з прийняттям рішень. Вона може бути корисна на будь-якому рівні управління.

Ця технологія орієнтована на роботу в середовищі інформаційної системи керування і використовується при більш поганій структурованості розв'язуваних задач, якщо їх порівнювати з задачами, які розв'язуються за допомогою інформаційної технології обробки даних.

Інформаційна технологія керування ідеально підходять для задоволення подібних інформаційних потреб робітників різних функціональних підсистем (підрозділів) або рівнів управління фірмою. Поставляється нею, містить відомості про минуле, дійсне і ймовірне майбутнє фірми. Ця інформація має вигляд регулярних або спеціальних управлінських звітів.

Для прийняття рішень на рівні управлінського контролю інформація повинна бути подана в агрегованому вигляді, так, щоб проглядалися тенденції зміни даних, причини відхилень і можливі рішення. На цьому етапі розв'язуються такі задачі опрацювання даних:

• оцінка планованого стану об'єкта управління;

• оцінка відхилень від планованого стану;

• виявлення причин відхилень;

• аналіз можливих рішень і дій.

Інформаційна технологія керування спрямована на створення різноманітних видів звітів. Регулярні звіти створюються відповідно до встановленого графіка, що визначає час їх створення, наприклад місячний аналіз продажів компанії.

Спеціальні звіти створюються за вимогою керівників, або коли в компанії відбулося щось незаплановане. І ті, і інші види звітів можуть мати форму підсумкових, порівняльних і надзвичайних звітів.

У підсумкових звітах дані об'єднані в окремі групи, відсортовані і подані у вигляді проміжних і остаточних результатів по окремих полях.

Порівняльні звіти містять дані, отримані з різних джерел або класифіковані по різноманітних ознаках і використовуються для порівняння.

Надзвичайні звіти містять дані виняткового (надзвичайного) характеру.

Використання звітів для підтримки керування є особливо ефективним при реалізації так званого керування по відхиленнях. Управління за відхиленнями припускає, що головним змістом одержуваних спеціалістом даних повинні бути відхилення стану господарської діяльності фірми від деяких встановлених стандартів (наприклад, від її запланованого стану). При використанні на фірмі принципів керування по відхиленнях до звітів, які створюються пред'являються наступні вимоги:

• звіт необхідно створювати тільки тоді, коли відхилення відбулося

• відомості у звіті повинні бути відсортовані за значенням критичного для даного відхилення показника;

• усі відхилення бажано показати разом, щоб спеціаліст міг уловити існуючий між ними зв'язок;

• у звіті необхідно показати, кількісне відхилення від норми.

Основні компоненти

Вхідна інформація надходить із систем операційного рівня. Вихідна інформація формується у вигляді управлінських звітів у зручному для ухвалення рішення вигляді. Вміст бази даних за допомогою відповідного програмного забезпечення перетворюється в періодичні і спеціальні звіти, що надходять до спеціалістів, що приймають участь в прийнятті рішень в організації. База даних, яка використовується для отримання зазначеної інформації, повинна складатися з двох елементів:

1) даних, що накопичуються на основі оцінки операцій, проведених фірмою;

2) планів, стандартів, бюджетів та інших нормативних документів, що визначають планований стан об'єкта керування (підрозділи фірми).

2.3 Інформаційна технологія підтримки прийняття рішень

Система управління інтерфейсом

Ефективність і гнучкість інформаційної технології багато в чому залежать від характеристик інтерфейсу системи підтримки прийняття рішень. Інтерфейс визначає: мову користувача; мова повідомлень комп'ютера, що організує діалог на екрані дисплея; знання користувача.

Мова користувача - це ті дії, які користувач робить у відношенні системи шляхом використання можливостей клавіатури; електронних олівців, що пишуть на екрані; джойстика; "Миші"; команд, що подаються голосом, і т.п. Найбільш простою формою мови користувача є створення форм вхідних і вихідних документів. Отримавши вхідну форму (документ), користувач заповнює його необхідними даними і вводить в комп'ютер. Система підтримки прийняття рішень робить необхідний аналіз і видає результати у вигляді вихідного документа установленої форми.

Мова повідомлень - це те, що користувач бачить на екрані дисплея (символи, графіка, колір), дані, отримані на принтері, звукові вихідні сигнали і т.п. Важливим показником ефективності інтерфейсу є обрана форма діалогу між користувачем і системою. В даний час найбільш поширеним є такі форми діалогу: запитання-відповідь режим, командний режим, режим меню, режим заповнення пропусків у виразах, запропонованих комп'ютером. Кожна форма в залежності від типу завдання,

особливостей користувача і прийнятого рішення може мати свої переваги і недоліки. Довгий час єдиною реалізацією мови повідомлень був видрукуваний чи виведений на екран дисплея звіт або повідомлення. Тепер з'явилася нова можливість уявлення вихідних даних-машинна графіка. Вона дає можливістьт створювати на екрані і папері кольорові графічні зображення в тривимірному вигляді. Використання машинної графіки, значно підвищує наочність і інтерпретуємість вихідних даних, стає все більш популярним в інформаційній технології підтримки прийняття рішень.

Знання користувача -те, що користувач повинен знати, працюючи з системою. До них відносяться не тільки план дій, що знаходиться в голові у користувача, але і підручники, інструкції, довідкові дані, що видаються комп'ютером.

Удосконалення інтерфейсу системи підтримки прийняття рішень визначається успіхами у розвитку кожного з трьох зазначених компонентів. Інтерфейс повинен мати такі можливості:

* Маніпулювати різними формами діалогу, змінюючи їх в процесі прийняття рішення щодо вибору користувача;

* передавати дані системі різними способами;

* отримувати дані від різних пристроїв системи в різному форматі;

* гнучко підтримувати (надавати допомогу за запитом, підказувати) знання користувача.

2.4 Інформаційна технологія експертних систем

Характеристика і призначення

Найбільший прогрес серед комп'ютерних інформаційних систем відзначений в області розробки експертних систем. Експертні системи дають можливість менеджеру або фахівця отримувати консультації експертів з будь-яких проблем, про які цими системами накопичені знання.

Рішення спеціальних завдань вимагає спеціальних знань. Однак не кожна компанія може собі дозволити тримати у своєму штаті експертів по всім пов'язаним з її роботою проблемам або навіть запрошувати їх щоразу, коли виникає якась проблема. Головна ідея використання технології експертних систем полягає в тому, щоб отримати від експерта його знання і, завантаживши їх у пам'ять комп'ютера, використовувати кожного разу, коли в цьому виникне необхідність. Все це робить можливим використовувати технологію експертних систем як радять систем.

Подібність інформаційних технологій, що використовуються в експертних системах і системах підтримки прийняття рішень, полягає в тому, що обидві вони забезпечують високий рівень підтримки прийняття рішень. Однак існують три суттєві відмінності:

Перша пов'язана з тим, що рішення проблеми в рамках систем підтримки прийняття рішень відображає рівень її розуміння користувачем і його можливості одержати й осмислити рішення. Технологія експертних систем, навпаки, пропонує користувачу прийняти рішення, що перевершує його можливості.

Друга відмінність зазначених технологій проявляється у здатності експертних систем пояснювати свої міркування у процесі одержання рішення. Дуже часто ці пояснення виявляються більш важливими для користувача, ніж саме рішення.

Третя відмінність пов'язана з використанням нового компонента інформаційної технології - знань.

Основні компоненти

Основними компонентами інформаційної технології, яка використовується в експертній системі, є: інтерфейс користувача, база знань, інтерпретатор, модуль створення системи

Інтерфейс користувача. Менеджер (фахівець) використовує інтерфейс для введення інформації і команд в експертну систему та одержання вихідної інформації з неї. Команди містять у собі параметри, що спрямовують процес обробки знань. Інформація зазвичай видається у формі значень, що присвоюються певним змінним.

Технологія експертних систем передбачає можливість одержувати в якості вихідної інформації не тільки рішення, але і необхідні пояснення.

Розрізняють два види пояснень:

• пояснення, що видаються за запитами. Користувач в будь-який момент може вимагати від експертної системи пояснення своїх дій;

• пояснення отриманого рішення проблеми. Після отримання рішення користувач може зажадати пояснень того, як воно було отримано. Система повинна пояснити кожен крок своїх міркувань, що ведуть до розв'язання задачі. Хоча технологія роботи з експертною системою не є простою, призначений для користувача інтерфейс цих систем є дружнім і звичайно не викликає труднощів при веденні діалогу.

База знань. Вона містить факти, що описують проблемну галузь, а також логічний взаємозв'язок цих фактів. Центральне місце в базі знань належить правилам. Правило визначає, що слід робити в даній конкретній ситуації, і складається з двох частин: умови, яке може виконуватися чи ні, і дії, які слід здійснити, якщо умова виконується.

Всі використовувані в експертній системі правила утворюють систему правил, яка навіть для відносно простої системи може містити кілька тисяч правил.

Інтерпретатор. Це частина експертної системи, що виконує у певному порядку обробку знань (мислення), що знаходяться в базі знань. Технологія роботи інтерпретатора зводиться до послідовного розгляду сукупності правил (правило за правилом). Якщо умова, що міститься в правилі, дотримується, виконується певна дія, і користувачу надається варіант вирішення його проблеми.

Крім того, у багатьох експертних системах вводяться додаткові блоки: база даних, блок розрахунку, блок введення і коректування даних. Блок розрахунку необхідний у ситуаціях, пов'язаних з прийняттям управлінських рішень. При цьому важливу роль відіграє база даних, де містяться планові, фізичні, розрахункові, звітні та інші сталі або оперативні показники. Блок введення і коректування даних використовується для оперативного і своєчасного відображення поточних змін в базі даних.

Модуль створення системи. Він служить для створення набору (ієрархії) правил. Існують два підходи, які можуть бути покладені в основу модуля створення системи: використання алгоритмічних мов програмування і використання оболонок експертних систем.

Для уявлення бази знань спеціально розроблені мови Лісп і Пролог, хоча можна використовувати і будь-який відомий алгоритмічний мову.

Оболонка експертних систем являє собою готову програмну середу, яка може бути пристосована до вирішення певної проблеми шляхом створення відповідної бази знань. У більшості випадків використання оболонок дозволяє створювати експертні системи швидше і легше в порівнянні з програмуванням.

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Проблеми та перспективи використання інформаційних технологій

2. Навчальна мета. Студент повинен:

знати: напрямки розвитку і використання інформаційних тьехнологій

уміти:

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Старіння інформаційної технології
2) Методології використання інформаційних тьехнологій
3) Впровадження інформаційних технологій
4) Розвиток способів обробки даних

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Чому "старіють" інформаційні технології?
2) Де і коли підприємство використовує інформаційні технології

5.Контроль засвоєння теми

Письмова робота.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X
3. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

3 ПРОБЛЕМИ ТА ПЕРСПЕКТИВИ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

3.1 Старіння інформаційної технології

Для інформаційних технологій є цілком природним те, що вони застарівають і заміняються новими.

Так, наприклад, на зміну технології пакетної обробки програм на великий ЕОМ в обчислювальному центрі прийшла технологія роботи на персональному комп'ютері на робочому місці користувача. Телеграф передав всі свої функції телефону. Телефон поступово витісняється службою експресдоставки. Телекс передав більшість своїх функцій факсу й електронній пошті.

При впровадженні нової інформаційної технології в організації необхідно оцінити ризик відставання від конкурентів у результаті її неминучого старіння, так як інформаційні продукти, як ніякі інші види матеріальних товарів, мають надзвичайно високу швидкість змінюваності новими видами або версіями. Періоди змінюваності коливаються від декількох місяців до одного року. Якщо в процесі впровадження нової інформаційної технології цьому факторі не приділяють належної уваги, можливо, що до моменту завершення переходу фірми на нову інформаційну технологію вона вже застаріє і прийдеться вживати заходів щодо її модернізації. Такі невдачі з впровадженням інформаційних технологій звичайно пов'язані з недосконалістю технічних засобів, тоді як основною причиною невдач є відсутність або слабка опрацьованість методології використання інформаційної технології.

3.2 Методологія використання інформаційної технології:

Централізована обробка інформації на ЕОМ обчислювальних центрів була першою історично сформованою технологією. Створювалися великі обчислювальні центри колективного користування, оснащені великими ЕОМ (в нашій країні - ЕОМ ЄС). Застосування таких ЕОМ дозволяло обробляти великі масиви вхідної інформації н отримати на цій основі різні види інформаційної продукції, яка потім передавалася користувачам. Такий технологічний процес був обумовлений недостатнім оснащенням обчислювальною технікою підприємств і організацій у 60 - 70-ті рр.

Переваги методології централізованої технології:

• можливість звернення користувача до великих масивів інформації у вигляді баз даних і до інформаційної продукції широкої номенклатури;

• відносна легкість упровадження методологічних рішень по розвитку й удосконаленню інформаційної технології завдяки централізованому їхньому прийняттю

Недоліки такої методології очевидні

• обмежена відповідальність нижчого персоналу, який не сприяє оперативному отриманню інформації користувачем, тим самим перешкоджаючи правильності виробітку управлінських рішень;

• обмеження можливостей користувача в процесі одержання н використання інформації.

Децентралізована обробка інформації пов'язана з появою в 8О-х р.р .. персональних комп'ютерів і розвитком засобів телекомунікацій. Вона має велике значення потіснила попередню технологію, оскільки дає користувачу широкі можливості в роботі з інформацією і не обмежує його ініціатив.

Перевагами такої методології є:

• гнучкість структури, що забезпечує простір ініціативам користувача;

• посилення відповідальності нижчої ланки співробітників;

• зменшення потреби в користуванні центральним комп'ютером і відповідно контролі з боку обчислювального центру;

• більш повна реалізація творчого потенціалу користувача завдяки використанню засобів комп'ютерного зв'язку.

Однак ця методологія має і свої недоліки:

• складність стандартизації через великої кількості унікальних розробок;

• психологічне неприйняття користувачами що рекомендуються обчислювальним центром стандартів в готових програмних продуктів;

• нерівномірність розвитку рівня інформаційної технології на локальних місцях, що в першу чергу визначається рівнем кваліфікації конкретного працівника.

Описані переваги і недоліки централізованої і децентралізованої інформаційної технології привели до необхідності дотримуватися лінії розумного застосування і того, і іншого підходу.

Такий підхід назвемо р а ц і о н а л ь н о ю методологією і покажемо, як у цьому випадку будуть розподілятися обов'язки:

• обчислювальний центр повинен відповідати за створення загальної стратегії використання інформаційної технології, допомагати користувачам як у роботі, так і в навчанні. встановлювати стандарт і визначати політику застосування програмних і технічних засобів;

• персонал, який використовує інформаційну технологію, повинен дотримуватися вказівок обчислювального центру, здійснювати розробку своїх локальних систем і технологій відповідно до загального плану організації.

Раціональна методологія використання інформаційної технології дозволить досягти більшої гнучкості, підтримувати загальні стандарти, здійснити сумісність інформаційних локальних продуктів, знизити дублювання діяльності та ін.

3.3 Вибір варіантів впровадження інформаційної технології в фірмі

При впровадженні інформаційної технології у фірму необхідно вибрати одну з двох основних концепцій, що відбивають сформовані точки зору на існуючу структуру організації і роль в ній комп'ютерної обробки інформації.

Перша концепція орієнтується на існуючу структуру фірми. Інформаційна технологія пристосовується до організаційної структури, і відбувається лише модернізація методів роботи. Комунікації розвинені слабко, раціоналізуються тільки робочі місця. Відбувається розподіл функцій між технічними робітниками і фахівцями. Ступінь ризику від впровадження нової інформаційної технології мінімальна., Так як витрати незначні і організаційна структура фірми не змінюється.

Основний недолік такої стратегії - необхідність безперервних змін форми подання інформації, пристосованої до конкретних технологічних методів і технічних засобів. Будь-яке оперативне рішення "в'язне" на різних етапах інформаційної технології.

До переваг стратегії можна віднести мінімальні ступінь ризику і витрати.

Друга концепція орієнтується на майбутню структуру фірми. Існуюча структура буде модернізуватися.

Дана стратегія передбачає максимальний розвиток комунікацій і розробку нових організаційних взаємозв'язків. Продуктивність організаційної структури фірми зростає, так як раціонально розподіляються архіви даних, знижується обсяг циркулюючої по системним каналам інформації та досягається збалансованість між вирішуваних завдань.

До основних її недоліків слід віднести:

• істотні витрати на першому етапі, пов'язаному з розробкою загальної концепції та обстеженням всіх підрозділів фірми;

• наявність психологічної напруженості, викликаної передбачуваними змінами структури фірми і, як наслідок, змінами штатного розкладу і посадових обов'язків

Перевагами даної стратегії є:

•раціоналізація організаційної структури фірми;

•максимальна зайнятість усіх працівників;

•високий професійний рівень;

•інтеграція професійних функцій за рахунок використання комп'ютерних мереж.

Нова інформаційна технологія в фірмі повинна бути такою, щоб рівні інформації і підсистеми, її обробні, пов'язувалися між собою єдиним масивом інформації. При цьому пред'являються дві вимоги. По-перше, структура системи переробки інформації повинна відповідати розподілу повноважень у фірмі. По-друге, інформація усередині системи повинна функціонувати так, щоб досить повно відображати рівні управління.

Розвиток способів обробки даних

Складність сучасної обробки даних є результатом розвитку протягом кількох десятиріч способів обробки даних і управління інформацією.

Обробка даних розвивалась від примітивних методів 50-тих років до складних інтегрованих систем сьогодення.

Перші системи обробки даних виконували лише канцелярську роботу, зменшуючи кількість документів. Оскільки ці системи виконували звичайні функції роботи з документами, вони були названі *системами обробки даних.* Розробники цих систем втілювали в програми ті операції, які раніше виконувались вручну. Дані зберігались на магнітних носіях в файлах. Файл вміщував ту інформацію, яка раніше могла лежати в одній папці. В той час в якості магнітних носіїв виступали магнітні стрічки, які допускали лише послідовний доступ. Обробка таких файлів вимагала багато часу..

Використання прямого доступу в 60-ті роки до файлу скоротило час обробки даних, але використання таких файлів вирішило проблему лише частково.

Системи обробки даних виконували конкретну роботу, тобто така система давала конкретний кінцевий результат: нарахована заробітна плата, дані про виконану роботу, витрати на закупки, тощо.

А якщо доповнити систему таким чином, щоб вона могла відповідати на деякі запитання тоді, коли це необхідно? Наприклад, який поточний баланс компанії? Таке запитання є запитання до інформаційної системи. Тобто, в кінці 60, на початку 70 років сталася зміна: перехід від обробки даних до обробки інформації. Це привело до створення інформаційно-управляючих систем. Такі системи використовують дані, що вже є в комп'ютері, відповідаючи на широке коло питань.

Необхідно відрізняти *дані* і *інформацію*.

Дані – це є деякі факти. Ці факти розміщуються в файлах.

Інформація – це оброблені дані. Інформація отримується в результаті обробки великої кількості фактів. Таким чином, інформація – це ресурс, який володіє визначеною цінністю і отже потребує упорядкування і управління.

Технологія організації даних, яка дозволяє упорядковувати дані і управляти ними є ***технологією баз даних.***

База даних – це сукупність даних, яка має внутрішню структуру.

Інформаційні системи, що використовують БД, дозволили подолати обмеження файлових систем.

Структуру і взаємозв'язки даних в базі даних визначає модель даних, або концептуальні методи структурування даних.

Основою є три базові моделі: Ієрархічна, Мережева, Реляційна.

Перша система, що використовувала базу даних, з'явилась в середині 60-х років і була основана на ієрархічній моделі.

В кінці 60, на початку 70 з'явились мережеві СУБД.

І в ієрархічній і в мереженій системах БД для зв'язку файлів використовувались фізичні вказники. Ці вказники дозволяли вилучати дані, зв'язані відповідними відношеннями. Але недоліком таких систем було те, що ці відношення повинні бути визначені до запуску системи. Вилучити дані на основі інших відношень було неможливо.

В 1970 році Е.Ф.Кодд опублікував статтю в якій висунув ідею, що дані потрібно зв'язувати в відповідності з їх внутрішніми логічними взаємовідношеннями, а не фізичними вказниками. Таким чином користувачі зможуть комбінувати дані з різних джерел, якщо логічна інформація, що необхідна для такого комбінування, присутня в висхідних даних. Така модель даних дістала назву *реляційної* моделі. Для роботи з даними використовується реляційна алгебра і реляційне числення. Це забезпечує роботу з даними на основі логічних характеристик. В реляційних системах однією командою може оброблятись цілий файл, а в ієрархічних і мережених – тільки один запис.

Логічний підхід до даних зробив можливим створення мов запитів, що є більш доступними для користувачів, що не є спеціалістами.

В другій половині 70 років з'явились реляційні системи, які підтримували мову структурованих запитів (SQL), мову запитів (Quel), запити по зразку (QBE).

Сьогодні реляційні системи розглядаються як стандарт систем роботи з даними.

Реляційні бази даних продовжують вдосконалюватись, їх внутрішня природа змінюється. Це є використання об'єктно-орієнтованих баз даних (використання концептуальної моделі даних) і використання технології клієнт-серевер.

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Фізична організація баз даних

2. Навчальна мета. Студент повинен:

знати: поняття фізичної організації даних і її реалізацію

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Що означає поняття “фізична організація”
2) Характеристика файлових структур
3) Використання СКБД
4) Види адресації пам’яті
5) Структури даних, що зберігаються

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Які механізми обробки даних можуть використовуватись в сучасній технології обробки даних?
2) Розміщення даних в просторі пам’яті
3) Поняття запису і його використання

5.Контроль засвоєння теми

Різнорівневі питання

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. Дейт К. Введення в системи баз даних: Пер. з англ. - 6-е вид. - Київ: Діалектика, 1998. - 784 с.
3. Ревунков Г.І., Самохвалов Е.Н., Чистов В.В. Бази і банки даних і знань: Учеб. для вузів / За ред. Четверикова В.Н. - М .: Вища. шк., 1992. - 367 с.
4. Мейер Д. Теорія реляційних баз даних: Пер. з англ. - М .: Світ, 1987. - 608 с.
5. Тіорі Т., Фрай Дж. Проектування структур баз даних: У 2-х кн. / Пер.з англ. - М .: Світ, 1985.

Фізична організація бази даних

Фізичні моделі баз даних визначають способи розміщення даних в середовищі зберігання і способи доступу до цих даних, які підтримуються на фізичному рівні. Історично першими системами зберігання і доступу були файлові структури і системи управління файлами (СУФ), які фактично були частиною операційних систем. *СУБД* створювала над цими файловими моделями свою надбудову, яка дозволяла організувати всю сукупність файлів таким чином, щоб вона працювала як єдине ціле і отримувала *централізоване управління* від *СУБД.* Однак безпосередній *доступ* здійснювався на рівні файлових команд, які *СУБД* використовувала при маніпулюванні всіма файлами, складовими збережені дані однієї або декількох баз даних.

Однак *механізми* буферизації і управління файловими структурами не пристосовані для вирішення завдань власне *СУБД,* ці *механізми* розроблялися просто для традиційної обробки файлів, і з ростом обсягів даних, що зберігаються вони стали неефективними для використання *СУБД.* Тоді поступово відбувся перехід від базових файлових структур до безпосереднього управління розміщенням даних на зовнішніх носіях самої *СУБД.* І *простір* зовнішньої пам'яті вже виходив з-під володіння СУФ і управлявся безпосередньо *СУБД.* При цьому *механізми,* що застосовуються в файлових системах, перейшли в чому і в нові системи організації даних у зовнішній пам'яті, звані частіше сторінковими системами зберігання інформації.

Файлові структури, використовувані для зберігання інформації в базах даних

В кожній СУБД по-різному організовані зберігання і доступ до даних, однак існують деякі файлові структури, які мають загальноприйняті способи організації та широко застосовуються практично у всіх СУБД.

У системах баз даних файли та файлові структури, які використовуються для зберігання інформації у зовнішній пам'яті, можна класифікувати наступним чином.

Рисунок 1-Класифікація файлів, використовуваних в системах баз даних

З точки зору користувача, файлом називається пойменована лінійна послідовність записів, розташованих на зовнішніх носіях.

Так як файл - це лінійна послідовність записів, то завжди в файлі можна визначити поточний запис, що передує і наступний. Завжди існує поняття першого та останнього запису файлу.

Механізми середовища зберігання і архітектура СУБД

Механізми середовища зберігання БД служать для управління двома групами ресурсів - ресурсами **збережених даних** і ресурсами **простору пам'яті.** У завдання цього механізму входить відображення структури збережених даних в простір пам'яті, що дозволяє ефективно використовувати пам'ять і визначити місце розміщення даних при запам'ятовуванні і при пошуку даних. У більшості випадків в якості одиниці зберігання приймається збережена запис.

Механізми середовища зберігання виконують такі операції:

1. При запам'ятовуванні нового об'єкта:
   - визначення місця розміщення нового об'єкта "фізичної" БД в просторі пам'яті;
   - виділення необхідного ресурсу пам'яті;
   - запам'ятовування цього об'єкта;

- формування зв'язків з іншими об'єктами.

2. При пошуку об'єкта:

- пошук місця розміщення об'єкта в просторі пам'яті за заданими атрибутам або "адресою";
- вибірка об'єктів для обробки.

3. При видаленні об'єкта:

- видалення об'єкта зі звільненням пам'яті (фізичне видалення) або без звільнення (логічне видалення);
- руйнування зв'язків з іншими об'єктами.

Примітка: в СУБД формування зв'язків здійснюється на логічному рівні (тобто за значеннями атрибутів), а в ієрархічних і мережевих СУБД - на фізичному рівні (за адресами записів).

Всі операції виконуються за запитами механізмів концептуального рівня СУБД. На цьому рівні ніяких операцій безпосереднього поновлення призначених для користувача даних або перетворень уявлення збережених даних не відбувається, це завдання більш високих архітектурних рівнів. Управління пам'яттю виконується операційною системою за запитами СУБД або безпосередньо самої СУБД.

Простір пам'яті і розміщення даних, що зберігаються

Ресурсів простору пам'яті відповідають об'єкти зовнішньої пам'яті ЕОМ, керовані засобами операційної системи або СУБД.

Для забезпечення природної структуризації даних, що зберігаються, більш ефективного управління ресурсами і / або для технологічного зручності весь простір пам'яті БД зазвичай розділяється на частини (області, розділи і ін.). (У багатьох системах область відповідає файлу.) *Області пам'яті* використовуються для розміщення збережених записів одного або декількох типів і розбиваються на пронумеровані *сторінки* фіксованого розміру. У більшості систем обробку даних на рівні сторінок веде операційна система (ОС), а обробку записів всередині сторінки забезпечує тільки СУБД.

Сторінки представляються в середовищі ОС блоками зовнішньої пам'яті, кластерами або секторами, доступ до яких здійснюється за одне звернення. У системах, які дозволяють управляти розміром сторінки, доводиться шукати компроміс між продуктивністю системи і необхідним обсягом оперативної пам'яті.

Сторінка має **заголовок,** що містить службову інформацію, слідом за яким розташовуються власне дані. На сторінці розміщується, як правило, кілька записів, і є вільна ділянка для розміщення нових записів. Якщо запис не поміщається на одній сторінці, вона розбивається на фрагменти, які зберігаються на різних сторінках і мають посилання один на одного.

Структура даних, що зберігаються

Одиницею зберігання даних в БД є **збережений запис.** Вона може представляти як повний запис концептуального рівня, так і деяку її частину. Якщо запис розбивається на частини, то її фрагменти представляються екземплярами збережених записів будь-яких типів. Всі частини запису зв'язуються покажчиками (посиланнями) або розміщуються за спеціальним законом так, щоб механізми междууровневого відображення могли впізнати всі компоненти і здійснити збірку повного запису концептуальної БД за запитом механізмів концептуального рівня.

Збережені записи одного типу складаються з фіксованої сукупності полів і можуть мати формат фіксованою або змінною довжини.

Записи змінної довжини виникають, якщо допускається використання повторюваних груп полів (агрегатів) зі змінним числом повторів або полів змінної довжини. Робота з збереженими записами змінної довжини істотно ускладнює управління простором пам'яті, але може бути продиктована бажанням зменшити обсяг необхідної пам'яті або характером моделі даних концептуального рівня. В останньому випадку для підвищення продуктивності системи запису можуть розбиватися на фрагменти фіксованої довжини, можливо, різних типів.

Збережений запис складається з двох частин:

1. *Службова частина.* Використовується для ідентифікації запису, завдання її типу, зберігання ознаки логічного видалення, для кодування значень елементів запису, для встановлення структурних асоціацій між записами. Ніякі призначені для користувача програми не мають доступу до службової частини збереженої записи.

2.*Інформаційна частина.* Містить значення елементів даних.

Елементи збереженої записи можуть мати фіксовану або змінну довжину. При цьому зазвичай елементи фіксованої довжини зберігаються на початку запису і розміщуються в пам'яті з заздалегідь визначених позицій, а за ними розміщуються елементи змінної довжини. Зберігання елементів змінної довжини здійснюється одним із двох способів: розміщення через роздільник або зберігання розміру елемента даних. Наявність полів змінної довжини дозволяє не зберігати незначні символи і тим істотно знижує витрати пам'яті на зберігання даних; але при цьому збільшується час на витяг елементів запису.

Кожного запису БД система привласнює внутрішній ідентифікатор, званий (за стандартом CODASYL) **ключем бази даних** (КБД). Значення КБД формується системою при розміщенні записи і містить інформацію, що дозволяє однозначно визначити місце розміщення записи (її адреса). Як КБД може виступати, наприклад, послідовний номер запису у файлі або сукупність адреси сторінки і зміщення від початку сторінки.

Конкретні складові КБД залежать від операційної системи і від СУБД, точніше, від виду використовуваної адресації і від структуризації пам'яті, прийнятої в даній СУБД.

Види адресації збережених записів

Розглянемо два види адресації: пряму і непряму.

Пряма адресація передбачає зазначення безпосереднього місця розташування записи в просторі пам'яті. Найпростіший варіант адресації використовується в тому випадку, якщо в пам'яті зберігається один вид записи фіксованої довжини. Тоді як адресу записи може використовуватися її порядковий номер. (Пряма адресація використовується, наприклад, в системах ADABAS і dBaseIII PLUS).

Пряма адресація не дозволяє переміщати записи в пам'яті без зміни ключа бази даних. Такі зміни призвели б до необхідності корекції різних покажчиків на записи в середовищі зберігання, що було б надзвичайно трудомісткою процедурою. У зв'язку з відсутністю можливості переміщення при цьому виникає явище *фрагментації,* тобто поява розрізнених незаповнених ділянок пам'яті.

Зазначені недоліки можна подолати, використовуючи **непряму адресацію.** Існує безліч способів непрямої адресації. Один з них полягає в тому, що частина адресного простору сторінки виділяється під індекс сторінки. Число статей (слотів) в ньому однаково для всіх сторінок. Ключем БД як і раніше служить номер цього запису в області. За допомогою простих арифметичних дій можна отримати за номером запису номер потрібної сторінки і номер необхідного слота в індексі цієї сторінки. Знайдений слот вкаже місце розташування записи на цій сторінці, де N, n - це відповідно номер сторінки пам'яті і номер слота на цій сторінці, в якому зберігається адреса записи (зміщення від початку сторінки).

Організація зв'язків між збереженими записами

Для мережевої та ієрархічної моделей даних ці зв'язки підтримуються на фізичному рівні. Способи подання зв'язків визначаються схемою зберігання і найчастіше засновані на використанні покажчиків або на розміщенні даних в суміжних областях пам'яті.

У мережевих і ієрархічних БД асоціацію між даними підтримуються груповими відносинами. Найбільш поширений спосіб підтримки групових відносин - створення ланцюгового списку. Для цього запис-власник у службовій частині містить адресну посилання (ключ бази даних) на перший підлеглий запис, а кожний підпорядкований запис містить посилання на наступнй.

Посилання можуть бути односпрямованим (тільки на наступний запис) або двонаправленими (на наступну і на попередню записи.

При створенні нового запису в багатьох випадках істотним представляється розміщення цього запису в пам'яті, що має великий вплив на час вибірки. Найпростіша стратегія розміщення даних полягає в тому, що новий запис розміщується на першому вільному ділянці (якщо ведеться облік вільного простору) або слідом за останньою з раніше розміщених записів. Серед більш складних методів можна відзначити хешування і кластеризації даних.

Способи доступу до записів

Розглянемо основні способи доступу до даних.

- **Послідовна обробка області БД** . Областю БД може бути файл або інше безліч сторінок. Послідовна обробка передбачає, що система послідовно переглядає сторінки, пропускає порожні ділянки та видає записи в фізичної послідовності їх зберігання.

- **Доступ по ключу бази даних** (КБД). КБД визначає місце розташування записи в пам'яті ЕОМ. Знаючи його, система може отримати потрібну запис за одне звернення до пам'яті.
- **Доступ по структурі.** Цей різновид доступу застосовується для групових відносин і дозволяє перейти до попереднього або наступного екземпляру групового відносини, до примірника-власнику групового відносини або до списку підлеглих примірників.
- **Доступ по первинному ключу.** Первинний ключ ідентифікує записи всередині типу. Якщо система забезпечує доступ по первинному ключу, то він (ключ) використовується також при запам'ятовуванні записи і, більш того, його значення в цьому випадку зазвичай використовується при розміщенні записи в пам'яті. Найбільш поширені механізми доступу по первинному ключу - індексування і хешування.

Індексування даних

При випадковому доступі до окремих записів найбільш ефективним є доступ по ключу. Для прискорення доступу до записів по ключовому атрибуту (або групі атрибутів) створюється спеціальна структура - індекс , який визначає відповідність значення атрибута (групи атрибутів) і розташування записи.

Значення індексованого атрибута упорядковуються (найчастіше, за зростанням). Індекс зазвичай зберігається в окремому файлі або окремої області пам'яті. Порожні значення атрибутів (null) не індексують.

Індекси підтримуються динамічно, тобто після поновлення БД - додаванні або видаленні записів, а також при модифікації полів записи, що входять в ключ, - індекс приводиться у відповідність з оновленою версією БД.

Організація паралельного доступу до даних

Паралельний доступ до даних має на увазі одночасне виконання двох і більше запитів до одних і тих же об'єктів даних (таблиць, блокам і т.п.). Для організації одночасного доступу не обов'язкова наявність багатопроцесорноїсистеми. На однопроцессорной ЕОМ запити виконуються не одночасно, а паралельно. Зазвичай для кожного запиту виділяється певна кількість процесорного часу (квант часу), після закінчення якого виконання запиту призупиняється, він ставиться в чергу запитів, а на виконання запускається наступний (по черзі) запит. Таким чином, процесорний час ділиться між запитами, і створюється ілюзія, що запити виконуються одночасно.

При паралельному доступі до даних проблеми виникають в тому випадку, якщо доступ має на увазі внесення змін. Для того щоб виключити порушення логічної цілісності даних при многопользовательском доступі, використовується механізм транзакцій.

Механізм транзакцій

Транзакція - це послідовність операторів обробки даних, яка розглядається як логічно неподільна одиниця роботи з базою даних.

Транзакція має такі властивості:

1. Логічна неподільність (**атомарность**) означає, що виконуються або всі операції, що входять в транзакцію, або жодної. (Логічна неподільність не має на увазі фізичної неподільності).

Система гарантує неможливість фіксації частини змін, вироблених транзакцією. До тих пір, поки транзакція не зафіксована, її можна "відкотити", тобто скасувати всі зроблені операторами з транзакції зміни в БД. Успішне виконання транзакції означає, що всі оператори транзакції проаналізовані, інтерпретовані як правильні і безпомилково виконані.

2. **Узгодженість** : транзакція починається на узгодженому безлічі даних і після її завершення безліч даних також погоджено.

3. **Ізольованість** , тобто відсутність впливу транзакцій один на одного. (Насправді цей вплив існує і регламентується стандартом: см. Розділ 7.2. "Взаємовплив транзакцій").

4. **Тривалість** : результати зафіксованої транзакції не можуть бути втрачені. Повернення БД в попередній стан може бути досягнутий тільки шляхом запуску компенсує транзакції.

МЕТОДИЧНІ ВКАЗІВКИ

із самостійної роботи студентів

1. Тема: Семантичні моделі

2. Навчальна мета. Студент повинен:

знати: визначення і призначення семантичної моделі даних

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Що означає семантика?
2) Мета моделювання даних
3) Використання семантичної моделі
4) Методології семантичного моделювання

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Основна мета використання семантичних моделей
2) Варіанти реалізації семантичних моделей

5.Контроль засвоєння теми

Різнорівневі питання

6. Рекомендована література

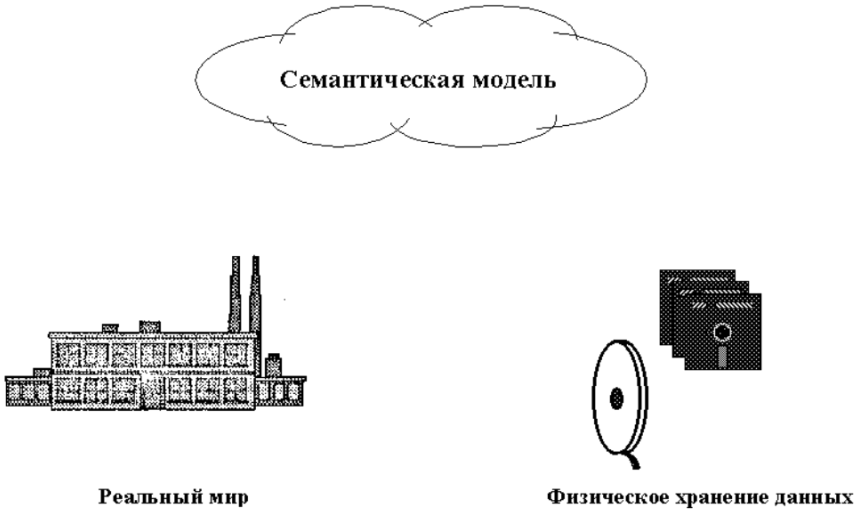
1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)

2. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

Семантичні моделі

Цілі моделювання даних

Логічна структура даних СУБД, ієрархічна, мережева або реляційна, не може повністю задовольняти вимогам до концептуального визначення даних, оскільки вона має обмежені рамки і обумовлюється стратегією реалізації СУБД. Необхідність визначення даних з концептуальної точки зору привела до розробки методології моделювання даних, заснованої на семантиці, тобто до трактування даних в контексті їх взаємозв'язків з іншими даними. Як показано на рис.2-4, реальний світ в термінах ресурсів, ідей, подій і т.п. можна символічно представити в рамках фізичного зберігання даних. Семантична модель даних є абстрактною схемою, яка б показала, як зберігаються символи співвідносяться з реальним світом. Тобто така модель повинна бути вірним відображенням реального світу.



Семантичні моделі даних

Семантична модель даних може застосовуватися в різних цілях. Зазначимо найважливіші з них.

1. Планування ресурсів даних

Попередня модель даних допомагає при виробленні широкого погляду на дані, необхідні для діяльності підприємства. Потім ця модель може бути досліджена для побудови спільно використовуваних ресурсів даних.

2. Побудова спільно використовуваних баз даних

Повністю розроблена модель може застосовуватися для представлення даних незалежно від їх конкретного використання. Це уявлення може бути перевірено користувачами і потім перетворено в фізичний проект бази даних для будь-якої з різних технологій СУБД. Крім того, що розроблені бази даних будуть суперечити одна одній і спільно використовуваними, моделювання даних істотно скоротить витрати на розробку.

3. Оцінка програмного забезпечення, що купується

Модель даних, що відображає дійсну інфраструктуру організації, дозволяє оцінити, наскільки купується програмне забезпечення відповідає моделі даних компанії і чи не суперечить інфраструктура, що накладається програмним забезпеченням, способу ведення справ компанії.

4. Об'єднання існуючих баз даних

Визначивши зміст існуючих баз даних через семантичні моделі даних можна отримати інтегроване визначення даних. Концептуальна схема може використовуватися для управління обробкою запитів в середовищі розподіленої бази даних. Проект "Підтримка інформаційних інтегрованих систем" ВВС США (IISS) є експериментальною розробкою, в якій демонструється застосування технологій такого типу до неоднорідному середовищі СУБД.

IDEF1X-підхід

IDEF1X - це методологія семантичного моделювання даних. Вона розроблена з урахуванням таких вимог:

1. Підтримує розробку концептуальних схем

Синтаксис IDEF1X підтримує семантичні конструкції, необхідні для розробки концептуальної схеми. Остаточна версія IDEFlX-моделі володіє бажаними характеристиками -непротіворечівостью, розширюваністю і адаптованість.

2. Забезпечує ясну мову

IDEF1X має просту, ясну, несуперечливу структуру і чіткі семантичні поняття. Синтаксис і семантика IDEF1X порівняно легкі для розуміння, хоча і є досить потужним засобом.

3. Проста для вивчення

Семантичне моделювання даних - нове поняття для багатьох користувачів IDEF1X. Проблема навченості цій мові є важливим фактором. Мова розрахований на розуміння і використання як професійними бізнесменами і системними аналітиками, так і адміністраторами даних і розробниками баз даних.

4. Надійно перевірена на практиці

IDEF1X базується на багаторічному досвіді попередніх методологій і ретельно перевірена як у проектах ВВС, так і в промисловості.

5. Можливість автоматизації

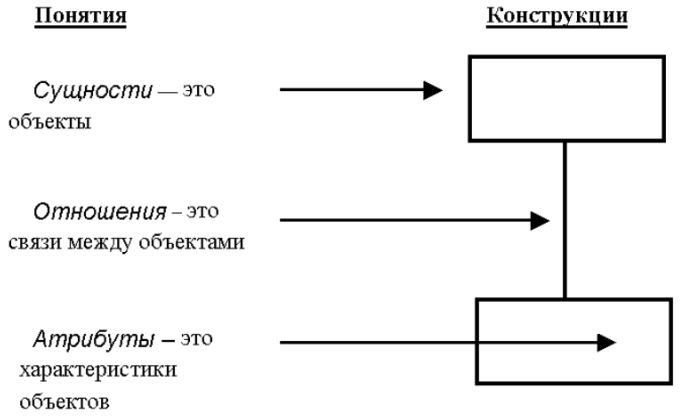
IDEFlX-діаграми можуть створюватися великим числом графічних програмних пакетів.

IDEF1X використовує підхід сутностей-відносин до семантичного моделювання даних. Вихідна розробка IDEF1 полягала в розширенні понять сутності-відносини за методом П.Ченна, об'єднаних з поняттями реляційної теорії Т. Кодда. Крім того, для поліпшення графічного представлення і процедур моделювання IDEFlX-методологія семантично збагачена введенням відносин категоризації (званих також відносинами узагальнення). Мова IDEF1X включає комерційні розробки ***D.Appleton Company*** і ***The Database Design Group.***

Основними конструкціями IDEFlX-моделі є:

1. Предмети, до яких відносяться дані, тобто люди, місця, ідеї, події і т.д. Вони зображуються блоками.
2. Відносини між цими предметами, зображувані з'єднують блоки лініями.
3. Характеристики цих предметів, що зображуються іменами атрибутів всередині блоків.

Основні конструкції показані на рис. . Вони використовуються і далі в цьому керівництві.



МЕТОДИЧНІ ВКАЗІВКИ

із самостійної роботи студентів

1. Тема: Функціональна модель

2. Навчальна мета. Студент повинен:

знати: визначення і призначення функціональної моделі даних

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Діаграми потоків даних
2) Опис операцій
3) Обмеження

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Основна мета використання функціональних моделей
2) Реалізація моделі потоків даних
3) Основні обмеження

5.Контроль засвоєння теми

Різнорівневі питання

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)

2. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

3. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X

Функціональна модель

1) Діаграми потоків даних
2) Опис операцій
3) Обмеження

Функціональна модель описує обчислення в системі. Вона показує, яким чином вихідні дані обчислюються за вхідними даними, не розглядаючи порядок і спосіб реалізації обчислень. Функціональна модель складається з набору діаграм потоку даних, які показують потоки значень від зовнішніх входів через операції і внутрішні сховища даних до зовнішніх виходів. Функціональна модель описує сенс операцій об'єктної моделі і дій динамічної моделі, а також обмеження на об'єктну модель.

Діаграми потоків даних

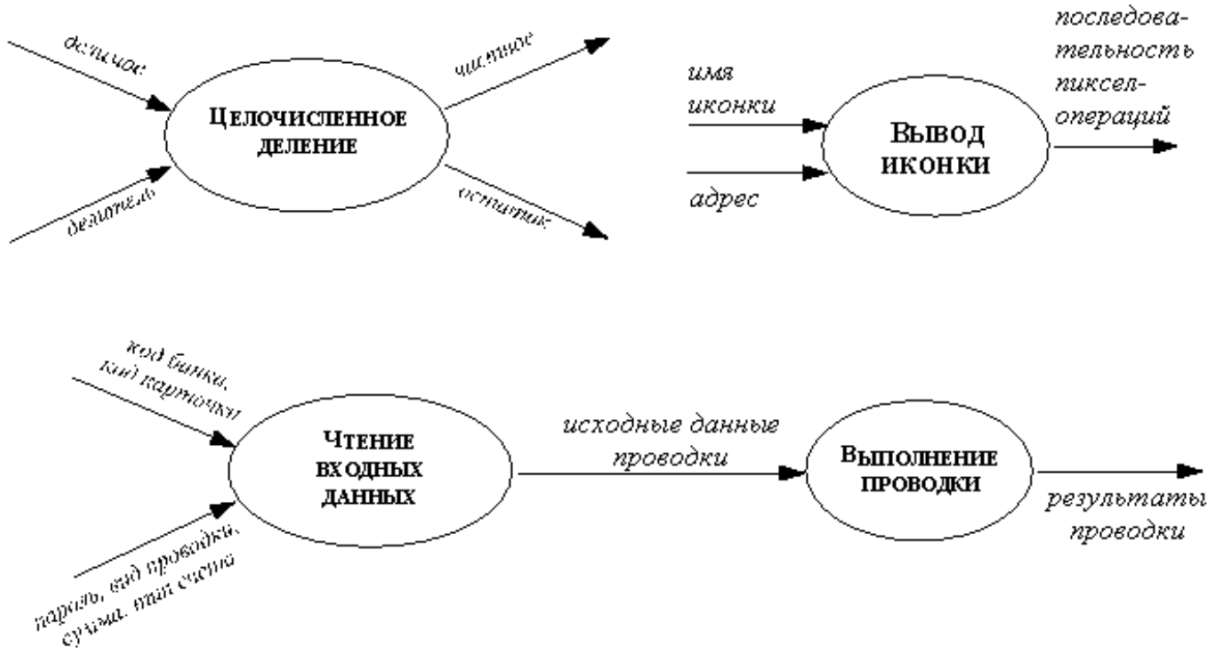
Функціональна модель являє собою набір діаграм потоків даних (далі - ДПД), які описують зміст операцій і обмежень. ДПД відображає функціональні залежності значень, що обчислюються в системі, включаючи вхідні значення, вихідні значення і внутрішні сховища даних. ДПД - це граф, на якому показано рух значень даних від їх джерел через перетворюють їх процеси до їх споживачам в інших об'єктах.

ДПД містить процеси, які перетворюють дані, потоки даних, які переносять дані, активні об'єкти, які виробляють і споживають дані, і сховища даних, які пасивно зберігають дані.

процеси

Процес перетворює значення даних. Процеси самого нижнього рівня являють собою функції без побічних ефектів (прикладами таких функцій є обчислення суми двох чисел, обчислення комісійного збору за виконання проводки за допомогою банківської картки і т.п.). Весь граф потоку даних теж являє собою процес (високого рівня). Процес може мати побічні ефекти, якщо він містить нефункціональні компоненти, такі як сховища даних або зовнішні об'єкти.

На ДПД процес зображується у вигляді еліпса, всередині якого міститься ім'я процесу; кожен процес має фіксоване число вхідних і вихідних даних, зображуваних стрілками

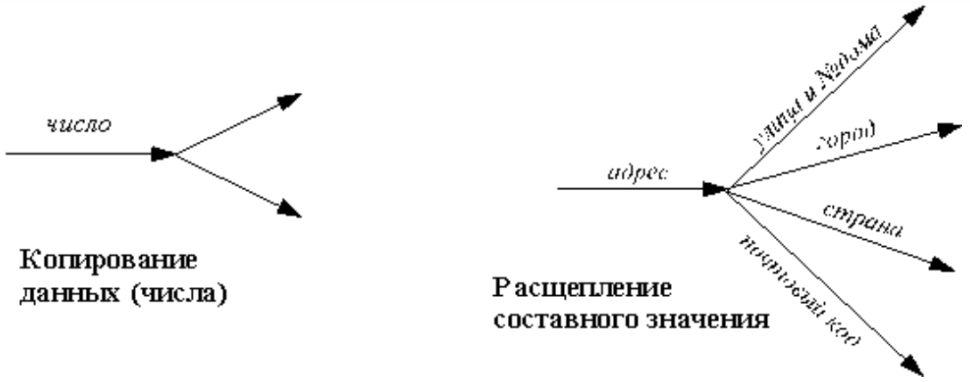


Приклади процесів

Процеси реалізуються у вигляді методів (або їх частин) і відповідають операціям конкретних класів.

Потоки даних

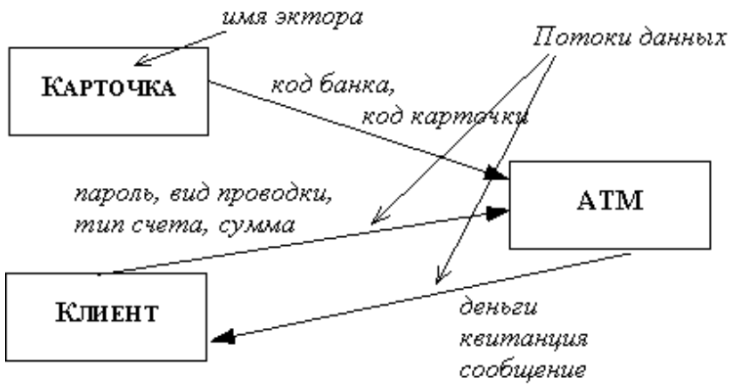
Потік даних з'єднує вихід об'єкта (або процесу) зі входом іншого об'єкта (або процесу). Він являє проміжні дані обчислень. Потік даних зображується у вигляді стрілки між виробником і споживачем даних з позначкою іменами відповідних даних; приклади стрілок, що зображують потоки даних, представлені на малюнку 2.61. На першому прикладі зображено копіювання даних при передачі одних і тих же значень об'єктах, на другому - розщеплення структури на її поля при передачі різних полів структури різних об'єктів.



Потоки даних

Активні об'єкти

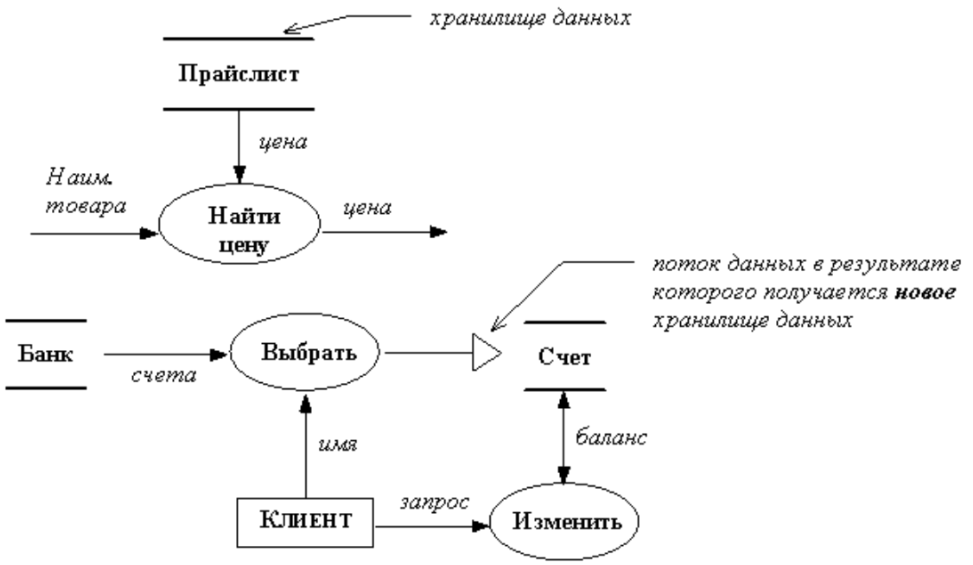
Активним називається об'єкт, який забезпечує рух даних, поставляючи або споживаючи їх. Активні об'єкти зазвичай бувають приєднані до входів і виходів ДПД. Приклади активних об'єктів показані на рис. На ДПД активні об'єкти позначаються прямокутниками.



Активні об'єкти (Ектор)

Сховища даних

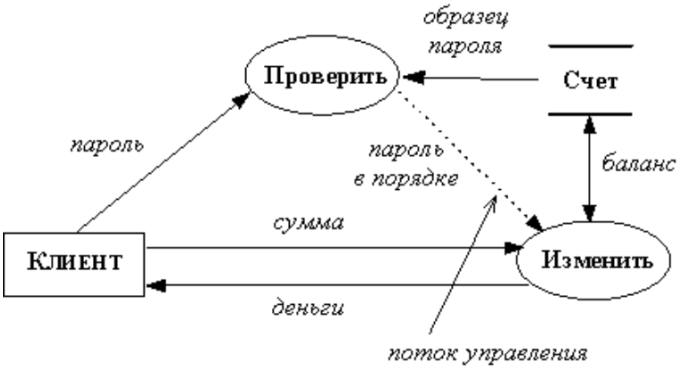
Сховище даних - це пасивний об'єкт у складі ДПД, в якому дані зберігаються для подальшого доступу. Сховище даних допускає доступ до збережених в ньому даними в порядку, відмінному від того, в якому вони були туди поміщені. Агрегатні сховища даних, як наприклад, списки і таблиці, забезпечують доступ до даних в порядку їх надходження, або по ключам. Приклади сховищ даних наведені на малюнку



Сховища даних

Потоки управління

ДПД показує всі шляхи обчислення значень, але не показує в якому порядку значення обчислюються. Рішення про порядок обчислень пов'язані з управлінням програмою, яке відбивається в динамічній моделі. Ці рішення, що виробляються спеціальними функціями, або предикатами, визначають, чи буде виконаний той чи інший процес, але при цьому не передають процесу ніяких даних, так що їх включення в функціональну модель необов'язково. Проте іноді буває корисно включати зазначені предикати в функціональну модель, щоб в ній були відображені умови виконання відповідного процесу. Функція, що приймає рішення про запуск процесу, будучи включеною в ДПД, породжує в ДПД потік управління (він зображується пунктирною стрілкою).



Потік управління

На рис. зображений приклад потоку управління: клієнт, який бажає зняти частину своїх грошей з рахунку в банку, вводить в АТМ пароль і необхідну суму, однак фактичне зняття і видача грошей відбувається тільки в тому випадку, коли введений пароль збігається з його зразком.

Незважаючи на те, що потоки управління іноді виявляються дуже корисними, слід мати на увазі, що включення їх в ДПД призводить до дублювання інформації, що входить в динамічну модель.

Опис операцій

Процеси ДПД в кінці кінців повинні бути реалізовані як операції об'єктів. Кожен процес нижнього (базового) рівня, так само як і процеси верхніх рівнів, до складу яких входять процеси більш нижніх рівнів, реалізуються як операції. При цьому реалізація процесів верхніх рівнів може відрізнятися від їх подання на ДПД, так як при реалізації зазвичай проводиться їх оптимізація: в результаті оптимізації процеси нижніх рівнів, що становлять процес більш високого рівня можуть "злитися", після чого вони стануть невидимі.

Всі операції повинні бути специфіковані. Специфікація операції містить її сигнатуру (ім'я операції, кількість, порядок і типи її параметрів, кількість, порядок і типи повертаються нею значень) і опис її ефекту (дії, перетворення). Для опису ефекту операції можна використовувати:

- математичні формули;
- табличні функції: таблиці, зіставляють вихідні значення вхідним;
- рівняння, що зв'язують вхідні і вихідні значення;
- аксіоматичне визначення операцій за допомогою перед- і пост-умов;
- таблиці прийняття рішень;
- псевдокод;
- природна мова.

Приклад опису операції (ефект її описаний на природній мові) наведено далі:

Специфікація операції ізміните_счет (при описі ефекту операції використані операції отмените_проводку, видать_запрос, видать_деньгі, дебетовать_счет і кредітовать_счет)

ізміните_счет (рахунок, сума, від_проводкі) -> гроші, квитанція

якщо сума знімається і більше балансу рахунку,

то "отмените_проводку"

якщо сума знімається і менше балансу рахунку,

то "дебетовать_счет" і "видать_деньгі"

якщо сума вноситься на рахунок

то "кредітовать_счет"

якщо запит

то "видать_запрос"

у всіх випадках:

квитанція повинна містити номер АТМ, дату, час,

номер рахунку, вид проводки, суму проводки (якщо вона є),

новий баланс рахунку

Зовнішня специфікація операції описує тільки ті зміни, які видно поза операції. Операція може бути реалізована таким чином, що при її виконанні будуть використовуватися деякі значення, певні всередині операції (наприклад, з метою оптимізації), деякі з цих значень можуть навіть бути частиною стану об'єкта. Ці деталі реалізації операції приховані від інших об'єктів і не беруть участь у визначенні зовнішнього ефекту операції.Зміни внутрішнього стану об'єкта, що не видні поза ним, не змінюють значення об'єкта.

Все нетривіальні операції можна розділити на три категорії: запити, дії і активності. Запитом називається операція без побічних ефектів над видимим ззовні об'єкта його станом (чиста функція). Запит, у якого немає параметрів, крім цільового об'єкта, є похідним атрибутом. Наприклад, для точки на координатній площині, радіус і полярний кут - похідні атрибути; з цього прикладу видно, що між основними і похідними атрибутами немає принципової різниці, і вибір основних атрибутів багато в чому випадковий.

Дією називається операція, що має побічні ефекти, які можуть впливати на цільовий об'єкт і на інші об'єкти системи, які досяжні з цільового об'єкта. Дія не займає часу (логічно, воно відбувається миттєво). Кожна дія може бути визначено через ті зміни, які воно виробляє в стані об'єкта, змінюючи значення його атрибутів і зв'язків; в якому порядку проводяться ці зміни, несуттєво: ми вважаємо, що всі вони відбуваються одночасно і миттєво. Найбільш поширеним способом опису дії є завдання алгоритму його виконання на комп'ютері.

Активністю називається операція, виконувана об'єктом, або над об'єктом, виконання якої займає певний час. Активність має побічні ефекти. Активності можуть бути тільки у активних об'єктів, так як пасивні об'єкти є просто склади даних.

Обмеження

Обмеження вказує на залежність між відповідними значеннями двох об'єктів, або між різними значеннями одного об'єкта. Обмеження може бути виражено у вигляді деякої функції (кількісне обмеження), або відносини (якісне обмеження). Нас цікавлять обмеження на атрибути об'єктів, а також на стану і події. Важливим видом обмежень є інваріанти: твердження про те, що значення деякої функції від атрибутів, станів і подій залишається постійним при функціонуванні об'єкта.

МЕТОДИЧНІ ВКАЗІВКИ

із самостійної роботи студентів

1. Тема: Динамічна модель системи

2. Навчальна мета. Студент повинен:

знати: визначення і призначення динамічної моделі даних

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Події, стани об'єктів і діаграми станів
2) Умови
3) Активності і дії
4) Одночасні дії. Синхронізація
5) Вкладені діаграми класів

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Призначення динамічної моделі
2) Побудова динамічної млделі
3) Що означає синхронізація в системі

5.Контроль засвоєння теми

Різнорівневі питання

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X
3. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X

Динамічна модель системи

1) Події, стани об'єктів і діаграми станів
2) Умови
3) Активності і дії
4) Одночасні дії. Синхронізація
5) Вкладені діаграми класів

Об'єктна модель являє статичну структуру проектованої системи (підсистеми). Однак знання статичної структури недостатньо, щоб зрозуміти і оцінити роботу підсистеми. Необхідно мати кошти для опису змін, які відбуваються з об'єктами і їх зв'язками під час роботи підсистеми. Одним з таких засобів є динамічна модель підсистеми. Вона будується після того, як об'єктна модель підсистеми побудована і попередньо узгоджена і налагоджена. Динамічна модель підсистеми складається з діаграм станів її об'єктів і підсистем.

Події, стани об'єктів і діаграми станів

Поточний стан об'єкта характеризується сукупністю поточних значень його атрибутів і зв'язків. Під час роботи системи складові її об'єкти взаємодіють один з одним, в результаті чого змінюються їх стану. Одиницею впливу є подія: кожна подія призводить до зміни стану одного або декількох об'єктів в системі, або до виникнення нових подій. Робота системи характеризується послідовністю відбуваються в ній подій.

п о д і ї

Подія відбувається в певний момент часу (нерідко воно використовується для визначення відповідного моменту часу). Приклади подій: старт ракети, старт забігу на 100 м, початок проводки (в банківській мережі), видача грошей і т.п. Подія не має тривалості (точніше, воно займає пренебрежимо малий час).

Одне з подій може логічно передувати іншому, або слідувати за іншим, або вони можуть бути незалежними; прикладами логічно (причинно) пов'язаних подій є старт і фініш одного забігу, початок проводки і видача грошей клієнтові (в результаті цієї проводки), прикладами незалежних подій - старт ракети і фініш забігу, проводки, запущені з різних АТМ, і т.п. Якщо події не мають причинного зв'язку (тобто вони логічно незалежні), вони називаються незалежними (concurrent); такі події не впливають один на одного. Незалежні події не має сенсу організовувати, так як вони можуть відбуватися в довільному порядку. Модель розподіленої системи обов'язково повинна містити незалежні події і активності.

Події передають інформацію з одного об'єкта на інший. Існують класи подій, які просто сигналізують про те, що щось сталося чи відбувається (приклади: загоряння лампочки ліфта, гудок в телефонній трубці). У програмуванні розглядаються виняткові події (іноді їх називають виключеннями), які сигналізують про порушення роботи апаратури, або програмного забезпечення.

Сценарії і траси подій

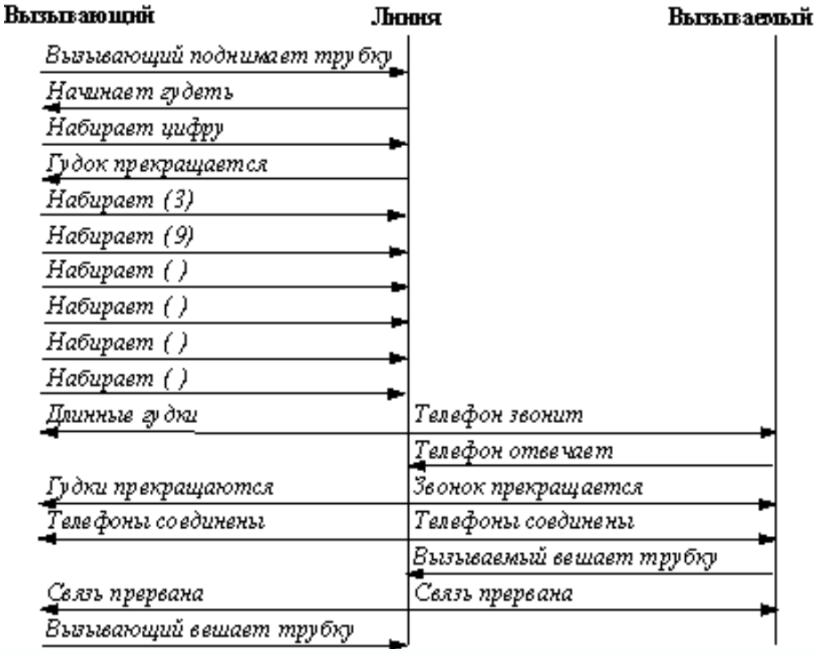
Сценарієм називається послідовність подій, яка може мати місце при конкретному виконанні системи. Сценарії можуть мати різні області впливу: вони можуть включати всі події, що відбуваються в системі, або тільки події, що виникають і впливають тільки на певні об'єкти системи.

Долі наведено приклад сценарію користування телефонною лінією. Кожна подія в цьому сценарії передає інформацію з одного об'єкта на інший; наприклад подія починається довгий гудок передає сигнал від телефонної лінії до викликає (користувачеві). При аналізі динаміки роботи системи необхідно скласти і розглянути кілька сценаріїв, що відображають типові варіанти її роботи.

```
викликає  знімає  трубку
починається  довгий  гудок
викликає  набирає  цифру  (9)
гудок  припиняється
викликає  набирає  цифру  (3)
викликає  набирає  цифру  (9)
викликає  набирає  цифру  ()
викликає  набирає  цифру  ()
викликає  набирає  цифру  ()
викликає  набирає  цифру  ()
викликаний  телефон  починає  дзвонити
викликає  чує  гудки
```

```
викликаний  телефон  відповідає
гудки  припиняються
телефони  з'єднані
викликаний  по  телефону  вішає  трубку
телефони  роз'єднані
викликає  вішає  трубку
```

Наступним етапом після розробки і аналізу сценаріїв є визначення об'єктів, що генерують і приймають кожну подію сценарію. Послідовності подій з прив'язкою до об'єктів проектованої системи зручно представляти на діаграмах, які називаються трасами подій. Приклад траси подій для розмови по телефону представлений на малюнку 2.45. Вертикальні лінії зображують на цій діаграмі об'єкти, а горизонтальні стрілки - події (стрілка починається в об'єкті, що генерує подія, і закінчується в об'єкті, що приймає подія). Пізніші події поміщені нижче попередніх, одночасні - на одному рівні.



Траса подій для розмови по телефону

Стани

Стан визначається сукупністю поточних значень атрибутів і зв'язків об'єкта. Наприклад, банк може мати стану платоспроможний і неплатоспроможний (коли велика частина банків одночасно виявляється в другому стані, настає банківська криза).

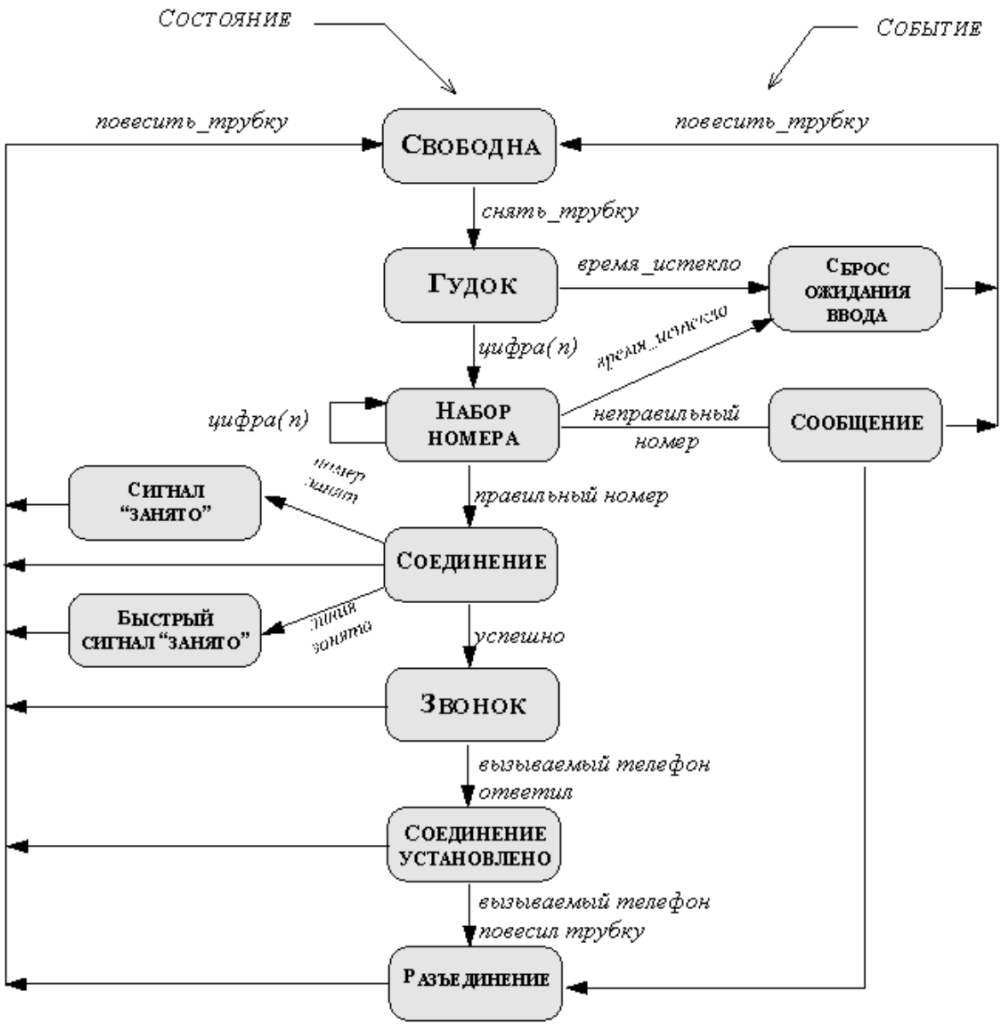
Стан визначає реакцію об'єкта на яке надходить в нього подія (в тому, що реакція різна неважко переконатися за допомогою банківської картки: в залежності від стану банку обслуговування (реакція банку на пред'явлення картки) буде різним). Реакція об'єкта на подія може включати деяку дію і / або переклад об'єкта в новий стан.

Об'єкт зберігає свій стан протягом часу між двома послідовними подіями, які він приймає: події представляють моменти часу, стану - відрізки часу; стан має тривалість, яка зазвичай дорівнює відрізку часу між двома послідовними подіями, прийнятими об'єктом, але іноді може бути більше.

При визначенні станів ми не розглядаємо тих атрибутів, які не впливають на поведінку об'єкта, і об'єднуємо в один стан все комбінації значень атрибутів і зв'язків, які дають однакові реакції на події.

Діаграми станів

Діаграма станів пов'язує події і стани. При прийомі події наступний стан системи залежить як від її поточного стану, так і від події; зміна стану називається переходом. Діаграма станів - це граф, вузли якого представляють стану, а спрямовані Деги, помічені іменами відповідних подій, - переходи. Діаграма станів дозволяє отримати послідовність станів по заданій послідовності подій.

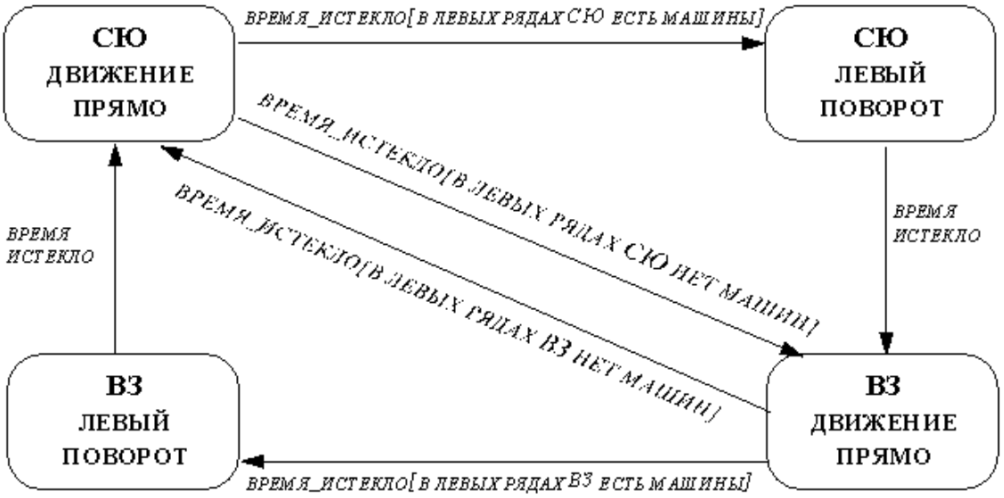


Діаграма станів телефонної лінії

Умови

Умова - це логічна (булева) функція від значень об'єктів, як наприклад, картку вдалося прочитати, температура нижче нуля і т.п. Умова може виконуватися протягом деякого відрізка часу; подія, на відміну від умови, відбувається миттєво і не має тривалості в часі.

Умови можуть використовуватися як обмеження на переходи: умовний перехід виконується тільки тоді, коли і відбулося відповідне подія, і виконана умова цього переходу (діаграма станів, представлена на рис, демонструє це на прикладі автомобільного руху по магістралях "Північ-Південь" і "Захід -Восток "). На діаграмах станів умови записуються слідом за подіями в квадратних дужках.

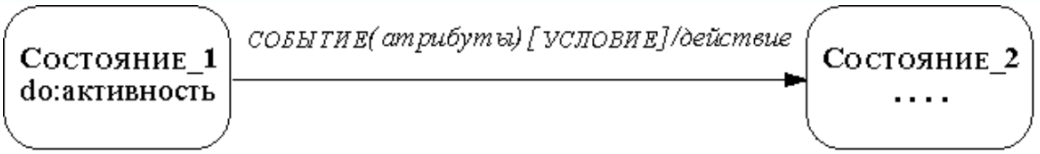


Діаграма станів, на якій вказані умови

Активності і дії

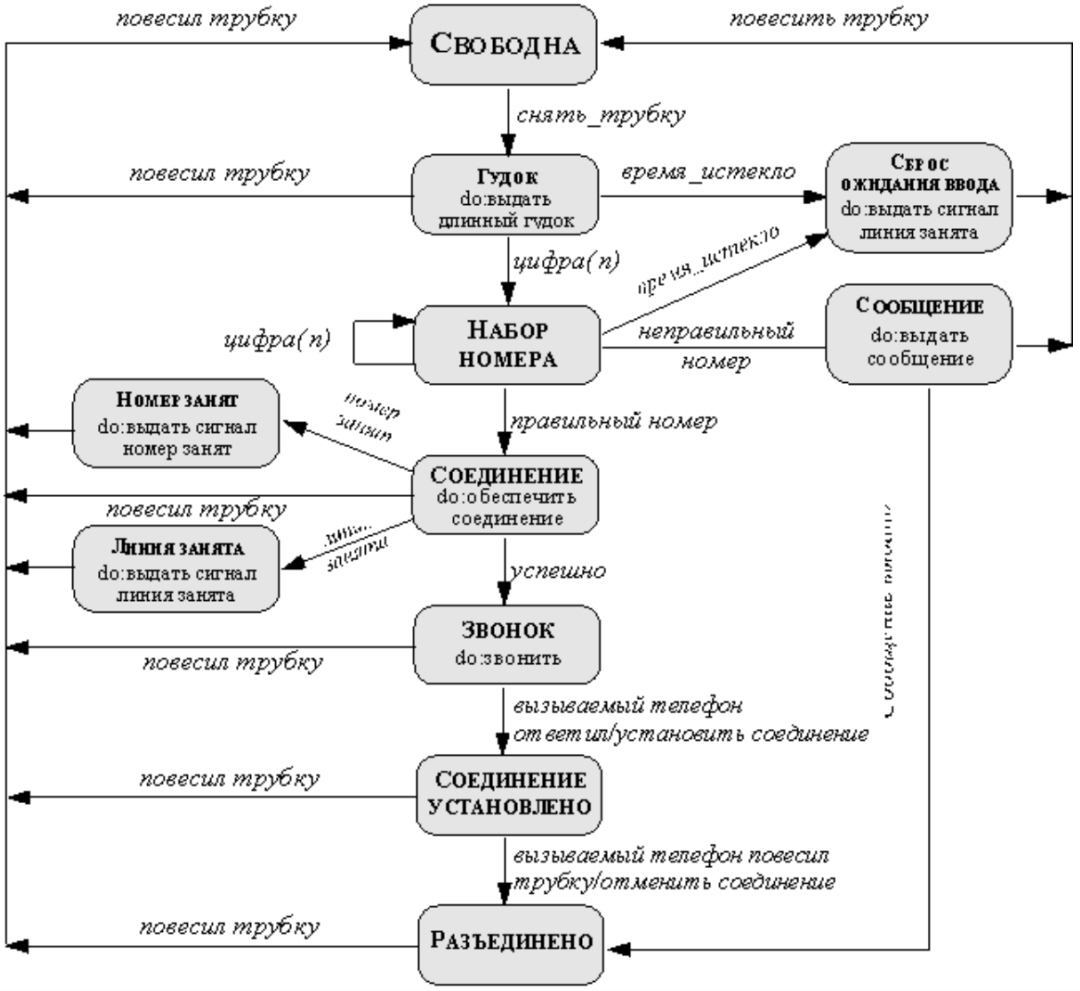
Діаграма станів була б не дуже корисною, якби вона містила тільки переходи (безумовні та умовні), відповідні генеруються під час роботи системи подій. Будучи описом поведінки об'єкта, діаграма станів повинна описувати, що робить об'єкт у відповідь на перехід в якийсь стан або на виникнення деякої події. Для цього в діаграму станів включаються опису активностей і дій.

Активністю називається операція, пов'язана з будь-яким станом об'єкта (вона виконується, коли об'єкт потрапляє в зазначений стан); виконання активності вимагає певного часу. Приклади активностей: видача картинки на екран телевізора, телефонний дзвінок, зчитування порції файлу в буфер і т.п .; іноді активністю буває просто припинення виконання програми (пауза), щоб забезпечити необхідний час перебування у відповідному стані (це буває особливо важливо для паралельної асинхронної програми). Активність пов'язана зі станом, тому на діаграмі станів вона позначається через "do: імя_актівності" у вузлі, що описує відповідний стан



Вказівка активностей і дій на діаграмі станів

Дією називається миттєва операція, пов'язана з подією: при виникненні події відбувається не тільки перехід об'єкта в новий стан, а й виконується дія, пов'язане з цією подією. Наприклад, в телефонній мережі подія повісив трубку супроводжується дією роз'єднати зв'язок. Дія вказується на діаграмі станів слідом за подією, якому воно відповідає, і його ім'я (або опис) відділяється від імені події косою рисою ("/")



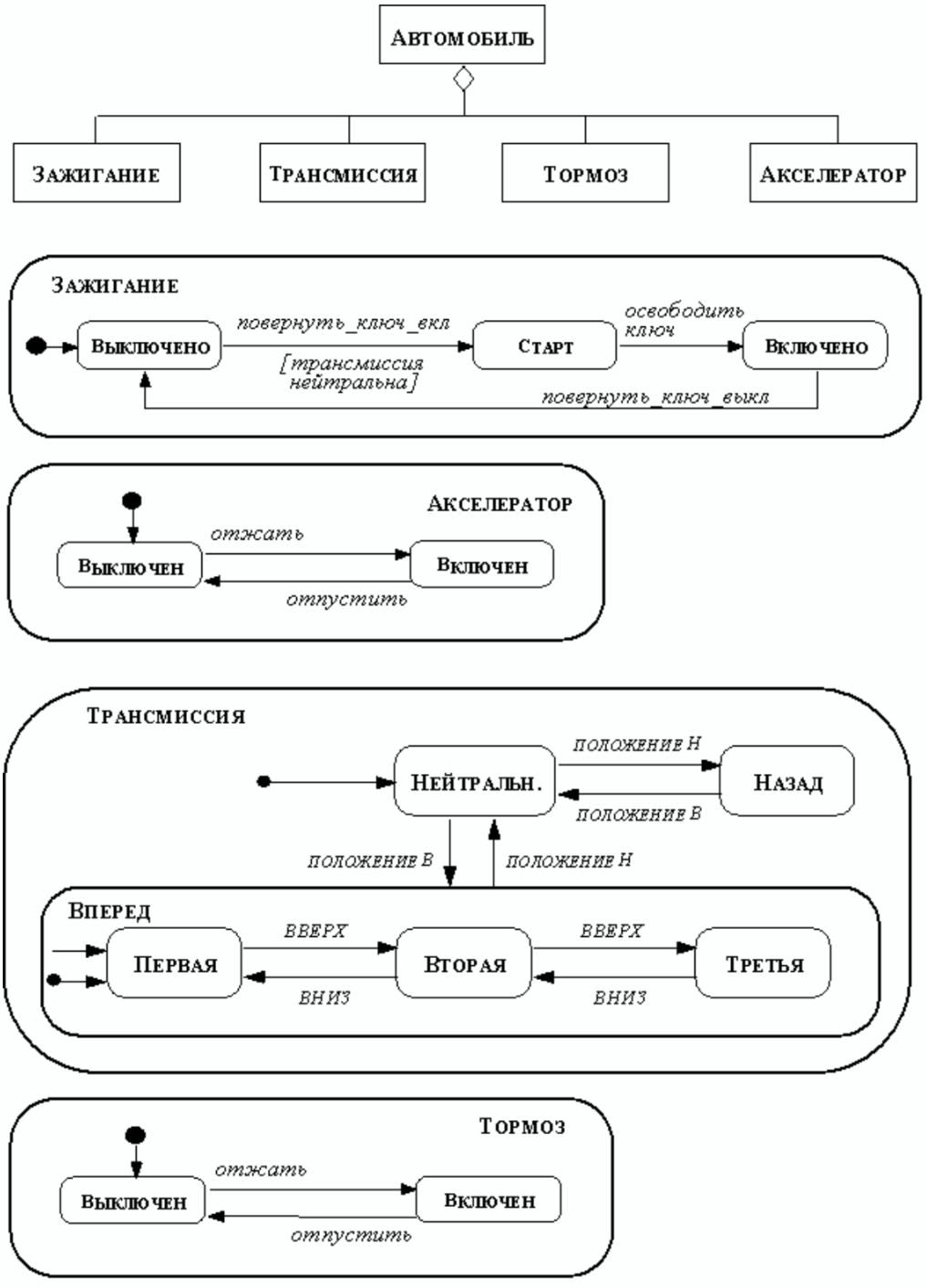
Діаграма станів телефонної лінії, на якій вказані активності і дії

Дії можуть також представляти внутрішні операції управління об'єкта, як наприклад, присвоювання значень атрибутів або генерація інших подій.

Одночасні події. Синхронізація

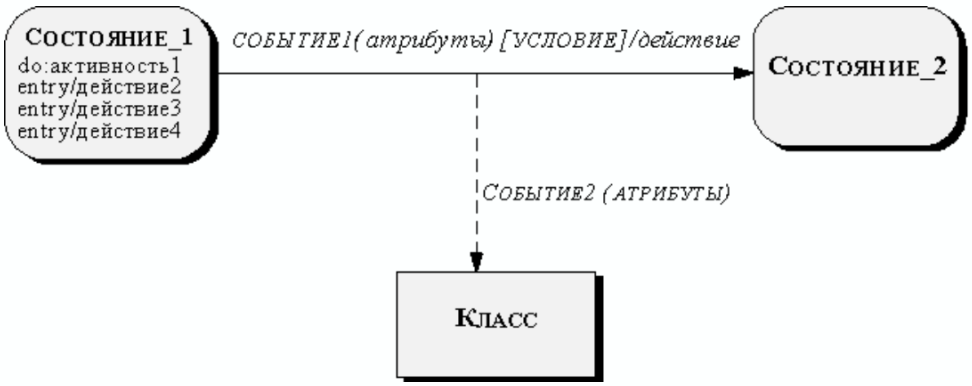
На рис. представлена діаграма станів складеного об'єкта: він складається з чотирьох паралельно і незалежно працюють компонентів (стан кожного компоненту не залежить від станів інших компонентів). Стан такого об'єкта визначається як кортеж, членами якого є стани кожного зі складових об'єктів. Діаграма станів при цьому розпадається на чотири незалежні діаграми станів кожного з компонентів.

У системах, що складаються з декількох паралельно працюючих об'єктів, іноді буває необхідно узгодити (синхронізувати) роботу двох або більше об'єктів. Для цього теж використовуються події, так як синхронізація по суті полягає в необхідності затримати перехід одного з синхронізуються об'єктів в черговий стан, поки інший об'єкт не прийде в деяке фіксоване стан, а переходи зі стану в стан управляються подіями.Синхронізуючий подія може вироблятися в будь-якому з синхронізуються об'єктів, або в будь-якому третьому об'єкті.



Діаграма станів складеного об'єкта (підсистеми) (А)

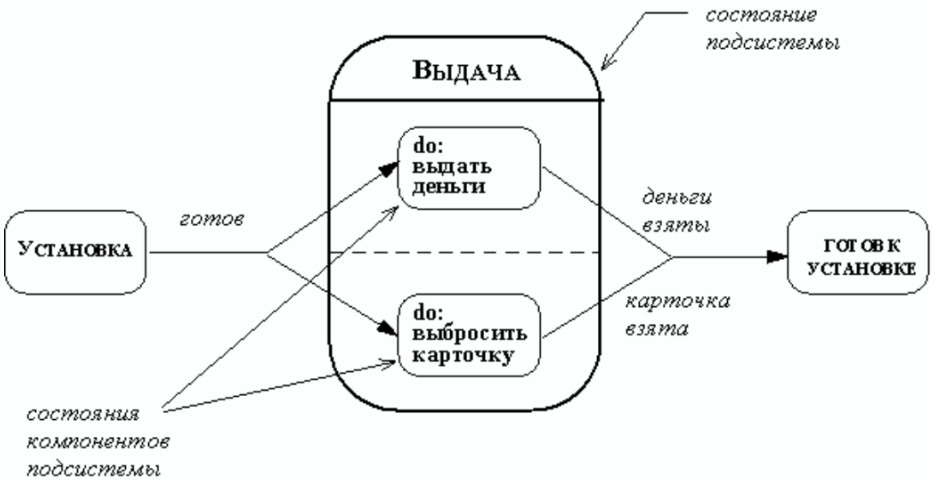
Якщо синхронізуючий подія виробляється поза потрібно синхронізувати об'єкта, воно може бути передано йому з іншого об'єкта; на діаграмі це позначається як показано на рис.



Передача події з одного об'єкта іншому

Якщо об'єкт отримує події з кількох незалежних об'єктів, то стан, в яке він перейде, може залежати від порядку, в якому будуть отримані ці події (а цей порядок випадковий, так як об'єкти незалежні). Це називаєтьсяумовою конкуренції. Одним із завдань проектування є виняток небажаних умов конкуренції. Це досягається за допомогою синхронізації.

Синхронізація використовується і в разі, коли в будь-якому стані потрібно паралельно виконати кілька активностей. Як приклад розглянемо пристрій виведення АТМ (рисунок). Воно одночасно (паралельно) видає готівку і картку; обидві ці операції можна розглядати як складову активність, що складається з двох паралельно працюючих активностей (пунктирна лінія на діаграмі станів ділить стан на дві області, в кожній з яких виконується одна із зазначених активностей). Поділ управління на два паралельні потоки схематично показано у вигляді стрілки, яка поділяється на дві: подія готовий викликає перехід зі стану установка відразу в два паралельних підстану стану видача; перехід в наступний стан відбувається по двом подіям гроші взяті і картка взята; якщо будь-яка з цих подій відбудеться раніше іншого, переходу все одно не буде, поки не відбудеться і друга подія (в цьому і полягає синхронізація).



Синхронізація в підсистемі

З розглянутого прикладу видно, що в різних станах може паралельно працювати різну кількість процесів (активностей).

Вкладені діаграми станів

Діаграма станів компонента трансмісія (А) має вузол (стан) вперед, який сам є діаграмою станів (вкладена діаграма станів). Інтерпретація цієї діаграми наступна: трансмісія має три стани: нейтральне, тому і вперед; стан вперед має три підстану (вони уточнюють стан вперед): перша, друга і третя. На діаграмах станів суперсостояніе зображується як прямокутник із закругленими кутами, всередину якого поміщаються всі його підстану.

МЕТОДИЧНІ ВКАЗІВКИ

із самостійної роботи студентів

1. Тема: Класифікація баз даних

2. Навчальна мета. Студент повинен:

знати: класифікацію баз даних за різними ознаками

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Класифікація БД за моделлю даних
2) Класифікація БД за технологією фізичного зберігання
3) Класифікація БД за вмістом
4) Класифікація БД за ступенем розподіленості
5) Класифікація БД за видом структурування
6) За технологією обробки даних
7) За способом доступу до даних

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Що лежить в основі класифікації
2) Відмінності типів баз даних

5.Контроль засвоєння теми

Різнорівневі питання

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)

2. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

3. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X

Класифікація баз даних

Існує величезна кількість різновидів баз даних, що відрізняються за критеріями (наприклад, в Енциклопедії технологій баз даних визначаються понад 50 видів БД).

Відзначимо тільки основні класифікації.

Класифікація БД *за моделлю даних*:

- ієрархічні,
- мережеві,
- реляційні,
- об'єктні,
- об'єктно-орієнтовані,
- об'єктно-реляційні.

Класифікація БД *за технологією фізичного зберігання*:

- БД у вторинній пам'яті (традиційні);
- БД в оперативній пам'яті (in-memory databases);
- БД у третинній пам'яті (tertiary databases).

Класифікація БД *за вмістом*:

- географічні.
- історичні.
- наукові.
- мультимедійні.

Класифікація БД *за ступенем розподіленості*:

- централізовані (зосереджені);
- розподілені.

Класифікація БД *за видом структурування*

Окреме місце в теорії та практиці займають просторові (англ. spatial), тимчасові, або темпоральні (temporal) і просторово-часові (spatial-temporal) БД.

Ієрархічні бази даних можуть бути представлені як дерево, що складається з об'єктів різних рівнів. Верхній рівень займає один об'єкт, другий - об'єкти другого рівня і т.д.

Між об'єктами існують зв'язки, кожен об'єкт може включати в себе декілька об'єктів більш низького рівня. Такі об'єкти перебувають у відношенні предка (об'єкт більш близький до кореня) до нащадка (об'єкт більш низького рівня), при цьому можлива ситуація, коли об'єкт-предок не має нащадків або має їх декілька, тоді як у об'єкта-нащадка обов'язково тільки один предок. Об'єкти, що мають загального предка, називаються близнюками.

Мережеві бази даних подібні до ієрархічних, за винятком того, що в них є покажчики в обох напрямках, які з'єднують споріднену інформацію.

До основних понять мережевої моделі бази даних відносяться: рівень, елемент (вузол), зв'язок.

Вузол - це сукупність атрибутів даних, що описують деякий об'єкт. На схемі ієрархічного дерева вузли представляються вершинами графа. У мережній структурі кожен елемент може бути пов'язаний з будь-яким іншим елементом.

Незважаючи на те, що ця модель вирішує деякі проблеми, пов'язані з ієрархічною моделлю, виконання простих запитів залишається досить складним процесом.

Також, оскільки логіка процедури вибірки даних залежить від фізичної організації цих даних, то ця модель не є повністю незалежною від програми. Іншими словами, якщо необхідно змінити структуру даних, то потрібно змінити і додаток.

Реляційна модель орієнтована на організацію даних у вигляді двовимірних таблиць. Кожна реляційна таблиця являє собою двовимірний масив і має наступні властивості:

- кожен елемент таблиці - один елемент даних;
- всі осередки в стовпчику таблиці однорідні, тобто всі елементи в стовпчику мають однаковий тип (числовий, символьний тощо);
- кожен стовпчик має унікальне ім'я;
- однакові рядки в таблиці відсутні;
- порядок проходження рядків і стовпчиків може бути довільним.

Об'єктна СУБД ідеально підходить для інтерпретації складних даних, на відміну від реляційних СУБД, де додавання нового типу даних досягається ціною втрати продуктивності або за рахунок різкого збільшення термінів і вартості розробки додатків. Об'єктна база, на відміну від реляційної, не вимагає модифікації ядра при додаванні нового типу даних. Новий клас і його екземпляри просто надходять у зовнішні структури бази даних. Система управління ними залишається без змін.

Об'єктно-орієнтована база даних (ООБД) - база даних, в якій дані оформлені у вигляді моделей об'єктів, що включають прикладні програми, які управляються зовнішніми подіями. Результатом поєднання можливостей (особливостей) баз даних і можливостей об'єктно-орієнтованих мов програмування є об'єктно-орієнтовані системи управління базами даних (ООСУБД). ООСУБД дозволяють працювати з об'єктами баз даних також, як з об'єктами у програмуванні в об'єктно-орієнтованих мовах програмування. ООСУБД розширює мови програмування, прозоро вводячи довготривалі дані, управління паралелізмом, відновлення даних, асоційовані запити й інші можливості.

Об'єктно-орієнтовані бази даних звичайно рекомендовані для тих випадків, коли потрібна високопродуктивна обробка даних, які мають складну структуру.

Система, яка забезпечує об'єктну інфраструктуру і набір реляційних розширювачів, називається "*об'єктно-реляційною*".

Об'єктно-реляційні системи поєднують переваги сучасних об'єктно-орієнтованих мов програмування з такими властивостями реляційних систем як множинні представлення даних і високорівневі непроцедурні мови запитів.

***За технологією обробки даних* бази даних поділяються на централізовані й розподілені.**

Централізована база даних зберігається у пам'яті однієї обчислювальної системи. Якщо ця обчислювальна система є компонентом мережі ЕОМ, можливий розподілений доступ до такої бази. Такий спосіб використання баз даних часто застосовують у локальних мережах ПК.

Розподілена база даних складається з декількох, можливо пересічних або навіть дублюючих одна одну частин, які зберігаються в різних ЕОМ обчислювальної мережі. Робота з такою базою здійснюється за допомогою системи управління розподіленою базою даних (СУРБД).

***За способом доступу* до даних** бази даних поділяються на

- бази даних з локальним доступом і
- бази даних з віддаленим (мережевим) доступом.

Системи централізованих баз даних з мережевим доступом припускають різні архітектури подібних систем:

- файл-сервер;
- клієнт-сервер.

МЕТОДИЧНІ ВКАЗІВКИ

із самостійної роботи студентів

1. Тема: Дванадцять правил Кодда для реляційної СКБД

2. Навчальна мета. Студент повинен:

знати: правила і їх трактовку

3.Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1)Правило інформації
2)Правило підтримки недійсних значень
3)Правило динамічного каталогу, заснованого на реляційній моделі
4)Правило динамічного каталогу, заснованого на реляційній моделі
5)Правило вичерпної підмови даних
6)Правило відновлення представлень
7)Правило додавання, відновлення і вилучення
8)Правило незалежності фізи данихчних
9)Правило незалежності логічних даних
10)Правило незалежності умов цілісності
11)Правило незалежності поширення
12)Правило одиничності

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

Використання правил

5. Контроль засвоєння теми

Різнорівневі питання

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)

2. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

3. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X

Дванадцять правил Кодда, яким повинна відповідати реляційна СКБД

1. Правило інформації.

Вся інформація в базі даних повинна бути надана винятково на логічному рівні і тільки одним способом - у виді значень, що містяться в таблицях.

2.Правило гарантованого доступу.

Логічний доступ до всіх і кожного елементу даних (атомарному значенню) у реляційній базі даних повинний забезпечуватися шляхом використання комбінації імені таблиці, первинного ключа та імені стовпця.

3.Правило підтримки недійсних значень.

У реляційній базі даних повинна бути реалізована підтримка недійсних значень, що відрізняються від рядка символів нульової довжини, рядка символів пробілів чи нуля будь-якого іншого числа і використовуватися для представлення відсутніх даних незалежно від типу цих даних.

4. Правило динамічного каталогу, заснованого на реляційній моделі.

Опис бази даних на логічному рівні необхідно представити в тому ж виді, що й основні дані, щоб користувачі, які володіють відповідними правами, могли працювати з ним за допомогою тієї ж реляційної мови, яку вони застосовують для роботи з основними даними.

5.Правило вичерпної підмови даних.

Реляційна система може підтримувати різні мови і режими взаємодії з користувачем (наприклад, режим питань і відповідей). Однак повинна існувати принаймні одна мова, оператори якої підтримують наступні елементи: визначення даних; визначення представлень; обробку даних (інтерактивну і програмну); умови цілісності; ідентифікацію прав доступу; границі транзакцій (початок, завершення і скасування).

6. Правило відновлення представлень.

Усі представлення, які теоретично можна обновити, повинні бути доступні для відновлення.

7. Правило додавання, відновлення і вилучення.

Можливість працювати з відношенням як з одним операндом повинна існувати не тільки при читанні даних, але і при додаванні, відновленні і вилученні даних.

8. Правило незалежності фізи данихчних.

Прикладні програми й утиліти для роботи з даними повинні на логічному рівні залишатися недоторканими при будь-яких змінах методів збереження даних чи методів доступу до них.

9. Правило незалежності логічних даних.

Прикладні програми й утиліти для роботи з даними повинні на логічному рівні залишатися недоторканими при внесенні в базові таблиці будь-яких змін, які теоретично дозволяють зберегти недоторканими дані, що містяться в цих таблицях.

10. Правило незалежності умов цілісності.

Повинна існувати можливість визначати умови цілісності, специфічні для конкретної реляційної бази даних, на підмові реляційної бази даних і зберігати їх у каталозі, а не в прикладній програмі.

11. Правило незалежності поширення.

Реляційна СКБД не повинна залежати від потреб конкретного клієнта.

12. Правило одиничності.

Якщо в реляційній системі є низькорівнева мова (що обробляє один запис за один раз), то повинна бути відсутня можливість використання її для того, щоб обійти правила й умови цілісності, виражені на реляційному мові високого рівня (що обробляє кілька записів за один раз).

Правило 2 вказує на роль первинних ключів при пошуку інформації в базі даних. Ім'я таблиці дозволяє знайти необхідну таблицю, ім'я стовпця дозволяє знайти необхідний стовпець, а первинний ключ дозволяє знайти рядок, який містить шуканий елемент даних.

Правило 3 вимагає, щоб відсутні дані можна було представити за допомогою недійсних значень (NULL).

Правило 4 говорить, що реляційна база даних повинна сама себе описувати. Іншими словами, база даних повинна містити набір системних таблиць, що описують структуру самої бази даних.

Правило 5 вимагає, щоб СКБД використовувала мову реляційної бази даних, наприклад SQL, хоча явно SQL у правилі не згадають. Така мова повинна підтримувати всі основні функції СКБД - створення бази даних, читання і введення даних, реалізацію захисту бази даних і т.д.

Правило 6 дозволяють показувати різним користувачам різні фрагменти структури бази даних.

Правило 7 акцентує увага на тому, що бази даних по своїй природі орієнтовані на множини. Воно вимагає, щоб операції додавання, вилучення і відновлення можна було виконувати над множинами рядків. Це правило призначене для того, щоб заборонити реалізації, у яких підтримуються тільки операції над одним рядком.

Правила 8 і 9 стверджують, що конкретні способи реалізації чи збереження доступу, які використовуються в СКБД, і навіть зміни структури таблиць бази даних не повинні впливати на можливість користувача працювати з даними.

Правило 10 вимагає, щоб мова бази даних підтримували обмежувальні умови, які накладаються на дані, що вводяться, і дії, що можуть бути виконані над даними.

Правило 11 говорить, що мова бази даних повинна забезпечувати можливість роботи з розподіленими даними, розташованими на інших комп'ютерних системах.

І, нарешті, правило 12 запобігає використання інших можливостей для роботи з базою даних, крім мови бази даних, оскільки це може порушити її цілісність.

МЕТОДИЧНІ ВКАЗІВКИ

із самостійної роботи студентів

1. Тема: Реляційні мови обробки даних – реляційна алгебра і реляційне числення

2. Навчальна мета. Студент повинен:

знати: операції реляційної алгебри і реляційного числення

уміти: виконувати дії над відношеннями

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Реляційна алгебра
   - об'єднання;
   - різниця;
   - твір;
   - перетин;
   - проекція;
   - вибір;
   - з'єднання;
   - розподіл.
2) Реляційне числення

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Використання операторів реляційної алгебри
2) Реляційне числення кортежів і доменів

5.Контроль засвоєння теми

Письмова робота.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. http://books.ifmo.ru/file/pdf/677.pdf

Реляційні мови обробки даних – реляційна алгебра і реляційне числення

Для управління реляційної базою даних Е. Ф. Кодд ввів реляційні мови обробки даних - реляционную алгебру і реляційне числення.

Реляційна алгебра - це процедурна мова обробки реляційних таблиць. Це означає, що в реляційній алгебрі використовується покроковий підхід до створення реляційних таблиць, містять відповіді на запити.

Реляційне числення - непроцедурна мова. У реляційному обчисленні запит створюється шляхом визначення таблиці запиту за один крок.

Кодд показав логічну еквівалентність реляційної алгебри і реляційного числення. Це означає, що будь-який запит, який можна сформулювати за допомогою реляційного числення, також можна сформулювати, користуючись реляційної алгеброю, і навпаки. І реляційна алгебра, і реляційне числення в тому вигляді, як вони були сформульовані Коддом, є теоретичними мовами.

Реляційна алгебра

Оператори реляційної алгебри використовують одне або два з існуючих відносин для створення нового відносини. Реляційна алгебра (або алгебра відносин) являє собою сукупність операцій високого рівня над відносинами. реляційна алгебра визначає наступні операції:

- об'єднання;
- різниця;
- твір;
- перетин;
- проекція;
- вибір;
- з'єднання;
- розподіл.

Перші чотири операції взяті Кожному з математичної теорії множин і практично збігаються з операціями теорії множин. Наступні чотири - нові операції, що стосуються лише реляційної моделі даних.

Об'єднання (Union)

Нехай є відносини r і s, тоді відношення t = r ∪ s називається об'єднанням r і s, якщо кожен кортеж, що належить t, належить або r, або s, або їм обом.

Приклад

Нехай дано відносини:

r – виріб 1

Код_вир	Назва	Вага
01	А	1
03	В	2
04	С	3

s – виріб 2

Код_вир	Назва	Вага
02	Д	2
03	В	2
04	С	3

Необхідно сформувати відповідь на такий запит: які типи деталей входять до складу обох виробів? Для досягнення цієї мети необхідно виконати операцію t = r ∪ s. результуюче відношення містить всі деталі, які входять до складу обох виробів.

Код_вир	Назва	Вага
01	А	1
02	Д	2
03	В	2
04	С	3

Різниця

Нехай є два відносини r і s, тоді відношення t = r-s називається різницею r і s, якщо кожен кортеж, що належить t, належить r, але не належить s. Операція застосовується до відносин однією арності. Нехай відношення r являє потреби в деяких видах деталей, а відношення s - відомості про тих видах деталей, які фірма може призвести сама, тоді відношення t = r-s містить відомості про тих видах деталей, які потрібно придбати.

r – потреба

Код_вир	Назва	Вага
01	А	1
02	Д	2
03	В	2
04	С	3
05	Е	1

s – можливість

Код_вир	Назва	Вага
02	Д	2
03	В	2
04	С	3

T = r – s

Код_вир	Назва	Вага
01	А	1
05	Е	1

Декартовий добуток

Ця операція не накладається ніяких обмежень на схеми вихідних відносин, і тому вона допустима для будь-яких двох відносин.

Під декартовим добутком двох відносин розуміється безліч впорядкованих пар кортежів. Нехай є два відносини r і s, тоді відношення t = r * s арності к = к1 + k2, де к1 - арность r, а k2 - арность s, називається декартовим твором r і s, якщо воно складається з кортежів, перші k1 компонентів яких утворюють кортежі з r, а інші k2 - з s.

Приклад

Нехай r → СТУДЕНТИ (Ном_зач_кн, ПІБ);
s → ІСПИТИ (Код_дісц, Назв_дісц, Дата, Оцінка),
тоді r * s → ЕКЗАМ_ВЕД (Ном_зач_кн, ПІБ, Код_дісц,Назв_дісц, Дата, Оцінка).

r – студенти

Номер_зал_кн	ПІБ
02-Е-01	Іваненко І.І.
02-Е-02	Петренко А.О.
02-Е-05	Сіренко П.П.

s – іспити

Код_дисц	Назва_дисц	Дата	Оцінка
01	Математика	10.01.15	
02	Фізика	15.01.15	
03	Іноз. мова	20.01.15	

Екзаменаційна відомість по всім дисциплінам

Номер_зал_кн	ПІБ	Код_дисц	Назва_дисц	Дата	Оцінка
02-Е-01	Іваненко І.І.	01	Математика	10.01.15	
02-Е-01	Іваненко І.І.	02	Фізика	15.01.15	
02-Е-01	Іваненко І.І.	03	Іноз. мова	20.01.15	
02-Е-02	Петренко А.О.	01	Математика	10.01.15	
02-Е-02	Петренко А.О.	02	Фізика	15.01.15	
02-Е-02	Петренко А.О.	03	Іноз. мова	20.01.15	
02-Е-05	Сіренко П.П.	01	Математика	10.01.15	
02-Е-05	Сіренко П.П.	02	Фізика	15.01.15	
02-Е-05	Сіренко П.П.	03	Іноз. мова	20.01.15	

Перетин (пересечение)

Нехай є два відносини r і s, тоді відношення t = r ∩ s називається перетином r і s, якщо кожен кортеж, що належить t, одночасно належить r і s. Операція застосовується до відносин однією арності. Справедлива наступна формула: t = r ∩ s = r - (r - s).

Приклад

Нехай є відносини:

r – потреба

Код_вир	Назва	Вага
01	А	1
02	Д	2
03	В	2
04	С	3
05	Е	1

s – можливість

Код_вир	Назва	Вага
02	Д	2
04	С	3
03	В	2
06	К	1

Сформуємо відповідь на такий запит: визначити деталі, що входять до склад обох виробів. Для цього необхідно виконати операцію перетину двох вихідних відносин. результат видається ставленням:

Код_вир	Назва	Вага
02	Д	2
04	С	3
03	В	2

Проекція (Project)

Оператор проекції (вертикальне підмножина) є унарним оператором на відносинах. Він здійснює вибір на безлічі стовпців.

Нехай щодо r (R) виділено деякий безліч атрибутів Y, тоді відношення t = πY (r) називається проекцією відносини r, якщо воно є вертикальним підмножиною стовпців відносини r з безлічі R.

Іншими словами, проекція R на Y є також ставлення, отримане викреслюванням стовпців, відповідних атрибутів R - Y, і видаленням, за визначенням відносини, які залишилися стовпців повторюваних рядків.

Нехай дано відношення r:

A	B	C
P	1	a
P	2	b
Q	2	c

Тоді:

$\pi_{AC(r)}$

A	C
P	a
P	d
Q	c

Вибір (Select)

Вибір або селекція - це одна з найважливіших операцій обробки інформації. Вона також як і попередня, відноситься до унарних операцій над відношенням. Результатом її застосування до відношення r є інше відношення, яке є підмножина кортежів відносини r, з певним значенням у виділеному атрибуті. Отже, результатом селекції відносини r по деякому Θ будемо вважати відношення t = δΘ (r), яке включає в себе кортежі відносини r, що задовольняють вказаним умові Θ. Умова Θ - це

формула, за якою визначається вибірка. Операндами в такій формулі є атрибути відносини, а знаками операцій - логічні операції та операції відносин.

$\delta C \neq b(r)$

A	B	C
P	1	a
Q	2	c

$\delta A \neq q \Lambda B > 1(r)$

A	B	C
P	2	b

Соединение (Join)

Нехай є два відносини r (S, У) і s (Y, Z) і деякий умова Θ, де X, У, Z-- непересічні безлічі атрибутів і Y- безліч атрибутів, загальних для r і s, тоді відношення t = r> Θ <s 97 називається Θ-з'єднанням r і s, якщо кожен кортеж, належить t, складається з кортежів r і s, при активному умови Θ.

Справедлива наступна формула:

t = δΘ (r * s),

тобто Θ-з'єднання являє собою декартовий добуток r і s, над яким виконана селекція по умові Θ.

Приклад

Із загальної екзаменаційної відомості з усіх дисциплін отримаємо екзаменаційну відомість з дисципліни математика. Для цього виконаємо операцію з'єднання (Join) за умови Код_дісц = "01"

Екзаменаційна відомість по дисциплінs Математика

Номер_зал_кн	ПІБ	Код_дисц	Назва_дисц	Дата	Оцінка
02-Е-01	Іваненко І.І.	01	Математика	10.01.15	
02-Е-02	Петренко А.О.	01	Математика	10.01.15	
02-Е-05	Сіренко П.П.	01	Математика	10.01.15	

Розподіл (деление)

Розподіл - це також бінарна несиметрична операція для отримання деякого відносини з двох вихідних, причому ступінь результуючого відношення не збігається зі ступенем жодного з операндів, а обчислюється як різниця між ступенем відносиниділеного і ступенем отношения-подільника.

Нехай є відносини r (X, У) арності k1 і p (Z) арності k2, де Y і Z визначені на одному домені, тоді відношення t = r ÷ p арності k1 - k2 називається розподілом r на p, якщо будь-який кортеж з t разом з будь-яким кортежем з p утворюють кортеж, наявний в r.

Приклад

Дано відносини, що містять відомості про іспити, які повинні були здавати студенти, і відомості про результати здачі цих іспитів:

VEDOM

Номер_зал_кн	ПІБ	Назва_дисц	Дата
02-Е-01	Іваненко І.І.	Фізика	10.01.15
02-Е-01	Іваненко І.І.	Хімія	14.01.15
02-Е-02	Сіренко П.П.	Фізика	10.01.15
02-Е-02	Сіренко П.П.	Хімія	14.01.15
02-Е-05	Коченко І.П.	Хімія	14.01.15

RASP

Назва_дисц	Дата
Хімія	14.01.15
Фізика	10.01.15

Сформуємо відповідь на такий запит: дати відомості про студентів, здала всі іспити. Для отримання відповіді на цей запит необхідно виконати наступну операцію ділення:

t = r ÷ p

STUD = VEDOM ÷ RASP

Номер_зал_кн	ПІБ
02-Е-01	Іваненко І.І.
02-Е-02	Сіренко П.П.

Реляційне числення

Більшість працюючих мов запитів засноване на реляційному обчисленні. Обчислення висловлюють тільки те, яким повинен бути результат обчислення, але не те, яким чином проводити обчислення. Ця обов'язок покладається на процесор мови запитів даної СУБД.

Розрізняють реляційне числення кортежів і реляційне обчислення доменів. Оскільки реляційне числення доменів схоже з реляційним обчисленням кортежів за винятком того, що змінні приймають значення в доменах, а не є кортежами, а обчислення кортежів використовується частіше, розглянемо тільки обчислення кортежів. Надалі, коли мова буде йти про реляційне обчислення, буде матися на увазі саме реляційне числення кортежів.

Реляційне числення кортежів є, по суті, формалізацією системи позначень, призначеної для освіти множин. В реляционном обчисленні використовуються булеві операції (І, АБО, НЕ) над умовами, які можуть бути істинними або помилковими. В ньому також використовуються квантори існування і загальності, означають, відповідно, що елемент певного типу існує або що умова істинно для кожного елемента певного типу.

Розглянемо відношення:

r – Деталь

Код_вир	Назва	Вага
01	А	1
02	Д	2
03	В	2
04	С	3

Запит: Які деталі мають вагу, рівну 2?

У реляційній алгебрі для виконання цього запиту необхідно використовувати вираз, яке повинно містити наступні операції:

- операцію ***Вибір*** над відношенням r, результатом її застосування до відношення r є інше відношення, яке представляє собою підмножина кортежів відносини r зі значенням, рівним 2 в атрибуті *Вага*:

Код_вир	Назва	Вага
02	Д	2
03	В	2

- проектування результату попередньої операції на атрибут *Назв_дет*. Остаточний результат запиту буде виглядати наступним чином:

Назва
Д
В

У реляційному ж численні формулювання цього запиту повинно мати наступний вигляд:

{T.Назв_дет | t in r and t.Вес = 2},

де t - це змінна, що позначає довільну рядок.

Відношення, з якого береться t, визначається виразом "in r" яке означає, що t- рядок відносини. t.Назв_дет - значення атрибута Назв_дет в рядку r; символ (|) - розділяє цільової список і визначає вираз. В даному випадку:

t.Назв_дет - цільової список;

t in r and t.Вес = 2 - визначальне вираз;

t.Вес = 2 означає, що значення атрибута Вага в рядку t дорівнює 2.

Фігурні дужки "{}", в які укладено вираз, визначають результат запиту, як безліч значень даних. Що саме входить в це безліч, описує вираз в дужках. наведене тут рішення ілюструє майже всі елементи реляційного числення.

Цільовий список і визначальний вираз

Рішенням кожного запиту в реляцфйному обчисленні є відношення, яке задається цільовим списком і визначальним виразом.

Цільовий список визначає атрибути відносення рішення.

Визначальнмй вираз - це умова, на підставі якої відбираються значення з бази даних, які увійдуть в відношення рішення. В попередньому прикладі Назв_дет береться з рядка і переміщується в рядок рішення, якщо рядок t задовольняє умові t in r and t. Вага = 2. Система переглядає рядки відносини r одну за одною. Першому рядку тимчасово присвоюється ім'я t і перевіряється істинність основного виразу. Оскільки в цьому випадку t.Вес = 1, що визначає вираз, помилково, тому А - значення атрибута t.Назв_дет в цей рядок не поміщається в відношення рішення. Потім система переходить до наступного рядка, дає їй ім'я t і знову перевіряє істинність визначального виразу. На цей раз вираз істинний, тому Д поміщається в відношення рішення. Процес повторюється для кожного рядка відносення r. В результаті виходить наступне відношення, ідентичне отриманому раніше:

Назва
Д
В

Квантор існування і квантор загальності

У розглянутих прикладах були використані операції вибору, перетині і проектування реляційної алгебри. аналогічно можна отримати конструкції реляційного числення, відповідні ряду інших операцій: об'єднання, різниці і добутку.

Дещо по-іншому виглядають аналоги тільки двох операцій реляційної алгебри - операцій з'єднання і ділення. для побудови аналогів цих операцій потрібні квантори: квантор існування для з'єднання і квантор загальності для поділу.

Квантор існування означає, що існує хоча б один екземпляр певного типу речей. У реляционном обчисленні квантор існування використовується для завдання умови того, що певний тип рядків щодо існує.

Квантор загальності означає, що деяка умова застосовується до всіх рядків або до кожного рядка деякого типу. Він використовується в тих же цілях, що і операція ділення реляційної алгебри.

МЕТОДИЧНІ ВКАЗІВКИ

із самостійної роботи студентів

1. Тема: Історія розвитку СКБД. Компоненти середовища СКБД

2. Навчальна мета. Студент повинен:

знати: історію розвитку баз даних, компоненти середовища СКБД і їх призначення

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Історія розвитку
2) Компоненти середовища БД

- Апаратне забезпечення
- Програмне забезпечення
- Дані
- Процедури
- Користувачі.

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Зміна підходу до обробки даних
2) Основні функції кожного компоненту БД

5.Контроль засвоєння теми

Різнорівневі питання

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)

2. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

3. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X

Історія розвитку баз даних

Перша система, що використовувала базу даних, з'явилась в середині 60-х років і була основана на ієрархічній моделі.

В кінці 60, на початку 70 з'явились мережеві СУБД.

І в ієрархічній і в мереженій системах БД для зв'язку файлів використовувались фізичні вказники. Ці вказники дозволяли вилучати дані, зв'язані відповідними відношеннями. Але недоліком таких систем було те, що ці відношення повинні бути визначені до запуску системи. Вилучити дані на основі інших відношень було неможливо.

В 1970 році Е.Ф.Кодд опублікував статтю в якій висунув ідею, що дані потрібно зв'язувати в відповідності з їх внутрішніми логічними взаємовідношеннями, а не фізичними вказниками. Таким чином користувачі зможуть комбінувати дані з різних джерел, якщо логічна інформація, що необхідна для такого комбінування, присутня в висхідних даних. Така модель даних дістала назву *реляційної* моделі. Для роботи з даними використовується реляційна алгебра і реляційне числення. Це забезпечує роботу з даними на основі логічних характеристик. В реляційних системах однією командою може оброблятись цілий файл, а в ієрархічних і мережених – тільки один запис.

Логічний підхід до даних зробив можливим створення мов запитів, що є більш доступними для користувачів, що не є спеціалістами.

В другій половині 70 років з'явились реляційні системи, які підтримували мову структурованих запитів (SQL), мову запитів (Quel), запити по зразку (QBE).

Сьогодні реляційні системи розглядаються як стандарт систем роботи з даними.

Реляційні бази даних продовжують вдосконалюватись, їх внутрішня природа змінюється. Це є використання об'єктно-орієнтованих баз даних (використання концептуальної моделі даних) і використання технології клієнт-серевер.

Компоненти середовища СУБД

У середовищі СУБД можна виділити наступних п'ять основних компонентів: апаратне і програмне забезпечення, дані, процедури і користувачів.

*Апаратне забезпечення***.** Це може бути персональний комп'ютер, мэйнфрейм або мережа з багатьох комп'ютерів. Використовуване апаратне забезпечення залежить від вимог даної організації і типу СУБД.

*Програмне забезпечення***.** Цей компонент охоплює програмне забезпечення самої СУБД і прикладних програм, операційну систему, мережне програмне забезпечення, якщо СУБД використовується в мережі. Звичайно додатки створюються на мовах третього покоління, таких як З, C++, Java, Visual Basic, COBOL, Fortran, Ada або Pascal, або на мовах четвертого покоління, таких як SQL. СУБД може мати свої власні інструменти четвертого покоління, призначені для швидкої розробки додатків з використанням убудованих непроцедурних мов запитів, генераторів звітів, форм, графічних зображень.

*Дані***.** Є самим важливим компонентом середовища СУБД (з погляду кінцевих користувачів). База даних містять як робітники дані, так і метадані, тобто "дані про дані".

Процедури. До процедур відносяться інструкції і правила, що повинні враховуватися при проектуванні і використанні бази даних. Користувачам і обслуговуючому персоналові бази даних необхідно надати документацію, що містить докладний опис процедур використання і супроводи даної системи:

- Реєстрація в СУБД.
- Використання окремого інструмента СУБД або додатка.
- Запуск і останов СУБД.
- Створення резервних копій СУБД.
- Обробка збоїв апаратного і програмного забезпечення, а також відновлення бази даних після усунення несправності.
- Зміна структури таблиці, реорганізація бази даних, розміщеної на декількох дисках, способи поліпшення продуктивності і методи архівування даних на вторинних пристроях збереження.

Користувачі. Серед них можна виділити чотири різні групи: адміністратори даних і адміністратори баз даних, прикладні програмісти і кінцеві користувачі баз даних.

- *Адміністратори даних і адміністратори баз даних*

Адміністратор даних, або АД (Data Administrator — DA), відповідає за керування даними, включаючи планування бази даних, розробку і супровід стандартів, прикладних алгоритмів і ділових процедур, а також за концептуальне і логічне проектування бази даних.

Адміністратор бази даних, або АБД (Database Administrator — DBA), відповідає за фізичну реалізацію бази даних, включаючи фізичне проектування і втілення проекту, за забезпечення безпеки і цілісності даних, за супровід операційної системи, а також за забезпечення максимальної продуктивності додатків і користувачів. У порівнянні з ПЕКЛО обов'язку АБД носять більш технічний характер, і для нього необхідне знання конкретної СУБД і системного оточення.

– *Розроблювачі баз даних*

У проектуванні великих баз даних беруть участь розроблювачі двох різних типів: розроблювачі логічної бази даних і розроблювачі фізичної бази даних. Розроблювач логічної бази даних займається ідентифікацією даних (тобто сутностей і їхніх атрибутів), зв'язків між даними, і встановлює обмеження, що накладаються на збережені дані. Робота розроблювача логічної бази даних поділяється на два етапи: Концептуальне проектування бази даних і Логічне проектування бази даних.

Розроблювач фізичної бази даних одержує готову логічну модель даних і займається її фізичною реалізацією, у тому числі:

- перетворенням логічної моделі даних у набір таблиць і обмежень цілісності даних;
- вибором конкретних структур збереження і методів доступу до даних, що забезпечують необхідний рівень продуктивності при роботі з базою даних;
- проектуванням будь-яких необхідних мір захисту даних.

– *Прикладні програмісти*

Після створення бази даних розробляють додатка, що надають користувачам необхідні їм функціональні можливості. Прикладні програмісти працюють на основі специфікацій, створених системними аналітиками.

– *Користувачі*

Користувачі є клієнтами бази даних — вона проектується, створюється і підтримується для того, щоб обслуговувати їхні інформаційні потреби. Користувачів можна класифікувати по способі використання ними системи.

– Рядові користувачі. Ці користувачі звичайно навіть і не підозрюють про наявність СУБД. Вони звертаються до бази даних за допомогою спеціальних додатків, що дозволяють у максимальному ступені спростити виконувані ними операції. Такі користувачі ініціюють виконання операцій бази даних, уводячи найпростіші команди або вибираючи команди меню

– Досвідчені користувачі. Це користувачі, що знайомі зі структурою бази даних і можливостями СУБД. Для виконання необхідних операцій вони можуть використовувати таку мову запитів високого рівня, як SQL. А деякі досвідчені користувачі можуть навіть створювати власні прикладні програми.

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Розподілені бази даних

2. Навчальна мета. Студент повинен:

знати: визначення, призначенні, принципи будови розподілених баз даних.

уміти:

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Визначення розподіленої бази даних
2) Архітектура розподіленої бази даних .
3) Системи розподілених баз даних

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Коли має сенс використання розподілених баз даних..
2) Відмінності розподілених і реляційних баз даних

5.Контроль засвоєння теми

Різнорівневі питання.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X
3. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

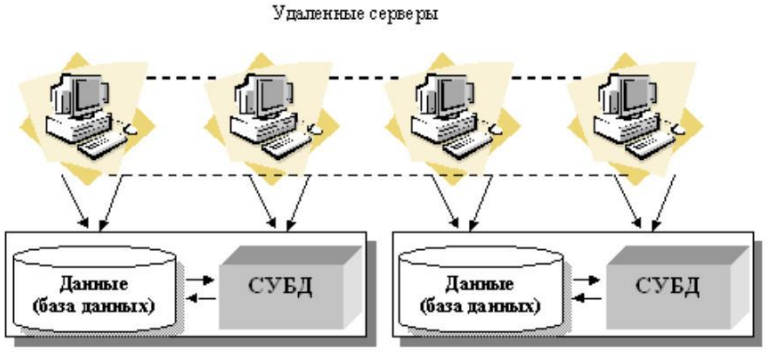
Розподілені бази даних

Термін "розподілена база даних" досить часто зустрічається в літературі. Однак в різних джерелах під цим терміном розуміються зовсім різні речі. Частина авторів розуміють під розподіленою базою даних то, що є віддалений сервер, на якому розташовані дані, а також клієнтські комп'ютери, розташовані територіально в іншому місці. Таке трактування нам представляється неправильної. Справжня розподілена база даних розташовується на кількох комп'ютерах. При цьому частина файлів розташована на одному комп'ютері, частина на іншому і т.д. Більш того, можлива і навіть часто зустрічається ситуація, коли інформація на цих комп'ютерах перетинається, дублюється.

Розподілена база даних - сукупність логічно взаємопов'язаних поділюваних даних (і описів їх структур), фізично розподілених у комп'ютерній мережі.

Система управління розподіленою базою даних - програмна система, що забезпечує роботу з розподіленою базою даних і дозволяє користувачеві працювати як з його локальними даними, так і з усією базою даних в цілому.

Система управління розподіленою базою даних (РаСУБД) є розподіленою системою. кожен фрагмент бази даних працює під управлінням окремої СУБД, яка здійснює доступ до даних цього фрагмента. Користувачі взаємодіють з розподіленою базою даних через локальні і глобальні програми. Локальні додатки дають користувачеві можливість працювати зі своїми локальними даними і не вимагають доступу до інших фрагментів. Глобальні додатки дають користувачеві можливість працювати з іншими фрагментами бази даних, розташованими на інших комп'ютерах мережі. Загальна схема розподіленої бази даних представлена на рис. .



Об'єднання даних організовується віртуально. Відповідний підхід, по суті, відображає організаційну структуру підприємства (і навіть суспільства в цілому), що складається з окремих підрозділів. Причому, хоча кожен підрозділ обробляє свій набір даних (ці набори, як правило, перетинаються), існує необхідність доступу до цих даних як до єдиного цілого (зокрема, для управління всім підприємством).

Одним із прикладів реалізації такої моделі може служити мережа Інтернет: дані вводяться і зберігаються на різних комп'ютерах по всьому світу, будь-який користувач може отримати доступ до цих даних, не замислюючись про те, де вони фізично розташовані.

Архітектура розподіленої бази даних

Розподілена база даних (distributed database) - це група баз даних, яка виглядає для користувачів і додатків як одна база даних. У більшості випадків бази даних, що становлять розподілену базу даних, розташовані на окремих комп'ютерах, взаємодіючих по мережі. Після того як система розподіленої бази даних Oracle налаштована, всі дані в цій системі стають доступними додатків, як якщо б вони перебували в одній логічної базі даних. Наприклад, транзакція, подібна до тієї, яка зображена на рис.1, може включати до свого складу оператори DML, за допомогою яких модифікується інформація кількох баз даних.

Взаємодія і автономність

Кожен сервер бази даних в системі розподіленої бази даних управляє доступом до своєї локальної базі даних - за керування системою в цілому не відповідає жоден сервер. Однак всі сервери системи повинні взаємодіяти один з одним, щоб забезпечити узгодженість і точність даних, у всій системі.

Розширення моделі клієнт / сервер

Системи розподілених баз даних Oracle - це просто розширення базової моделі клієнт / сервер, так як сервер бази даних в розподіленої системі може виступати як в ролі клієнта, так і в ролі сервера, в залежності від виду операції, яку вони виконують в транзакції.

Мережі та системи розподілених баз даних

Більшість систем розподілених баз даних складається з декількох баз даних, керованих різними серверами, розташованими в різних місцях. При цьому для встановлення ліній зв'язку між серверами потрібно мережу. Крім того, всі сервери (і клієнти) в системі розподіленої бази даних Oracle повинні використовувати Net8 - мережеве програмне засіб Oracle для взаємодії один з одним по мережі.

Сервіси бази даних і іменування в розподіленої базі даних

Всі сервіси (черги друку, сервери електронної пошти і т.д.), доступні в мережі, повинні мати унікальні імена, щоб користувачі і додатки знали, як до них звертатися. У розподіленої системі сервер бази даних, або екземпляр є просто сервісом бази даних(database service), доступним в мережі. Сервіс бази даних в системі розподіленої бази даних повинен мати унікальне ім'я, щоб додатки та інші сервери баз даних могли звертатися до інформації, за яку відповідає цей сервер.

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Технологія CORBA

2. Навчальна мета. Студент повинен:

знати: принципи технології

уміти:

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Визначення технології
2) Архітектура CORBA
3) Внутрішній алгоритм передачі запитів

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Специфіка архітектури CORBA..

5.Контроль засвоєння теми

Письмова робота.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)

Технологія CORBA

Специфікація CORBA (Common Object Request Broker Architecture) розроблена групою компаній OMG (Object Management Group) і описує заснований на об'єктах спосіб створення розподілених додатків, тобто як здійснюється взаємодія об'єктів клієнта і сервера. Після інсталяції спеціального програмного забезпечення сервер додатків і клієнти можуть взаємодіяти, використовуючи об'єкти і інтерфейси CORBA.

Програмне ядро CORBA розроблено для всіх основних апаратних і програмних платформ. Об'єкти додатків CORBA, використовуючи в якості посередника ПО CORBA, взаємодіють з іншими об'єктами CORBA, використовуючи для цього інтерфейси. Для пошуку і організації взаємодії об'єктів CORBA призначене спеціальне ПЗ - об'єктний брокер запитів (Object Request Broker - ORB) і мережевий агент Smart Agent. В результаті клієнтські і серверні додатки CORBA, що працюють на платформі Wintel, однаково легко взаємодіють з аналогічними додатками CORBA, відкомпілювалися для операційних систем UNIX або AS400.

Таким чином, використання архітектури CORBA дозволяє розробникам багаторівневих додатків в Delphi створювати дійсно гетерогенні системи, використовуючи переваги кожної платформи.

Архітектура CORBA

Клієнтські програми та програми-сервери інкапсулюють об'єкти CORBA. Специфікація CORBA визначає, яким чином повинні взаємодіяти об'єкти CORBA в розподіленої гетерогенної середовищі.

Архітектура CORBA являє собою сукупність програмних засобів, які забезпечують передачу об'єктних запитів в рамках розподіленої гетерогенного середовища відповідно до специфікації CORBA. Архітектура включає наступні елементи.

Брокер запитів VisiBroker for C ++ ORB (Version 3.3.2) - головна складова частина архітектури і призначений для обробки запитів і організації взаємодії між об'єктами клієнтів і серверів.

Служба Implementation Repository зберігає інформацію про зареєстровані об'єкти, їх ідентифікатори і т. Д. Крім цього, в ньому зберігаються додаткові відомості про розподіленої середовищі: місце розташування ресурсів, налагоджувальна інформація, рівень безпеки і т. Д. VisiBro ker використовує цю службу для звернення до об'єктів .

Basic Object Adaptor забезпечує активізацію об'єктів сервера при їх виклик клієнтом. Це ПО реєструє об'єкт в мережевий розподіленої службі Smart Agent.

Служба Smart Agent забезпечує взаємодію частин архітектури в мережевому середовищі. Для роботи програми CORBA досить одного запущеного екземпляра цієї служби на будь-якому комп'ютері середовища. Він об'єднує між собою окремі брокери запитів.

Object Activation Daemon виконує автоматичний запуск сервера при першому запиті від клієнта.

Взаємодія служб CORBA

ORB повинен бути встановлений і функціонувати на кожному комп'ютері, що містить додатки CORBA. Він забезпечує управління запитами від клієнта серверу.

Клієнтську програму, використовуючи внутрішній механізм виконання запитів (див. Нижче), звертається до примірника VisiBroker, що функціонує на комп'ютері клієнта.

Після отримання запиту об'єктний брокер запиту опитує мережеве оточення з метою знайти працюючий екземпляр служби Smart Agent. Для цього ORB розсилає по мережі повідомлення. Робочий сеанс встановлюється з тієї службою Smart Agent, яка відгукнеться першої. З'єднання брокера запитів з мережевою службою працює на основі протоколу UDP. Для цього їм потрібно мати однаково налаштовані номера мережевих портів.

Служба Smart Agent має доступ до бази даних служби Implementation Repository, в якій зареєстровані всі працюючі сервери, запущені вручну, і всі сервери, що запускаються автоматично за допомогою Object Activation Daemon. Відповідно до отриманої інформації. Smart Agent передає запит потрібного екземпляру брокера.

знайдений ORB забезпечує передачу запиту серверу. Результат запиту повертається клієнтові аналогічним чином.

Реєстрація сервера CORBA

Сервер може запускатися вручну або автоматично за допомогою OAD. При ручному запуску сервер звертається до служби BOA, що функціонує на комп'ютері сервера. BOA передає інформацію про місцезнаходження і параметрах сервера для зберігання службі Implementation Repository. А до Implementation Repository звертається Smart Agent при адресації клієнтського запиту.

Реєстрація сервера для ручного запуску здійснюється з середовища Delphi або за допомогою утиліти idl2ir.

Для автоматичного запуску сервер повинен бути зареєстрований за допомогою утиліти oadutil. При автоматичному запуску служба Smart Agent звертається до OAD, який і запускає додаток- сервер.

Внутрішній механізм передачі запитів

У сервері додатків взаємодія з клієнтами забезпечує спеціальний об'єкт, званий скелетом (skeleton). Скелет працює в адресному просторі сервера.Він отримує запити від клієнта, перетворює їх і передає серверу. Відповіді сервера він форматує і при посередництві розглянутих вище служб відправляє клієнту, а точніше, заглушці.

Заглушка (stub) - особливий об'єкт CORBA, який функціонує в адресному просторі клієнта. Цей об'єкт перехоплює запити клієнта, перетворює їх в транспортний формат і відправляє серверу за допомогою служб CORBA, а точніше, скелету. Він же отримує відповіді сервера.

Розглянемо ситуацію, коли клієнт відправляє запит серверу.

Клієнт передає параметри виклику в стек і здійснює виклик відповідної функції через покажчик інтерфейсу. Якщо виклик спрямовується до іншого адресний простір, то він перехоплюється заглушкою, упаковується (параметри виклику зчитуються з стека), поміщається в спеціальний буфер обміну і через служби передається скелету сервера.

Скелет звертається до буферу, отримує з нього і розпаковує пакет, поміщає параметри виклику в стек і передає виклик сервера. Тим самим скелет реконструює виклик клієнта в адресному просторі сервера.

В результаті, сервер і клієнт як би працюють в одному адресному просторі. Причому клієнта "здається", що сервер працює в його процесі, а сервер "перебуває у впевненості", що клієнт функціонує в його адресному просторі.

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Порівняння архітектур COM І CORBA

2. Навчальна мета. Студент повинен:

знати: характеристики і порівняння COM і CORBA

уміти:

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Характеристики COM і CORBA
2) Відмінності COM і CORBA.

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) В яких випадках використовуються технології COM і CORBA

5.Контроль засвоєння теми

Письмова робота.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X
3. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

Порівняння архітектур COM I CORBA

Під розподіленими системами зазвичай розуміють програмні комплекси, складові частини яких функціонують на різних комп'ютерах в мережі. Ці частини взаємодіють один з одним, використовуючи ту чи іншу технологію різного рівня - від безпосереднього використання сокетів TCP / IP до технологій з високим рівнем абстракції, таких, як RMI або CORBA.

Зростання популярності розподілених систем викликаний істотним підвищенням вимог, що пред'являються замовником до сучасних програмних продуктів. Мабуть, найважливішими з цих вимог є такі:

- Забезпечення масштабованості систем, тобто здатності ефективно обслуговувати як мале, так і дуже велика кількість клієнтів одночасно.
- Надійність створюваних додатків. Програмний комплекс повинен бути стійкий не тільки до помилок користувачів (це визначається головним чином якістю клієнтських додатків), але і до збоїв в системі комунікацій. Надійність має на увазі використання транзакцій, тобто гарантованого переходу системи в процесі функціонування з одного стійкого і достовірного стану в інше.
- Можливість безперервної роботи протягом тривалого часу (так званий режим 24x7, тобто режим цілодобової роботи протягом тижнів і місяців).
- Високий рівень безпеки системи, під якою розуміється не тільки контроль доступності тих чи інших ресурсів системи і захищеність інформації на всіх етапах функціонування, але і відстеження виконуваних дій з високим ступенем достовірності.
- Висока швидкість розробки додатків і простота їх супроводу і модифікації з використанням програмістів середньої кваліфікації.

Виявилося, що забезпечити відповідність цим вимогам, використовуючи традиційні технології - а саме, Дволанковий системи "клієнт-сервер", в яких в якості серверів виступають системи управління базами даних, майже неможливо.

Зауважимо, що далеко не для всіх додатків вищеперелічені вимоги є суттєвими. Легко уявити розподілену систему, яка функціонує на невеликому (до 100) кількість комп'ютерів, що працюють в локальній мережі, де немає проблем з перезапуском одного або двох-трьох серверів в разі необхідності, а головними завданнями, для вирішення яких використовуються розподілені технології, є завдання використання функціональності декількох стандартних серверів додатків (наприклад, текстових процесорів, браузерів і електронних таблиць) в інтересах клієнтських завдань. Необхідно мати на увазі, що термін "розподілені системи" відноситься до величезного числа завдань самого різного класу.

Даний огляд містить порівняльний аналіз двох найбільш популярних і комплексних систем створення розподілених додатків, а саме, CORBA консорціуму OMG і COM (DCOM, COM +) фірми Microsoft. Цей огляд призначений головним чином для менеджерів проектів, керівників інформаційних служб та ін., Тобто для тих, кому доводиться приймати відповідальні, "стратегічні" рішення.Внаслідок цього, в ньому будуть відсутні чисто технічні аспекти обох технологій, які цікаві тільки для розробників.

Крім того, при написанні огляду не ставилося завдання зробити остаточний висновок типу "... таким чином, CORBA (COM), безперечно, краще, ніж COM (CORBA)". Пов'язано це не з модною в наш час "політкоректністю" або відсутністю у автора своєї точки зору з цього питання. Справа навіть не в тому, що існують певні труднощі з формалізацією такого порівняння - я міг би представити результати порівняльних аналізів, в яких використовуються чисельні оцінки (бали), виставлені експертами, вагові коефіцієнти та інше, що надає звіту серйозний і вагомий вид. Справа в тому, що COM і CORBA можна порівнювати тільки з урахуванням певних припущень. Відмова від таких припущень легко дозволяє отримати будь-якій результат. Ось ці припущення:

- CORBA, на відміну від COM'а, є концепцією, а не її реалізацією. Коли ми говоримо "COM", то розуміємо під цим швидше набір конкретних засобів - елементів операційної системи, бібліотек, утиліт і т.п., які є складовою частиною того, що називається Microsoft Windows. Під терміном "CORBA" розуміється саме складна і розвинена концепція, сформульована на рівні спеціальної мови описів - IDL. Реалізації ж цієї концепції можуть сильно відрізнятися один від одного за різними критеріями, найбільш важливим в тому чи іншому випадку. Inprise / Corel VisiBroker і Application Server, BEA WebLogic, Iona Orbix, Oracle Application Server і "картриджі" Oracle, IBM BOSS - всі ці продукти використовують ті чи інші можливості CORBA. Технологія Sun Enterpise JavaBeans створена поверх CORBA і використовує такі її можливості, як віддалений виклик об'єктів, службу імен, управління

транзакціями. Реалізація EJB фірми Inprise пов'язано з CORBA ще більш тісно. Ми будемо порівнювати COM і CORBA саме як концепції створення розподілених систем, а не як ту чи іншу їх реалізацію.

- При роботі з реальним проектом необхідно враховувати область застосування тієї чи іншої технології. COM (як цілісне і комплексне рішення) здатний функціонувати тільки під Windows NT / Windows2000. Міркування про перенесення COM на інші операційні системи зараз носять скоріше рекламний характер. Ні компонентна модель COM (тобто ActiveX), ні монітор транзакцій (Microsoft Transaction Server, MTS) не здатні працювати ніде, крім Windows, і сама Microsoft не проявляє ніякої активності в цьому напрвлении (питаннями перенесення тих чи інших елементів COM на інші платформи займається консорціум Open Group). CORBA є набір інструментів, написаний рішенням просто за визначенням.
- Крім чисто технологічних аспектів, при ухваленні рішення необхідно оцінити витрати на закупівлю необхідного програмного забезпечення, його супроводу і навчання персоналу. COM (на відміну від CORBA) офіційно є безкоштовною технологією. Ви отримуєте все необхідне, просто придбавши, наприклад, Windows NT Server. (До речі, сам факт конкуренції дорогої технології з "аналогічної", але безкоштовної, багато що говорить про їх технічні можливості).
- Наявність готових серверів додатків, придатних для вирішення вашого завдання. Якщо Ви можете вирішити свої проблеми, використовуючи функціональність Microsoft Office, то нічого кращого COM ви, природно, не знайдете.

Таким чином, головним завданням цього огляду є спроба допомогти керівнику того чи іншого рівня прийняти кваліфіковане рішення в кожному конкретному випадку. Оскільки і CORBA, і COM позиціонуються відповідно OMG і Microsoft як універсальні технології створення розподілених систем, ми будемо оцінювати і порівнювати їх саме з цієї точки зору. Передбачається, що для проекту використовується платформа Windows (в іншому випадку нічого розглядати COM) і є достатньо коштів для закупівлі основних стандартних частин тієї чи іншої реалізації (інакше обговорення CORBA втрачає сенс).

Концептуальний фундамент технології

COM

Технологія створювалася фірмою Microsoft як засіб взаємодії додатків (у тому числі складових частин операційної системи) Windows, що функціонують на одному комп'ютері, з подальшим розвитком для використання в межах локальної мережі. Головне завдання на момент створення - забезпечення технології Object Linking and Embedding (OLE 1.0). Характерно, що обмін даними між додатками (Dynamic Data Exchange, DDE) спочатку будувався не за COM-технології, а з використанням механізму повідомлень (messages). Розвиток технології йде в міру додавання нових можливостей. Як універсальна технологія взаємодії додатків COM почав використовуватися з OLE 2.0 (1991). Концепція технології нерозривно пов'язана з її реалізацією. Поява нових можливостей - це просто поява нових бібліотек, функцій API і утиліт Windows. "Спільний знаменник технології" - двійкова структура об'єкта, хоча зараз існує мова опису структури об'єкта - Interface definition Language (IDL).

CORBA

Технологія створювалася консорціумом OMG як універсальна технологія створення розподілених систем в гетерогенних середовищах. OMG представляє собою некомерційну організацію, що є співтовариством розробників програмного забезпечення і його споживачів, що об'єднали свої зусилля для створення специфікацій цієї технології. На даний момент в OMG складається більш 800 членів, включаючи всіх скільки-небудь серйозних виробників програмного забезпечення (і навіть с недавнього часу Microsoft). Перша специфікація CORBA з'явилася в 1991 р Нові можливості офіційно вважаються доданими до CORBA в момент затвердження відповідної специфікації. Як правило, в розробці специфікації беруть участь найбільші фахівці в даній області. Розробка реалізації - завдання конкретної фірми. Зазвичай від затвердження специфікації до появи високоякісної реалізації проходить досить багато часу - іноді кілька років. "Спільний знаменник" технології - оголошення на мові IDL, який є "серцем" CORBA з моменту її появи. (Існують три різних мови описів з одним і тим же назвою - OSF IDL, Microsoft IDL і OMG IDL).

висновки

Технологія CORBA носить істотно більш загальний і універсальний характер, ніж COM, що закладено в її фундаменті. Випередження розробки специфікацій (в порівнянні з реалізаціями) дозволяє домогтися більш зв'язковою, цілісної і гармонійної системи. З іншого боку, при розробці реального проекту потрібно попередньо переконатися, що високоякісна реалізація того чи іншого сервісу CORBA вже доступна (джерелами проблем можуть служити, наприклад, Persistence Service і Security Service).

комплексність системи

COM

COM містить все необхідне, що потрібно для побудови розподіленої системи: технологію віддаленого виклику методів (як статичних, так і динамічних), бази даних серверних об'єктів (бібліотеки типів), які можуть бути імпортовані для аналізу структури серверів COM, універсальний протокол обміну між клієнтами і серверами, специфікації так званих "складових документів" (ActiveDoc), об'єктний монітор транзакцій (MTS), компонентну модель (ActiveX) і ін. Всі складові частини прекрасно відповідають один одному в рамках моделі COM. Унікальною можливістю COM є універсальна технологія доступу до баз даних - OLE DB / ADO.

CORBA

На даний момент CORBA не має своєї власної компонентної моделі; робота над нею почалася в 1998 р і ще не завершена. Це головний серйозний недолік. Правда, він дещо компенсується наявністю заснованої на CORBA компонентної моделлю Enterprise JavaBeans, так що програмісти на Java знаходяться в привілейованому становищі. Все інше, що є присутнім в COM, є і в CORBA, і навіть більше того - за винятком універсальної технології доступу до БД. Знову-таки, Java-програмісти мають перевагу і тут - за рахунок наявності загальної для Java технології доступу до даних JDBC.

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Об'єктно-орієнтовані бази даних

2. Навчальна мета. Студент повинен:

знати: принципи побудови об’єктно-орієнтованих баз даних

уміти:

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Сучасний стан досліджень у галузі об'єктно-орієнтованих баз даних
2) Характеристика
3) Об’єктно-орієнтовані моделі даних
4) Об'єктно-орієнтована модель ODMG

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Реалізація властивостей об’єктно-орієнтованого підходу в об’єктно-орієнтованих базах даних

5.Контроль засвоєння теми

Різнорівневі питання.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)

2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X

3. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

Об'єктно-орієнтовані бази даних

Сучасний стан досліджень у галузі об'єктно-орієнтованих баз даних

Дослідження у сфері об'єктно-орієнтованих баз даних (ООБД) були розпочаті у зв'язку з необхідністю розробити ефективний механізм, що дозволяє об'єктно-орієнтованим застосуванням зберігати об'єкти після завершення своєї роботи. У будь-якій СКБД користувачами можуть бути не лише люди, а й програми, відтак програми можуть звертатися до СКБД як для збереження даних, так і для їхнього подальшого вибирання. Майже всі сучасні мови програмування є об'єктно-орієнтованими, а СКБД — реляційними. У зв'язку з цим нагальною стає потреба у створенні об'єктно-орієнтованих СКБД або розробці механізмів відображення між двома моделями даних: об'єктно-орієнтованою та реляційною. Інакше кажучи, необхідно надати програмістам, які працюють в об'єктно-орієнтованому середовищі, прозорий механізм збереження і вибирання даних із баз даних.

Об'єктно-орієнтована СКБД (ООСКБД) — це СКБД, що базується на об'єктно-орієнтованій моделі даних.

Нині ООСКБД приділяється багато уваги як з теоретичної, так і з практичної точок зору. Можна виділити три характерні риси сучасного стану досліджень у цій галузі:

відсутність загальноприйнятої моделі даних;

відсутність єдиної формальної теорії;

активна експериментаторська діяльність.

Класичні роботи Кодда специфікували системи реляційних баз даних (модель даних і мову запитів), для систем об'єктно-орієнтованих баз даних подібна специфікація відсутня. Це не означає, що не існує жодної повної моделі об'єктно-орієнтованої бази даних. Таких моделей є багато, але жодна з них не може вважатися загальноприйнятим стандартом. Нині нема єдиної думки про те, чим є об'єктно-орієнтована система програмування, не кажучи вже про ООСКБД. Окрім того, відсутній надійний теоретичний базис, що робить практично неможливим досягнення спільного погляду на модель даних.

Ведуться численні експериментальні дослідження. Можна сподіватися, що разом із відповідною моделлю виникнуть її життєздатні реалізації. Можливо, незабаром за взірець буде взята певна система, але не тому, що вона буде найоптимальнішою, а тому, що вона виявиться першою системою, яка надасть багато функціональних можливостей, відповідних до вимог ринку.

Особливе місце в розвитку проблематики об'єктно-орієнтованих СКБД посідає діяльність групи з управління об'єктними базами даних ODMG (Object Data Management Group) — некомерційного консорціуму виробників об'єктних баз даних та інших організацій, зацікавлених у виробленні стандартів зі зберігання об'єктів у базах даних.

ODMG діяла з 1991 по 2001 рік. У 1993 році група випустила свій перший стандарт — ODMG-93. 1995 року був опублікований удосконалений варіант цього стандарту. Стандарти ODMG — це стандарти зі зберігання об'єктів в базах даних. Ці стандарти стосуються трьох галузей: баз даних (мова SQL), об'єктної моделі даних (стандарти OMG - Object Management Group - консорціуму, що розробляє стандарти міжкомпонентної взаємодії) та об'єктно-орієнтованих мов програмування (стандарти ANSI з мов програмування C++, Smalltalk і Java).

Складовими частинами стандартів ODMG є:

об'єктно-орієнтована модель (Object Oriented Model — OOM);

мова опису об'єктів (Object Definition Language — ODL);

об'єктна мова запитів (Object Query Language - OQL);

описи методів зв'язування OOM з об'єктно-орієнтованими мовами програмування C++, SmallTalk, Java (Language Bindings to C++, SmallTalk, Java).

Характеристика

Об’єктно-орієнтована база даних (ООБД) дозволяє програмістам, які працюють із мовами третього покоління, інтерпретувати всі свої інформаційні сутності як об'єкти, що зберігаються в оперативній пам'яті. Додатковий інтерфейсний рівень абстракції забезпечує перехоплення запитів, що звертаються до тих частин бази даних, які перебувають у постійному сховищі на диску. Зміни, внесені в об'єкти, оптимальним образом переносяться з пам'яті на диск.

Перевагою об’єктно-орієнтованих баз даних є спрощений код. Додатки одержують можливість інтерпретувати дані в контексті тієї мови програмування, на якому вони написані. Реляційна база даних повертає значення всіх полів у текстовому виді, а потім вони приводяться до локальних типів даних. В

об’єктно-орієнтованих базах даних цей етап ліквідований. Методи маніпулювання даними завжди залишаються однаковими незалежно від того, перебувають дані на диску або в пам'яті.

Дані в об’єктно-орієнтованих базах даних здатні прийняти вид будь-якої структури, яку можна виразити використовуваною мовою програмування. Відносини між сутностями також можуть бути довільно складними. Об’єктно-орієнтована база даних управляє кеш-буфером об'єктів, переміщаючи об'єкти між буфером і дисковим сховищем у міру необхідності.

За допомогою об’єктно-орієнтованих баз даних вирішуються дві проблеми. По-перше, складні інформаційні структури виражаються в них краще, ніж у реляційних базах даних, а по-друге, усувається необхідність транслювати дані з того формату, що підтримується в СКБД. Наприклад, у реляційній СКБД розмірність цілих чисел може становити 11 цифр, а у використовуваній мові програмування - 16. Програмістові прийде враховувати цю ситуацію.

Об’єктно-орієнтовані СКБД виконують багато додаткових функцій. Це окупається сповна, якщо відносини між даними дуже складні. У такому випадку продуктивність об’єктно-орієнтованих баз даних виявляється вище, ніж у реляційних СКБД. Якщо ж дані менш складні, додаткові функції виявляються надлишковими. В об'єктній моделі даних підтримуються нерегламентовані запити, але мовою їхнього складання не обов'язково є SQL. Логічне подання даних може не відповідати реляційній моделі, тому застосування мови SQL стане безглуздим. Найчастіше зручніше обробляти об'єкти в пам'яті, виконуючи відповідні види пошуку.

Великим недоліком об’єктно-орієнтованих баз даних є їхні тісні зв'язки із застосовуваною мовою програмування. До даних, що зберігаються в реляційній СКБД, можуть звертатися будь-які додатки, тоді як, приміром, Java-об'єкт, поміщений в об’єктно-орієнтовану базу даних, буде становити інтерес лише для додатків, написаних на Java.

Об’єктно-орієнтовані моделі даних

Першою формалізованої і загальновизнаною моделлю даних була реляційна модель Кодда. У цій моделі, як і у всіх наступних, виділялися три аспекти – структурний, цілісний і маніпуляційний. Структури даних в реляційній моделі грунтуються на плоских нормалізованих відносинах, обмеження цілісності виражаються за допомогою засобів логіки першого порядку і, нарешті, маніпулювання даними здійснюється на основі реляційної алгебри або равносильного їй реляційного числення. Як відзначають багато дослідників, своїм успіхом реляційна модель даних багато в чому зобов’язана тому, що спиралася на строгий математичний апарат теорії множин, відносин і логіки першого порядку. Розробники будь-якої конкретної реляційної системи вважали своїм обов’язком показати відповідність своєї конкретної моделі даних загальної реляційної моделі, яка виступала в якості міри “реляційної” системи.

Основні труднощі об’єктно-орієнтованого моделювання даних є наслідком того, що такого розвиненого математічекого апарату, на який могла б спиратися загальна об’єктно-орієнтована модель даних, не існує. У великій мірі тому до цих пір немає базової об’єктно-орієнтованої моделі. З іншого боку, стверджується, що загальна об’єктно-орієнтована модель даних в класичному сенсі і не може бути визначена через непридатність класичного поняття моделі даних до парадигми об’єктної орієнтованості.

Слідуючи практиці багатьох ООБД, пропонується виділити два рівні моделювання об’єктів: нижній (структурний) і верхній (поведінковий). На структурному рівні підтримуються складні об’єкти, їх ідентифікація та різновиди зв’язку “isa”. База даних – це набір елементів даних, пов’язаних відносинами “входить в клас” або “є атрибутом”. Таким чином, БД може розглядатися як орієнтований граф. Важливим моментом є підтримання поряд з поняттям об’єкта поняття значення (пізніше ми побачимо, як багато на цьому побудовано в одній з успішних об’єктно-орієнтованих СУБД .

Об'єктно-орієнтована модель ODMG

В об'єктно-орієнтованій моделі дані та методи, що їх обробляють, об'єднуються в структури, які називаються об'єктами. Типи об'єктів називаються класами.

З точки зору баз даних є такі важливі особливості ООМ:

- Підтримка структур даних, що мають довільний рівень складності;
- Ідентифікованість та унікальність об'єктів;
- Належність об'єктів класам; 4 інкапсуляція;
- успадкування та ієрархії класів;
- поліморфізм.

Складні структури даних

Наявність складноструктурованих даних не є відмінною рисою ООМ, проте існування ООМ без механізмів породження структур даних довільної складності важ- ко уявити. Складні об'єкти будуються з простіших за допомогою конструкторів. Найпростішими об'єктами єі числа, символи, символьні рядки довільної довжини, булеві змінні тощо. Існують різні конструктори складних об'єктів (кортежів, множин, мультимножин, списків та масивів). Мінімальний набір конструкторів, який повинна мати система, - це конструктори множин, списків і кортежів. Множини необхідні, оскільки завдяки їм набори об'єктів реального світу зображуються в природний сиосіб_ Кортежі надають спосіб зображення властивостей сутностей. Списки та масиви потрібні для відображення впорядкованості об'єктів реального світу, а також для зображення широкого кола математичних об'єктів.

Будь-який конструктор має бути застосовним до будь-якого об'єкту (наприклад, повинна надаватися можливість побудови множини з масивів або масиву з множин). Конструктори реляційно'і моделі не мають такої властивості, оскільки конструкція множини може бути застосована лише до кортежів, а конструкція кортежу - лише до атомарних значень. Навіть реляц:йна модель, що підтримує ненормалізовані відношення (тобто відношення, які не перебувають у першій нормальній формі), не має вказаної властивості, оскільки конструкцією верхнього рівня завжди має бути відношення.

Маніпулювання складними об'єктами забезпечується відповідними операціями, які часто розповсюджуються на всі компоненти таких об'єктів. Прикладом може бути вибирання чи видалення складного об'єкту або створення його копії. Існує можливість визначати додаткові операції над складними об'єктами.

Ідентифікованість, унікальність і стан об'єктів

Кожний об'єкт є унікальним, тобто забезпечується унікальна ідентифікація об'єктів (для мов проірамуванкя унікальними ідентифікаторами можуть бути адреси пам'яті, за якими зберігаються об'єкти),

Стан об'єкта — це поточне значення, приписане об'єкту. Об'єкт може мати еди- ний стан протягом свого життєвого циклу або переходити з одного стану в інший. Оскільки об'єкти мають властивість інкапсуляції (що буде розглянута нижче), то стан об'єкта є абстракцією, яка визначається лише через його поведінку (методи).

Унікальність об'єкта не залежить від його стану. Два об'єкти, що перебувають в одному й тому ж стані, є рівними, але не ідентичними. Не може існувати двох або більше екземплярів одного об'єкта (двох копій одного й того самого об'єкта з одним і тим самим ідентифікатором) У моделі з ідентифікованістю об'єктів об'єкт існує незалежно від свого значення. Отже, є два поняття еквівалентності об'єктів: об'єкти можуть бути ідентичними (бути Одним і тим самим об'єктом) або вони можуть бути рівними (перебувати в одному й тому самому стані). У цьому контексті слід розглянути такі два аспекти: розрізнення та змінення об'єктів.

Розрізнення об'єктів

У моделі з об'єктами, що ідентифікуються, два об'єкти можуть спільно використовувати компоненти (зокрема інші об'єкти). Отже, схематичним відображенням складного об'єкта є граф, натомість у системі без ідентифікованості об'єктів — це дерево. Розглянемо такий приклад: людина має ім'я, вік і дітей. Припустимо, що Іван і Марія мають сина на ім'я Петро. У реальному житті можуть мати місце дві ситуації: Іван і Марія є батьками однієї і тієї самої дитини або кожний із них має но сину. У моделі з ідентифікованістю об'єктів адекватно відображуються обидві ситуації.

Змінення об'єктів

Припустимо, що Іван і Марія є батьками хлопчика на ім'я Петро. Тоді зміни, що стосуються сина Марії, будуть виконуватися з об'єктом Петро, відтак і з сином Івана.

Поведінка об'єктів

Поведінка об'єкта це сукупність операцій (методів), які він надає. Лише через ці операції розкривається семантика об'єкта. Виконання операцій є єдиним способом взаємодії між об'єктами. Всі можливі операції об'єкта утворюють його інтерфейс Лише використовуючи операції, можна змінити стан об'єкта.

Класи об'єктів

В об'єктно-орієнтованій моделі клас узагальнює спільні риси об'єктів, що мають однакові властивості, й відповідає поняттю абстрактного типу даних. Клас означує спосіб реалізації множини

об'єктів, встановлюючи їхню структуру, поведінку та інтерфейс, тобто спосіб запам'ятовування інформації про їхні стани. Проте власне стан має запам'ятовувати сам об'єкт.

Клас є водночас фабрикою та сховищем об'єктів. Як фабрика об'єктів клас використовується для створення нових об'єктів. Термін сховище об'єктів означає, що до класу приєднується набір об'єктів, які є його екземплярами. Отже, класи використовуються для створення об'єктів і маніпулювання ними.

Однією з основних властивостей класу, відтак і його об'єктів, є інкапсуляція.

Інкапсуляція

Інкапсуляція вимагає, щоб дані та програмні коди для маніпулювання даними були приховані. З цієї точки зору об'єкт поділяється на інтерфейсну й реалізаційку частини. Інтерфейсна частина є специфікацією набору операцій, допустимих над об'єктом. Лише ця частина об'єкта видима для методів інших об'єктів. Реалізацій- па частина складається з даних, що описують стан об'єкта, і процедур, що реалізують операції над об'єктом. Інкапсуляція специфікується на рівні оголошення класу.

Успадкування

Успадкування є механізмом, що дає змогу створювати нові класи з використанням даних і методів інших класів. Це дає можливість деякі властивості, спільні для багатьох класів, описувати в базовому класі. Наприклад, якщо необхідно додати новий клас Одяг, що повністю збігається з класом

Поліморфізм

Принцип поліморфізму є розширенням принципу успадкування й дає змогу пере- означувати методи в успадкованих класах. Наприклад, є кілька різновидів товарів, для яких властиві специфічні правила обчислення ціни. Зокрема товарами можуть бути продукти, на які не діють націнки, та одяг, на який націнка встановлена. Всі товари успадковують метод Сумарна_ціна() з базового класу Товар, але правила обчислення ціни можуть бути різними. Це означає, що кожний клас, який є нащадком класу Товар, може мати власну реалізацію методу сумарнацінао. Тому вираз х.Сумарна_ціна() може означати різні правила обчислення ціни залежно від того, екземпляром якого класу є об'єкт х.

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Загальна характеристиа автоматизованих інформаційних систем

2. Навчальна мета. Студент повинен:

знати: визначення і типи автоматизованих систем

уміти:

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Визначення АІС
2) Типи АІС

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Призначення АІС
2) Класифікація АІС за різними ознаками

5.Контроль засвоєння теми

Письмова робота.

6. Рекомендована література

1. В.С.Пономаренко „Проектування інформаційних систем". Київ. Видавничий центр "Академія", 2002
2. В.Ф.Ситник, Т.А.Писаревська, Н.В.Єрьоміна, О.С.Краєва „Основи інформаційних систем", Київ, 2001
3. Петров В.Н. „Інформаційні системи", Санкт-Петербург-Москва-Харьков-Мінськ, 2002
4. В.С.Пономаренко „Інформаційні системи і технології в економіці". Київ. Видавничий центр "Академія", 2002

Загальна характеристика автоматизованих інформаційних систем

Визначення АІС.

Автоматизована інформаційна система — це система, що реалізує інформаційну технологію у сфері управління за спільної роботи управлінського персоналу та комплексу технічних засобів.

Вона призначена для автоматизованого збирання, реєстрації, збереження, пошуку, оброблення та видачі інформації за запитами користувачів. Це відбувається на основі використання економіко-математичних методів, моделей, засобів обчислювальної техніки і засобів комунікації.

АІС має забезпечувати:

- постійне спостереження за поточним станом об'єкта та його характеристик;
- підтримку професійної діяльності працівників;
- взаємодію з управлінським персоналом;
- здійснення збирання та аналізу даних для управління й автоматичного виконання програмних засобів при настанні заданого часу з формуванням необхідної звітності;
- реалізацію системи підказок і рекомендацій для користувачів;
- ефективне збереження даних у БД і можливість доступу до них будь-якого кінцевого користувача зі свого робочого місця;
- взаємодію користувачів між собою на основі безпаперової технології.
- високу економічну ефективність

Автоматизована ІС є людино-машинною системою, вона дає змогу підвищити якість управління завдяки оптимальному розподілу праці між. людиною та обчислювальною системою на всіх стадіях управління.

Споживчі властивості ІС виражають:

- функціональну повноту. Відображає рівень задоволення інформаційних потреб користувачів;
- своєчасність. Забезпечується можливістю своєчасного здобуття потрібної інформації;
- функціональну надійність. Відображає надійність інформаційного, програмного, технічного та ергономічного забезпечень під час оброблення даних;
- адаптивну надійність. Виражається у властивості системи виконувати свої функції при їх зміні під впливом навколишнього середовища.

Типи АІС.

Інформаційні системи можна класифікувати за різними ознаками, вони можуть значно різнитися за типами об'єктів управління, характером і обсягом розв'язуваних задач, типом інформації, що обробляється, і т.п.

- *Класифікація за масштабом*: одиночні (реалізуються на автономному персональному комп'ютері), групові (орієнтуються на колективне використання членами робочої групи і будуються на основі локальної мережі), корпоративні (розвиток попередніх; орієнтуються на великі компанії і можуть підтримувати рознесені територіально вузли або мережі).
- *Класифікація за рівнем або сферою діяльності*: державні, регіональні, галузеві, об'єднань, підприємств або установ, технологічних процесів.
- *Класифікація за рівнем автоматизації процесів управління*: інформаційно-пошукові, інформаційно-довідкові, системи прийняття рішень, інтелектуальні.
- *Класифікація* за ступенем обробки інформації: централізовані, децентралізовані, інформаційні системи колективного використання.
- *Класифікація за ступенем інтеграції функцій*: багаторівневі з інтеграцією за рівнями управління, багаторівневі з інтеграцією за рівнями планування.
- *Класифікація за типом інформації, що обробляється*: фактографічні і документальні. У фактографічних ІС реєструються факти – конкретні значення даних (атрибутів) про об'єкти реального світу. Основна ідея таких систем полягає в тім, що всі зведення про об'єкти (прізвища людей і назви предметів, числа, дати) повідомляються комп'ютеру в якомусь заздалегідь обумовленому форматі. Інформація, з яким працює фактографічна ІС, має чітку структуру, що дозволяє машині відрізняти одні дані від інших, наприклад прізвище від посади людини, дату народження від росту і т.п. Тому фактографічна система здатна давати однозначні відповіді на поставлені питання. Документальні ІС обслуговують принципово

інший клас задач, що не припускають однозначної відповіді на поставлене питання. Базу даних таких систем утворює сукупність неструктурованих текстових документів (статті, книги, реферати і т.д.) і графічних об'єктів, постачена тому чи іншому формалізованому апарату пошуку. Ціль системи, як правило, - видати у відповідь на запит користувача список документів або об'єктів, якоюсь мірою задовольняючих сформульовані у запиті умовам. Зазначена класифікація ІС у відомій мірі застаріла, тому що сучасні фактографічні системи часто працюють з неструктурованими блоками інформації (текстами, графікою, звуком, відео). При відомих факторах фактографічна система може перетворитися в документальну і навпаки.

- *Класифікація за способом організації (*використовується для групових і корпоративних систем*).:*
  - системи на основі архітектури файл-сервер (всі дані на сервері, а обробка виконується на клієнтських комп'ютерах: дані зчитуються клієнтом, обробляються і повертаються на сервер);
  - системи на основі архітектури клієнт-сервер (всі дані і їх обробка на сервері, клієнт тільки посилає запити і отримує відібрані дані);
  - системи на основі багаторівневої архітектури (розвиток архітектури клієнт-сервер і складається з трьох рівнів: нижчий – додатки клієнтів, середній – сервер додатків, де виконується прикладна логіка і логіка обробки даних, вищий – спеціалізований сервер даних і файлових операцій);
  - системи на основі Інтернет|Інтранет-технологій (в основному розробляються інструментальні програмні засоби, мало з базами даних).

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Структура автоматизованих інформаційних систем

2. Навчальна мета. Студент повинен:

знати: складові інформаційної системи і призначення їх елементів

уміти:

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Опис функціональної частини
2) Опис забезпечувальної частини

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Види і призначення елементів забезпечувальної частини
2) Вимоги до функціональної частини

5. Контроль засвоєння теми

Пізнорівневі питання.

6. Рекомендована література

1. В.С.Пономаренко „Проектування інформаційних систем”. Київ. Видавничий центр “Академія”, 2002
2. В.Ф.Ситник, Т.А.Писаревська, Н.В.Єрьоміна, О.С.Краєва „Основи інформаційних систем”, Київ, 2001
3. Петров В.Н. „Інформаційні системи”, Санкт-Петербург-Москва-Харьков-Мінськ, 2002
4. В.С.Пономаренко „Інформаційні системи і технології в економіці”. Київ. Видавничий центр “Академія”, 2002

Структура автоматизованих інформаційних систем

Як і будь-яка складна система, АІС складається з елементів — підсистем, які можна розглядати як АІС нижчого рівня, що складаються з елементів. У зв'язку з наявністю в АІС великої кількості елементів виникає необхідність визначення їхньої структури.

Структура АІС — внутрішня організація системи при поділі її на частини, виявлення зв'язків між цими частинами. Структуру АІС утворюють безліч елементів і відносин між ними. Найзагальнішим поділом АІС є виділення в ній *функціональної та забезпечувальної* частин.

Функціональна частина АІС є моделлю системи управління конкретним об'єктом. Вона розбивається на функціональні підсистеми. Склад функціональних підсистем АІС конкретного об'єкта різний.

Функціональна підсистема — це частина АІС, виділена за спільністю функціональних ознак управління. Назви функціональних підсистем пов'язують з функціями.

Функціональна ознака декомпозиції АІС визначає призначення підсистеми, тобто для якої сфери діяльності вона призначена і які основні цілі, задачі та функції виконує.

Специфічні особливості кожної конкретної підсистеми певного об'єкта відображаються у функціональних задачах, для автоматизації яких призначена підсистема.

Вимоги до функціональних підсистем. При виділенні функціональних підсистем в АІС необхідно дотримуватися таких вимог:

- межі задач, які утворюють підсистему, не повинні перетинатися між собою;
- задачі, що розв'язуються в підсистемах, мають бути тісно пов'язані між собою в інформаційному плані, тобто при їх розв'язуванні має використовуватися єдина вхідна інформація, а результати розв'язання одних задач мають використовуватися для розв'язання інших;
- результати розв’язання повинні мати єдиного споживача.

Параметри функціональних підсистем. При виділенні функціональних підсистем мають бути визначені їхні параметри:

- мета функціонування підсистеми;
- вид керованих ресурсів;
- особливості показників, що розраховують у підсистемі;
- підрозділи, які здійснюють управління.

Наприклад, на промисловому підприємстві в складі АІС виділяється функціональна підсистема управління санітарно-технічним постачанням. Мета функціонування підсистеми полягає в забезпеченні ритмічного виробничого процесу необхідними матеріалами та комплектуючими виробами при оптимальних запасах.

Вид керованих ресурсів — матеріальні ресурси.

Особливості показників визначаються в натуральному і вартісному виразах.

Підрозділи, які здійснюють управління, — відділ матеріально-технічного постачання і відділ зовнішньої комплектації та кооперації.

Мета функціонування підсистеми означає склад задач, що розв'язуються. Кожна задача характеризується змістом, функцією управління, ресурсом, який вона відображує, періодом часу, за який відбувається споживання ресурсу, взаємодією з іншими функціональними задачами. Зміст задачі визначає сукупність вихідних показників, які формуються й обчислюються в задачі за відповідними алгоритмами.

Саме задача є об'єктом розроблення, впровадження та експлуатації кінцевим користувачем. Поняття "задача" розглядається ширше — як закінчений комплекс оброблення інформації з забезпеченням видачі або керуючих уставок на хід виробничого процесу, або необхідної інформації для прийняття рішень управлінським персоналом, або генерації готового рішення для затвердження керівництвом. Тому задача розглядається як елемент системи управління, а не як елемент системи оброблення даних.

Внутрішня структура функціональної підсистеми. Кожній функціональній підсистемі АІС властива своя внутрішня структура. Організаційно кожна підсистема складається з сукупності АРМів управлінського персоналу, на яких розв'язуються комплекси задач. Між функціональними підсистемами встановлюються відповідні інформаційні відносини — інформаційні зв'язки. При використанні немережевих технологій оброблення інформації інформаційні зв'язки реалізуються передачею інформації на магнітних носіях. Найефективніше інформаційні зв'язки здійснюються в умовах використання мережі.

Функціональна структура АІС має орієнтуватися на ті інформаційні потреби кінцевих користувачів, які змінюються в умовах ринку, та відображати зміст і специфіку функцій управління конкретним економічним об'єктом. АІС повинна мати гнучку структуру і бути відкритою системою, тобто допускати внесення необхідних змін у розроблену модель та забезпечувати нарощування функціональних можливостей в міру необхідності.

Ця вимога реалізується за допомогою принципу модульності АІС. Кожний прикладний модуль системи має обслуговувати деяку інформаційну сферу. Головною вимогою при розробленні модулів повинна бути орієнтація системи на автоматизацію управління діяльністю об’єкта, а не на розв'язання локальних функціональних задач. При цьому функції, що реалізуються, та модулі мають розглядатися з точки зору потреб кінцевих користувачів, а не програмної реалізації.

Модульна побудова АІС передбачає безліч різних типів архітектурних рішень у межах єдиного комплексу. У різних АІС як модуль можуть розглядатися комплекси задач, АРМи або функціональна підсистема.

Забезпечувальна частина АІС.

Для експлуатації функціональних підсистем потрібні відповідні ресурси, які створюють забезпечувальні підсистеми АІС: інформаційне, технічне, програмне, математичне, організаційне, правове, лінгвістичне, ергономічне та технологічне забезпечення.

Інформаційне забезпечення є сукупністю єдиної системи класифікації та кодування, уніфікованих систем документації і масивів інформації, що використовуються в АІС.

Технічне забезпечення — комплекс технічних засобів, які забезпечують роботу АІС.

Програмне забезпечення — сукупність програм, які реалізують мету та задачі АІС і забезпечують функціонування комплексу технічних засобів системи.

Математичне забезпечення — сукупність економіко-математичних методів, моделей та алгоритмів оброблення інформації в АІС.

Організаційне забезпечення -— сукупність документів, що регламентують діяльність персоналу в АІС, взаємодію з технічними засобами і між собою в процесі розв'язування задач управління.

Правове забезпечення — сукупність правових норм, які регламентують правові відносини при функціонуванні АІС та її юридичний статус.

Лінгвістичне забезпечення — сукупність мовних засобів, призначених для формалізації природної мови, побудови і поєднання інформаційних одиниць при спілкуванні управлінського персоналу з засобами обчислювальної техніки.

Ергономічне забезпечення — сукупність методів і засобів, призначених для створення оптимальних умов високоефективної та безпомилкової діяльності людини в АІС і найшвидшого її освоєння.

Технологічне забезпечення — сукупність організаційних, методичних та технологічних документів, що регламентують процес людино-машинного оброблення інформації в АІС.

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Життєвий цикл ІС

2. Навчальна мета. Студент повинен:

знати: етапи життєвого циклу і їх призначення

уміти:

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) основні процеси життєвого циклу
2) допоміжні процеси життєвого циклу
3) організаційні процеси життєвого циклу

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) характеристика осного етапу життєвого циклу
2) Дії, що виконуються на кожному єтапі

5.Контроль засвоєння теми

Різнорівневі питання

6. Рекомендована література

1. В.С.Пономаренко „Проектування інформаційних систем". Київ. Видавничий центр "Академія", 2002
2. В.Ф.Ситник, Т.А.Писаревська, Н.В.Єрьоміна, О.С.Краєва „Основи інформаційних систем", Київ, 2001
3. Петров В.Н. „Інформаційні системи", Санкт-Петербург-Москва-Харьков-Мінськ, 2002
4. В.С.Пономаренко „Інформаційні системи і технології в економіці". Київ. Видавничий центр "Академія", 2002

Життєвий цикл ІС

У загальному випадку процес розробки інформаційної системи може бути розглянутий із двох точок зору:

- по змісту дій розроблювачів (дії. Послідовність дій);
- по часу, або стадіям життєвого циклу системи, що розробляється (організація процесу розробки: стадії, ітерації).

Життєвий цикл ІС являє собою безперервний процес, що починається з моменту прийняття рішення про створення ІС і закінчується в момент повного вилучення її з експлуатації.

Існує міжнародний стандарт, що регламентує життєвий цикл ІС - ISO/IEC 12207 (ISO – міжнародна організація по стандартам, IEC – міжнародна комісія з електротехніки). Стандарт визначає структуру життєвого циклу

Стандарт ISO/IEC 12207 визначає структуру життєвого циклу, що містить процеси, дії і задачі, що повинні бути виконані під час створення інформаційної системи. Відповідно до даного стандарту структура життєвого циклу ґрунтується на трьох групах процесів:

- основні процеси життєвого циклу (придбання, постачання, розробка, експлуатація, супровід);
- допоміжні процеси, що забезпечують виконання основних процесів (документування, керування конфігурацією, забезпечення якості, верифікація, атестація, оцінка, аудит, дозвіл проблем);
- організаційні процеси (керування проектами, створення інфраструктури проекту, визначення, оцінка і поліпшення самого життєвого циклу, навчання). Розглянемо кожну з зазначених груп більш докладно.

Основні процеси життєвого циклу

Серед основних процесів життєвого циклу найбільшу важливість мають три: розробка, експлуатація і супровід. Кожен процес характеризується визначеними задачами і методами їхнього рішення, вихідними даними, отриманими на попередньому етапі, і результатами.

Розробка

Розробка інформаційної системи містить у собі всі роботи зі створення інформаційного програмного забезпечення і його компонентів відповідно до заданих вимог:

- оформлення проектної й експлуатаційної документації;
- підготовку матеріалів, необхідних для проведення тестування розроблених програмних продуктів;
- розробку матеріалів, необхідних для організації навчання персоналу.

Розробка є одним з найважливіших процесів життєвого циклу інформаційної системи і, як правило, містить у собі стратегічне планування, аналіз, проектування і реалізацію (програмування).

Експлуатація

Експлуатаційні роботи можна підрозділити на підготовчі й основні. До підготовчого відносяться: – конфігурування бази даних і робочих місць користувачів;

– забезпечення користувачів експлуатаційною документацією;

– навчання персоналу.

Основні експлуатаційні роботи включають:

- безпосередньо експлуатацію;
- локалізацію проблем і усунення причин їхнього виникнення;
- модифікацію програмного забезпечення;
- підготовку пропозицій по удосконалюванню системи;
- розвиток і модернізацію системи.

Супровід

Служби технічної підтримки грають помітну роль у житті інформаційної системи. Наявність кваліфікованого технічного обслуговування на етапі експлуатації інформаційної системи є необхідною умовою для рішення поставлених перед нею задач, причому помилки обслуговуючого персоналу можуть приводити до явних або схованих фінансових утрат, порівнянним з вартістю самої інформаційної системи.

Основними попередніми діями при підготовці до організації технічного обслуговування інформаційної системи є наступні:

- виділення найбільш відповідальних вузлів системи і визначення для них критичності простою. Це дозволить виділити найбільш критичні складові інформаційної системи й оптимізувати розподіл ресурсів для технічного обслуговування;
- визначення задач технічного обслуговування і їхній поділ на внутрішні (розв'язуваними силами обслуговуючого підрозділу) і зовнішні (розв'язуваними спеціалізованими сервісними організаціями). У такий спосіб виробляється чітке визначення кола функцій, що виконуються, і поділ відповідальності;
- проведення аналізу наявних внутрішніх і зовнішніх ресурсів, необхідних для організації технічного обслуговування в рамках описаних задач і поділу компетенції. Основні критерії для аналізу: наявність гарантії на устаткування, стан ремонтного фонду, кваліфікація персоналу;
- підготовка плану організації технічного обслуговування, у якому необхідно визначити етапи дій, що виконуються, терміни їхнього виконання, витрати на етапах, відповідальність виконавців.

Забезпечення якісного технічного обслуговування інформаційної системи вимагає залучення фахівців високої кваліфікації, що у стані вирішувати не тільки щоденні задачі адміністрування, але і швидко відновлювати працездатність системи при збоях.

Допоміжні процеси життєвого циклу

Серед допоміжних процесів одне з головних місць займає керування конфігурацією для підтримки основних процесів життєвого циклу інформаційної системи, насамперед розробки і супроводу. При розробці проектів складних інформаційних систем, що складаються з багатьох компонентів, кожний з яких може розроблятися незалежно і, отже, мати кілька варіантів реалізації і/або кілька версій однієї реалізації, виникає проблема обліку їхніх зв'язків і функцій, створення єдиної структури і забезпечення розвитку всієї системи. Керування конфігурацією дозволяє організовувати, систематично враховувати і контролювати внесення змін у різні компоненти інформаційної системи на всіх стадіях її життєвого циклу.

Організаційні процеси життєвого циклу

Керування проектом зв'язано з питаннями планування й організації робіт, створення колективів розроблювачів і контролю за термінами і якістю виконуваних робіт. Технічне й організаційне забезпечення проекту включає:

- вибір методів і інструментальних засобів для реалізації проекту;
- визначення методів опису проміжних станів розробки;
- розробку методів і засобів іспитів створеного програмного забезпечення;
- навчання персоналу.

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Визначення вимог до бази даних

2. Навчальна мета. Студент повинен:

знати: перелік вимог, які можна висувати до баз даних

уміти: формувати вимоги до баз даних

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) основні вимоги до баз джаних
2) вимоги до спроектованої бази даних

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) перелік і опис вимог до бази даних

5.Контроль засвоєння теми

Письмова робота.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X
3. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

Визначення вимог до бази даних

Основні вимоги, які пред'являються до операційних баз даних:

1.***Простота поновлення даних***. Під операцією поновлення розуміють додавання, видалення і зміни даних.

2.***Висока швидкодія*** (малий час відгуку на запит). Час відгуку - проміжок часу від моменту запиту до БД і фактичним отриманням даних. Схожим є термін час доступу - проміжок часу між видачею команди запису (зчитування) і фактичним отриманням даних. Під доступом розуміється операція пошуку, читання даних або запису їх.

Найважливішими є перші дві суперечливих вимоги: підвищення швидкодії вимагає спрощення структури БД, що, в свою чергу, ускладнює процедуру поновлення даних, збільшує їх надмірність.

3.Незалежність даних.

Незалежність даних - можливість зміни логічної і фізичної структури БД без зміни уявлень користувачів. Незалежність даних передбачає інваріантність до характеру зберігання даних, програмного забезпечення і технічних засобів. Вона забезпечує мінімальні зміни структури БД при змінах стратегії доступу до даних і структури самих вихідних даних. Це досягається, як буде показано далі, «зміщенням» всіх змін на етапи концептуального і логічного проектування з мінімальними змінами на етапі фізичного проектування.

4. ***Спільне використання даних*** багатьма користувачами.

5. ***Безпека даних*** - захист даних від навмисного або ненавмисного порушення секретності, спотворення або руйнування.

Безпека даних включає їх цілісність і захист. Цілісність даних - стійкість збережених даних до руйнування і знищення, пов'язаних з несправностями технічних засобів, системними помилками і помилковими діями користувачів.

Вона передбачає:

• відсутність неточно введених даних або двох однакових записів про одне й те ж факт;

• захист від помилок при оновленні БД;

• неможливість видалення порізно (каскадне видалення) пов'язаних даних різних таблиць;

• неспотвореному даних при роботі в многопользовательском режимі і в розподілених базах даних;

• збереження даних при збоях техніки (відновлення даних).

Цілісність забезпечується тригерами цілісності - спеціальними додатками-програмами, що працюють при певних умовах. Для деяких СУБД (наприклад, Access, Paradox) тригери є вбудованими.

Захист даних від несанкціонованого доступу передбачає обмеження доступу до конфіденційних даних і може досягатися:

• введенням системи паролів;

• отриманням дозволів від адміністратора бази даних (АБД);

• забороною від АБД на доступ до даних;

• формуванням видів - таблиць, похідних від вихідних і призначених конкретним користувачам.

6.Стандартизація побудови та експлуатації БД (фактично СУБД).

Стандартизація забезпечує спадкоємність поколінь СУБД, спрощує взаємодію БД одного покоління СУБД з однаковими і різними моделями даних. Стандартизація (ANSI / SPARC) здійснена в значній мірі в частині інтерфейсу користувача СУБД і мови SQL. Це дозволило успішно вирішити завдання взаємодії різних реляційних СУБД як за допомогою мови SQL, так і з застосуванням програми Open DataBase Connection (ODBC). При цьому може бути здійснений як локальний, так і віддалений доступ до даних (технологія клієнт-сервер або мережевий варіант).

7.***Адекватность відображення даних*** відповідної предметної області.

8.***Дружелюбний інтерфейс користувача***.

Правильно спроектована БД повинна відповідати таким вимогам:

1. **Мінімальна надмірність. Несуперечливість**.

Мінімальна надмірність означає те, що дані в БД не повинні дублюватися. Надмірність даних, якщо вона існує, тягне дві небезпеки:

-неоправданно велика витрата пам'яті і зменшення часу відгуку системи при обробці надмірно великих обсягів даних.

-порушення несуперечності даних, тобто виникнення такої ситуації, коли в різних місцях машинної пам'яті зберігаються суперечливі дані. Виникнення суперечливості надзвичайно небезпечно для БД.

Суперечливість може виникнути в результаті*корректіровкііізбиточних даних.* При внесенні змін до логічну запис може трапитися так, що окремі екземпляри цієї записи, що зберігаються в різних місцях машинної пам'яті, виявляться нескоректована. Програмісту доводиться проявляти особливу увагу до організації процесу коригування надлишкових даних і розробляти спеціальні програми, що запобігають появі суперечливості.

Суперечливість може виникнути і при*корректіровкенеізбиточних даних.* Централізоване зберігання даних є причиною високої ймовірності того, що двом або більше користувачам одночасно знадобляться одні й ті ж дані. Якщо один з користувачів звертається до даних, а інший в той же час вносить в них зміни, будуть отримані суперечливі дані. Пояснюється це тим, що процес оновлення даних вимагає певного часу, протягом якого одні й ті ж дані виявляються на різних стадіях відновлення. Під час відвідування таких даними паралельно працюючих програм будуть отримані суперечливі відомості.

В СУБД існують складні механізми блокування оновлюваних даних від доступу до них інших користувачів. Паралельні запити до одних і тих же даних зазвичай виконуються послідовно.

У ряді СУБД є засоби, що запобігають дублювання і виникнення суперечності даних. В іншому випадку такі кошти розробляє системний програміст.

2. **Цілісність даних.** Вимога повноти і несуперечливість даних

Цілісність даних означає те, що в БД повинні зберігатися тільки правильні дані, тобто дотримуються логічні умови, відповідно до яких дані вважаються правильними. Руйнування і спотворення даних можливі в результаті необережних дій користувачів, в результаті помилок в програмах і збоїв обладнання.

Існують спеціальні методи і прийоми забезпечення цілісності.

Для забезпечення цілісності на дані, що зберігаються в БД, накладають ***обмеження.*** При цьому визначаються умови, яким повинні відповідати значення даних.Наприклад, один і той же службовець не може мати два різних року народження і т.п .. Такі обмеження називаються ***законами БД.*** Здійснимість законів БД періодично перевіряється СУБД.

Для запобігання можливості **введення неправильних даних** розробляються засоби контролю правильності даних, що вводяться. Так, наприклад, можна використовувати процедури, перевіряючі приналежність вводятьсязначень певного діапазону допустимих значень. Наприклад, кількість робочих днів обмежується зверху кількістю днів у поточному місяці.

Цілісність даних може порушитися при невдалому завершенні транзакції. ***Транзакцією*** називається деяка неподільна послідовність операцій над даними, виконувана по одному запиту до БД. Прикладом транзакції є операція переказу грошей з одного рахунку на інший в банківській системі. Тут необхідно послідовно виконати такі операцій. Гроші знімаються з одного рахунку, дані коригуються, потім гроші додаються до іншого рахунку і дані знову коригуються.Якщо хоча б одне з дій не виконується успішно, результат транзакції виявиться невірним. СУБД повинна відстежувати хід виконання транзакції від початку до її завершення. Якщо з якоїсь причини будь-яка з операцій не виконалася, то транзакція скасовується повністю. При цьому виконується "відкат" шляхом скасування всіх уже виконаних змін.

В БД повинні бути передбачені кошти відновлення даних після програмних збоїв і збоїв обладнання. Існують програми створення резервних копій і спеціальні програми, які автоматично фіксують будь-які внесені в БД зміни (створюється файл коректур). Якщо поточна версії БД зіпсована, то береться попередня версія, в неї вносяться зміни, зафіксовані в файлі коректур, і поточне (актуальне) стан БД відновлюється.

Різні СУБД в тій чи іншій мірі мають у своєму розпорядженні засобами забезпечення цілісності даних. В іншому випадку такі кошти розробляються системним програмістом.

3. **Незалежність даних**.

Незалежність даних означає те, що прикладні програми не повинні залежати від даних, що зберігаються, тобто від способу зберігання даних у фізичній пам'яті. Це дозволяє додавати в БД нові дані, змінювати структури зберігання даних, створювати на БД нові додатки. Раніше створені програми при цьому не повинні "відчувати" ці зміни.

СУБД зазвичай забезпечують цю вимогу.

4. **Можливість ведення** (додавання і видалення) та актуалізації (коригування, модифікації) даних. Структура БД повинна дозволяти включати нові і видаляти застарілі дані, коригувати збережені дані без руйнування логічних зв'язків, встановлених в схемі БД.Для цього схема БД повинна бути правильно розроблена, а операції ведення БД не повинні порушувати схему БД.

5. **Безпека і секретність**.

Безпека і секретність означає захист даних від несанкціонованого доступу, умисного і ненавмисного руйнування даних, розкрадання даних. Система захисту БД покликана вирішувати такі завдання.

- Ідентифікація користувачів. Даними, що зберігаються в БД повинні користуватися тільки особи, які мають на це право і які підтвердили свої повноваження.Найбільш поширеним способом вирішення цього завдання є система паролів.
- Обмеження доступу до даних. Кожен користувач повинен працювати тільки з тими даними, які необхідні для вирішення його завдань, інші дані повинні бути для нього "невидимими".

Кожному користувачеві надаються певні повноваження (привілеї) для роботи з даними. Йому може бути надано право тільки читання з БД, право введення в БД або право поновлення і т.п. Всі привілеї надаються тільки адміністратору БД.

-забезпечення Секретності даних. Секретні дані необхідно захищати від доступу системою спеціальних, досить складних паролів. Сильно вразливі дані слід шифрувати.

Засоби захисту та забезпечення безпеки даних містяться в СУБД або розробляються системним програмістом.

4. Висока продуктивність. Мінімальні витрати.

Організація БД і методи доступу до даних повинні забезпечувати високу швидкість обробки даних таку, щоб користувач міг працювати з БД в діалоговому режимі. Вартість обслуговування користувачів не повинна бути високою.

Можливість виконання цих вимог визначається цілою низкою чинників: об'ємом даних, що зберігаються, швидкодією техніки, способом організації даних в БД і багато в чому залежить від рішень, прийнятих розробниками на етапі створення БД. Наприклад, можна організувати спосіб розміщення даних на носії таким чином, що найбільш часто використовувані дані зберігаються на найбільш доступних ділянках зовнішньої пам'яті.

7. **Дотримання стандартів**.

Подання даних в БД, супровідна документація, спосіб взаємодії користувача з БД повинні відповідати певним стандартам. Стандарти можуть бути корпоративними, відомчими, промисловими, національними та міжнародними. Дотримання стандартів абсолютно необхідно для спільного використання даних і для організації обміну даними між окремими системами. Наприклад, без прийняття певних стандартів не можна було б організувати мережу Internet.

Розвиток теорії і практики створення інформаційних систем, заснованих на концепції баз даних, створення уніфікованих методів і засобів організації та пошуку даних дозволяють зберігати і обробляти інформацію про всі більш складних об'єктах і їх взаємозв'язках, забезпечуючи багатоаспектні інформаційні потреби різних користувачів. Основні вимоги, що висуваються до банків даних, можна сформулювати так:

- **Багаторазове використання даних:** користувачі повинні мати можливість використовувати дані по-різному.
- **Простота:** користувачі повинні мати можливість легко дізнатися і зрозуміти, які дані є в їх розпорядженні.
- **Легкість використання:** користувачі повинні мати можливість здійснювати (процедурно), простота доступу до даних, при цьому всі складності доступу до даних повинні бути приховані в самій системі управління базами даних.
- **Гнучкість використання:** звернення до даних або їх пошук повинні здійснюватися за допомогою різних методів доступу.
- **Швидка обробка запитів на дані:** запити на дані, повинні оброблятися за допомогою високорівневого *мови запитів,* а не тільки прикладними програмами, написаними з метою обробки конкретних запитів.
- **Мова взаємодії** кінцевих користувачів з системою повинен забезпечувати кінцевим користувачам можливість отримання даних без використання прикладних програм.

База даних - це основа для майбутнього нарощування прикладних програм: *бази даних* повинні забезпечувати можливість швидкої і дешевої розробки нових додатків.

- **Збереження витрат розумової праці:** існуючі програми і *логічні структури* даних не повинні перероблятися при внесенні змін до бази даних.
- **Наявність інтерфейсу прикладного програмування:** прикладні програми повинні мати можливість просто і ефективно виконувати запити на дані; програми повинні бути ізольованими від розташування файлів і *способів адресації* даних.
- **Розподілена обробка даних:** система повинна функціонувати в умовах обчислювальних мереж і забезпечувати ефективний доступ користувачів до будь-яких даних розподіленої БД, розміщеним в будь-якій точці мережі.
- **Адаптивність і розширюваність:** база даних повинна бути настроюється, причому установка не повинна викликати перезапису прикладних програм. Крім того, що поставляється з СУБД набір визначених типів даних повинен бути розширюваним - в системі повинні бути кошти для визначення нових типів і не повинно бути відмінностей у використанні системних і певних користувачем типів.
- **Контроль цілісності даних:** система повинна здійснювати контроль помилок в даних і виконувати перевірку взаємного логічного відповідності даних.
- **Відновлення даних після збоїв:** автоматичне відновлення без втрати даних транзакції. У разі апаратних або програмних збоїв система повинна повертатися до деякого узгодженим станом даних.
- **Допоміжні засоби** повинні дозволяти розробнику або *адміністратору бази даних* передбачити і оптимізувати продуктивність системи.
- **Автоматична реорганізація і переміщення:** система повинна забезпечувати можливість переміщення даних або автоматичну реорганізацію *фізичної структури* .

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Сутності сильного і слабкого типів

2. Навчальна мета. Студент повинен:

знати: поняття сутностей сильного і слабкого типу

уміти:

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Визначення сутності сильного типу
2) Визначення сутності слабкого типу

4.Результати самостійної позааудиторної діяльності, питання для самоконтролю

1)Відміни сутностей сильного і слабкого типу

5.Контроль засвоєння теми

Різнорівневі питання.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X
3. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

Сутності сильного і слабкого типів

Сутності сильного і слабкого типів (Головні та залежні сутності)

Типи сутностей можуть також підрозділятися на сильні і слабкі.

Сутність сильного типу. Тип сутності, існування якого не залежить від якогось іншого типу сутності.

Тип сутності називається сильним, якщо його існування не залежить від наявності сутностей іншого типу. Характерною особливістю сильного типу є те, що кожен екземпляр сутності може бути однозначно ідентифікований за допомогою атрибуту (атрибутів) первинного ключа сутності цього типу. Наприклад, кожен співробітник компанії може бути однозначно ідентифікований за допомогою атрибуту STAFF, який є первинним ключем сутності типу Персонал.

Сутність слабкого типу. Тип сутності / існування якого залежить від якогось іншого типу сутності.

Сутність слабкого типу залежить від наявності сутності іншого типу. Приклад суті слабкого типу Перевагу (Побажання). Характерною особливістю слабкою суті є те, що кожен екземпляр сутності можна однозначно ідентифікувати за допомогою тільки тих атрибутів, які відносяться до сутності цього типу. Наприклад, для сутності Перевагу немає первинного ключа. Це означає, що кожен екземпляр сутності типу Перевагу неможливо ідентифікувати тільки за допомогою атрибутів цієї сутності. Однозначно ідентифікувати кожне побажання клієнта можна тільки, враховуючи зв'язок конкретного побажання з деяким клієнтом, який однозначно ідентифікується з використанням первинного ключа для сутності типу клієнта, а саме: clientNo. У даному прикладі сутність Перевагу розглядається як залежна у своєму існуванні від сутності Клієнт яка описує майбутнього орендаря, охочого орендувати об'єкти нерухомості конкретного типу і на певних умовах.

Сутності слабкого типу іноді називають дочірніми, залежними або підлеглими, а сутності сильного типу - батьківськими, сутностями власниками або домінантними.

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Архітектура сервера

2. Навчальна мета. Студент повинен:

знати: елементи архітектури і їх призначення

уміти:

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Характеристика серверної частини
2) Характеристика клієнтської частини

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Чому алгоритм має властивість „визначеність”, „масовість”..

5.Контроль засвоєння теми

Письмова робота.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X
3. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

Архітектура сервера

SQL Server – це заснований на SQL сервер реляційних баз даних.

База даних – це звичайний файл даних, що представляє собою певну область, де зберігається інформація, її схема (schema) з описом структури і типів даних, визначеннями таблиць, представленнями, дозволами і т.п. По суті, схема містить всі допоміжні відомості, необхідні для роботи бази даних. Зазвичай база складається з двох компонентів: файлів даних з інформацією та програм доступу до бази даних. Ці програми називають системами управління базами даних або скорочено СКБД (database management system, DBMS).

SQL Server призначений для роботи в клієнт-серверних системах. Подібна архітектура має на увазі, що файли бази даних і СУБД розташовуються на комп'ютері-сервері. Додатки виконуються на окремих клієнтських комп'ютерах і взаємодіють з сервером по мережі за допомогою клієнтського інтерфейсу.

Для роботи з даними в базі використовується набір команд, які розуміє СУБД – мова (language) СКБД.

СУКД SQL Server підтримує централізовану базу даних з архітектурою *клієнт-сервер.* Технологія клієнт-сервер передбачає, що система поділяється на дві частини:

- *клієнтська частина* (front-end), яка забезпечує графічний інтерфейс і знаходиться на комп'ютері користувача і
- *серверна частина* (back-end), яка знаходиться на спеціально виділених комп'ютерах (серверах) і забезпечує зберігання бази даних, управління даними, поділ інформації, адміністрування і безпеку.

Запит на виконання операції з даними (наприклад, звичайна вибірка), що видається клієнтом (робочою станцією), породжує на сервері пошук та вилучення даних. Витягнуті дані (але не файли) транспортуються по мережі від сервера до клієнта.

Для функціонування мережі використовуються різні протоколи. SQL Server може використовувати більшість мережевих протоколів операційної системи. Для забезпечення мережевої взаємодії використовуються різні мережеві бібліотеки, що передають дані по певним мережевим протоколам. Ці бібліотеки реалізовані у вигляді DLL-файлів, що підключаються до операційної системи. Бібліотека розширює базові можливості протоколу.

У дуже великих системах підтримка баз даних здійснюється групою серверів баз даних. Сервера можуть об'єднуватися для розподілу навантаження. При цьому кожен сервер управляється індивідуально.

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Системні бази даних

2. Навчальна мета. Студент повинен:

знати: призначення системних баз даних

уміти:

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) перелік і розташування системних баз даних
2) призначення кожної системної бази даних

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) яка база даних використовується як шаблон при створенні нової бази даних

5.Контроль засвоєння теми

Письмова робота.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X
3. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

Системні бази даних

При установці SQL Server створюэться 4 системні бази даних:

- master
- msdb
- model
- tempdb

Ці бази дані і будь-які інші створювані бази мыстять в обов'язковому порядку 18 системних таблиць, що зберігають інформацію, що визначає структуру й організацію відповідної бази даних.

Усі бази даних підтримують можливість автоматичного збільшення свого розміру.

База даних master зберігає всю системну інформацію системи SQL Server, включаючи системні облікові записи і системні параметри конфігурації, інформацію про існування всіх інших баз даних і місць розташування їхніх первинних файлів, що містять інформацію ініціалізації для баз даних користувачів. Фактично база даних `master` складається з двох файлів:

- файлу даних Master.mdf (за замовчуванням 7.5 Мбайт)
- файлу журналу транзакцій Mastlog.ldf (за замовчуванням 1.0 Мбайт)

Обидва файли зберігаються в папці Data.

База даних tempdb. Зберігає всі тимчасові об'єкти, що використовуються при роботі з SQL Server: таблиці, збережені процедури, змінні, курсори і т.д. Зберігаються і системні і користувацькі об'єкти. База створюється заново при кожнім запуску SQL Server. База даних є глобальним ресурсом, доступним усім користувачам.

База даних model. Створення нових баз даних виконується шляхом копіювання цієї системної бази даних. Весь її вміст і всі параметри конфігурації цілком переносяться в нову базу, незалежно від методу створення. За замовчуванням розмір бази 1.5 Мбайт. Якщо при створенні своєї бази даних користувач хоче, щоб його база була точною копією бази model, то при створенні не потрібно вказувати ніяких параметрів, крім імені. Якщо розмір і склад зазначені явно, то скопійована база зміниться відповідним чином.

База даних msdb. Використовується службою SQL Server для планування подій і реєстрації операторів.

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Загальна характеристика мови SQL

2. Навчальна мета. Студент повинен:

знати: історія появи мови, мета розробки, стандартизація мови

уміти:

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) розробки IBM
2) початковий вигляд мови
3) відміна мови від інших мов

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) необхідність в появі нової мови баз ланих
2) Основні властивості мови

5.Контроль засвоєння теми

Письмова робота.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .– М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-0039-X
3. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.– К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

Загальна характеристика мови SQL

На початку 1970-х років в одній з дослідницьких лабораторій компанії IBM була розроблена експериментальна реляційна СУБД IBM System R, для якої потім був створений спеціальний мова SEQUEL, що дозволяв відносно просто управляти даними в цій СУБД. Абревіатура SEQUEL розшифровувалась як Structured English QUEry Language - «структурований англійська мова запитів». Пізніше з юридичних міркувань мову SEQUEL був перейменований в SQL.

Метою розробки було створення простого непроцедурного мови, яким міг скористатися будь-який користувач, який навіть не має навичок програмування. Власне розробкою мови запитів займалися Дональд Чемберлін (Donald D. Chamberlin) і Рей Бойс (Ray Boyce). Пет Селінджер (Pat Selinger) займалася розробкою вартісного оптимізатора (cost-based optimizer), Реймонд Лорі (Raymond Lorie) займався компілятором запитів.

Першими СУБД, що підтримують нову мову, стали в 1979 році Oracle V2 для машин VAX від компанії Relational Software Inc.(Згодом стала компанією Oracle) і System / 38 від IBM, заснована на System / R. Усупереч поширеній думці, першою стала саме СУБД Oracle.

Зростання кількості даних, необхідність їх зберігання і обробки привели до того, що виникла потреба у створенні стандартного мови баз даних, який міг би функціонувати в численних комп'ютерних системах різних видів. Дійсно, з його допомогою користувачі можуть маніпулювати даними незалежно від того, чи працюють вони на персональному комп'ютері, мережевій робочій станції або універсальної ЕОМ.

SQL (англ. Structured Query Language - «мова структурованіх Запитів») - універсальний комп'ютерний інформаційно-логічна мова, что з'явився в результате розробки реляційної моделі Даних, застосовуваного для создания, модіфікації та управління данімі в реляційніх базах Даних.

Спочатку, SQL БУВ основним способом роботи користувача з базою Даних и представляв собою невеликі сукупність команд (Операторів) допускаються создания таблиці, додавання в табліці Нових запісів, витяг запісів з таблиці (відповідно до заданого умів), відалення запісів и зміна структури таблиці. У зв 'язку з ускладненням мову SQL ставши більш прикладні мовою програмування, а Користувачі получил можлівість використовуват візуальні побудовнікі Запитів.

SQL принципова відрізняється від традіційніх алгорітмічніх мов програмування самперед тім, что ВІН відноситься до непроцедурного мов. Мовою типу Кобол або Сі можна Записати крок за кроком всі інструкції, необхідні для виконан програми. Мова SQL дозволяє Задати только ті, "що нужно делать", а самє виконан окрема операцій ("як робити") покладається безпосередно на СУБД. такий підхід значний мірою візначається самою філософією реляційніх баз Даних.СУБД в даного випадка розглядається як "чорний ящик", и что відбувається Всередині него, користувача не винних стосуватись. ЙОГО винне цікавіті только внесення до бази Даних необхідніх змін и Отримання правильної ВІДПОВІДІ на запит.

Іншою особлівістю SQL є так кличуть входити трізначна логіка. У більшості мов булево виразі может прійматі только два значення: істина и брехня. Мова SQL дозволяє запісуваті в базу Даних значення NULL (пусте значення). NULL - це Спеціальний код, Який поміщається в стовпець табліці, если з якоїсь причини в ньом відсутні дані. Коли значення NULL бере участь в операціях порівняння, логічний результат буде ні істина и ні брехня, а невідомо.

Всі мови маніпулювання данімі, створені для багатьох СУБД до з'явиться реляційніх баз Даних, були орієнтовані на операции з данімі, представлених у виде логічніх запісів файлів. Зрозуміло, це Вимагаю від користувача детального знання организации зберігання Даних и серйозно зусіль для вказівки того, Які дані необхідні , де смороду розміщуються и як їх отріматі.

Завдяки роботі з файловим сервером СУБД, безліч користувачів отримують доступ до одних и тих же баз Даних. Спрощується розробка різніх автоматизованого систем управління

організаціями. Однако при такому підході вся обробка Запитів з програм або з якихось терміналів користувача ЕОМ на них и віконується, тому для реализации даже простого запиту та патенти зчітуваті з файлового сервера або запісуваті на него цілі файли, а це веде до конфліктніх СИТУАЦІЙ и перевантаження мережі. Для віключення зазначену недоліків булу предложено технологія клієнт-сервер, но при цьом знадобівся єдина мова спілкування з сервером - вибір припав на SQL.

Розглянуто мову SQL орієнтований на операции з данимі, представлених у виде логічно взаємопов'язаніх Сукупний таблиці-отношений. Найважлівіша особлівість его структур - орієнтація на кінцевій результат Обробка даних, а не на процедуру цієї ОБРОБКИ. Мова SQL сам візначає, де знаходяться дані, Індекси и даже Які найбільш ефектівні послідовності операцій слід використовуват для Отримання результату, а тому вказувати ЦІ деталі в запіті до бази даних НЕ нужно.

SQL зараз отримав широке поширення і фактично превратился в стандартну мову реляційніх баз Даних. Стандарт на мову SQL був випущений американским національним інститутом стандартів (ANSI) в 1986 р, а в 1987 р Міжнародна організація стандартів (ISO) прийнять его в якості МІЖНАРОДНОГО . подалі розвиток мови постачальником СУБД зажадало Прийняття в 1992 году новой розшіреної стандарту (ANSI SQL-92 або просто SQL2). Наступний стандартом ставши SQL: +1999 (SQL3). у Сейчас годину Діє стандарт, чинний в 2003 году (SQL: 2003) з невелика модіфікаціямі, внесеними пізніше.

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Типи даних

2. Навчальна мета. Студент повинен:

знати: призначення типів даних, типи даних, що використовуються в SQL Server

уміти:

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Типи даних

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Який тип даних потрібно використовувати для збереження зображень

5.Контроль засвоєння теми

Використання на лабораторних роботах.

5. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .– М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-0039-X
3. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.– К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

Типи даних

Тип даних	Опис
Bit	Один біт. У стовпці з цим типом може зберігатися або 0, або 1
binary(n)	Двійкові дані фіксованої довжини до 8000 байт
vabinary(n)	Двійкові дані змінної довжини до 8000 байт
Image	Двійкові дані довжиною до 2 Гбайт. Простір для зберігання даних цього типу відводиться сторінками, розмір яких складає 8 Кбайт
Decimal	Використовується для зберігання дрібних чисел з фіксованою кількістю знаків до і після десяткової точки
Numeric	Є аналогом типу даних Decimal
float(n)	Використовується для зберігання дрібних чисел з приблизною точністю
Real	Є аналогом float(n)
Int	Займає 4 байта і може зберігати цілі числа в діапазоні від -2^{31} до $2^{31}-1$
Smallint	Невелике ціле. Займає 2 байта і може зберігати цілі числа в діапазоні -2^{15} до $2^{15}-1$ (от -32 768 до 32 767)
Tinyint	Маленьке ціле. Займає 1 байт і може зберігати цілі числа в діапазоні від 0 до 255
Bigint	Велике ціле. Займає 8 байт і може зберігати цілі числа в діапазоні від -2^{63} до $2^{63}-1$
char(n)	Строковий тип даних фіксованої довжини без підтримки Unicode довжиною 8000 символів
nchar(n)	Строковий тип даних фіксованої довжини з підтримкою Unicode довжиною до 4000 символів
varchar(n)	Строковий тип даних змінної довжини без підтримки Unicode довжиною до 8000 символів
nvarchar(n)	Строковий тип даних змінної довжини з підтримкою Unicode довжиною 4000 символів
text	Строковий тип даних без підтримки Unicode довжиною до 2 мільярдів символів. Простір для зберігання даних цього типу відводиться сторінкам, розмір яких складає 8 Кбайт
ntext	Строковий тип даних з підтримкою Unicode довжиною до 1 мільярда символів. Простір для зберігання даних цього типу відводиться сторінкам, розмір яких складає 8 Кбайт
datetime	Тип даних для зберігання дати та часу з високою точністю в діапазоні від 1 січня 1753 і до 31 грудня 9999, що займає 8 байт
small datetime	Тип даних для зберігання дати та часу з нормальною точністю в діапазоні з 1 січня 1900 і до б червня 2079, що займає 4 байт
money	Використовується для зберігання грошових даних у великому діапазоні. Забезпечує точність до 4 знаків після десяткової точки і займає 8 байт
smallmoney	Використовується для зберігання грошових даних у нормальному діапазоні. Забезпечує точність до 4 знаків після десяткової точки і займає 4 байта
sql variant	Тип даних, що дозволяє зберігати дані декількох типів в одному і тому ж стовпці. Наприклад, при використанні цього типу даних у стовпці можуть одночасно перебувати дані цілочисельні, дробові, символьні і ін.
sysname	Цей тип даних, по суті, є не вбудованим, *а* для користувача (user-defined data type), створюваним в системних базах даних автоматично при установці SQL Server 2000. Використовується для вказівки імен об'єктів. Аналог NVARCHAR (128)
timestamp	Часовий штамп. Значення в стовпці цього типу змінюються SQL Server 2000 автоматично при кожній зміні рядка. Для типу даних мітка також можна використовувати псевдонім RowVersion
table	Цей тип даних призначений для тимчасового зберігання складних наборів даних.

	Підтримується створення змінних типу таблиці. Крім того, функції користувача можуть повертати значення типу стіл, що надає широкі можливості для розробників. Однак тип даних таблиці не може використовуватися для стовпців таблиці. Він підходить тільки для змінних Transact-SQL, параметрів збережених процедур і для значень, що повертаються функціями користувача
uniqueidentifier	Використовується для зберігання глобальних унікальних ідентифікаторів (Global Unique Identifier, GUIO)
cursor	Призначений для зберігання посилань на курсори

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Вбудовані математичні функції

2. Навчальна мета. Студент повинен:

знати: функції, які можна використовувати

уміти: використовувати при роботі з базами даних

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Математичні функції

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

5. Контроль засвоєння теми

Письмова робота.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .– М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-0039-X
3. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.– К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

Математичні функції

ABS	обчислює абсолютне значення числа
ACOS	обчислює арккосинус
ASIN	обчислює арксинус
ATAN	обчислює арктангенс
ATN2	обчислює арктангенс з урахуванням квадратів
CEILING	виконує округлення вгору
COS	обчислює косинус кута
COT	повертає котангенс кута
DEGREES	перетворює значення кута з радіан в градуси
EXP	повертає експоненту
FLOOR	виконує округлення вниз
LOG	обчислює натуральний логарифм
L O G 1 0	обчислює десятковий логарифм
PI	повертає значення "пі"
POWER з	водить число до степеня
RADIANS	перетворює значення кута з градуса в радіани
RAND	повертає випадкове число
ROUND	виконує округлення із заданою точністю
S I G N	про пределяет знак числа
SIN	обчислює синус кута
SQUARE	виконує зведення числа в квадрат
SQRT	витягує квадратний корінь
TAN	повертає тангенс кута

```
SELECT      Товар.Название, Сделка.Количество,
Round(Товар.Цена*Сделка.Количество
   *0.05,1)
AS Налог
FROM Товар INNER JOIN Сделка
   ON Товар.КодТовара=
   Сделка.КодТовара
```

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Вбудовані рядкові функції

2. Навчальна мета. Студент повинен:

знати: функції, які можна використовувати

уміти: використовувати при роботі з базами даних

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) рядкові функції

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

5. Контроль засвоєння теми

Письмова робота.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .– М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-0039-X
3. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.– К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

Рядкові функції

ASCII повертає код ASCII лівого символу рядка
CHAR за кодом ASCII повертає символ
CHARINDEX визначає порядковий номер символу, з якого починається входження підрядка в рядок
DIFFERENCE повертає показник збігу рядків
LEFT повертає вказане число символів з початку рядка
LEN повертає довжину рядка
LOWER переводить всі символи рядка в нижній регістр
LTRIM видаляє пробіли на початку рядка
NCHAR повертає за кодом символ Unicode
PATINDEX виконує пошук підрядка в рядку за вказаною шаблоном
REPLACE замінює входження підрядка на вказане значення
QUOTENAME конвертує рядок в формат Unicode
REPLICATE виконує тиражування рядки певне число разів
REVERSE повертає рядок, символи якій записані у зворотному порядку
RIGHT повертає вказане число символів з кінця рядка
RTRIM видаляє прогалини в кінці рядка
SOUNDEX повертає код звучання рядка
SPAC повертає вказане число прогалин
STR виконує конвертування значення числового типу в символьний формат
STUFF видаляє вказане число символів, замінюючи нової підрядком
SUBSTRING повертає для рядка підрядок зазначеної довжини з заданого символу
UNICODE повертає Unicode-код лівого символу рядка
UPPER переводить всі символи рядка у верхній регістр

Приклад. Використання функції LEFT для отримання ініціалів клієнтів.

```
SELECT Фирма, [Прізвище]+""
   +Left([Ім’я],1)+"."
   +Left([По батьколві],1)
   +"." AS ПІБ
FROM Клієнт
```

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Функції для роботи з датою і часом

2. Навчальна мета. Студент повинен:

знати: функції, які можна використовувати

уміти: використовувати при роботі з базами даних

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Функції роботи з датою і часом

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

5. Контроль засвоєння теми

Письмова робота.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .– М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)
2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-0039-X
3. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.– К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

Функції для роботи з датою і часом

DATEADD додає до дати зазначена значення днів, місяців, годин і т.д.
DATEDIFF повертає різницю між вказаними частинами двох дат
DATENAME виділяє з дати вказану частину і повертає її в символьному форматі
DATEPART виділяє з дати вказану частину і повертає її в числовому форматі
DAY повертає число з вказаної дати
GETDATE повертає поточний системний час
ISDATE перевіряє правильність виразу на відповідність одному з форматів введення дати
MONTH повертає значення місяця з вказаної дати
YEAR повертає значення року з вказаної дати

```
SELECT Year(Дата) AS Год, Month(Дата)
  AS Месяц,
  Sum(Количество) AS Общ_Количество
FROM Сделка
GROUP BY Year(Дата), Month(Дата)
```

Використання функцій YEAR і MONTH для визначення загальної кількості товару, проданого за кожен місяць кожного року.

```
DECLARE @d DATETIME
DECLARE @y INT
SET @d=’29.10.03’
SET @y=DATEPART(yy,@d)
SELECT @y
```

Перетворення дати в зручний для виведення вид

```
CONVERT(varchar(12), GETDATE(), 101)
```

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Індексування в базах даних

2. Навчальна мета. Студент повинен:

знати: призначення і типи індексування

уміти: використовувати індексування

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Визначення індексу
2) Типи індексів

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Основне призначення індексів
2) Організація індексів

5.Контроль засвоєння теми

Письмова робота.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)

2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X

3. Пасічник, В.В. Організація баз даних та знань/ В.В.Пасічник, В.А.Резніченко.–К.: Видавнича група BHV, 2006.– 384 сю:іл.– ISBN966-552-156-X

Індексування в базах даних

індекс - структура даних, яка допомагає СУБД швидше виявити окремі записи в файлі і скоротити час виконання запитів користувачів.

Індекс в базі даних аналогічний предметному покажчику в книзі. Це - допоміжна структура, пов'язана з файлом і призначена для пошуку інформації за тим же принципом, що і в книзі з предметним покажчиком. Індекс дозволяє уникнути проведення послідовного або покрокового перегляду файлу в пошуках потрібних даних. При використанні індексів в базі даних шуканим об'єктом може бути одна або кілька записів файлу. Як і предметний покажчик книги, індекс бази даних впорядкований, і кожен елемент індексу містить назву шуканого об'єкта, а також один або кілька покажчиків (ідентифікаторів записів) на місце його розташування.

Хоча індекси, строго кажучи, не є обов'язковим компонентом СУБД, вони можуть істотно підвищити її продуктивність. Як і у випадку з предметним покажчиком книги, читач може знайти визначення даного його поняття, переглянувши всю книгу, але це зажадає занадто багато часу. А предметний покажчик, ключові слова в якому розташовані в алфавітному порядку, дозволяють відразу ж перейти на потрібну сторінку.

Структура індексу пов'язана з певним ключем пошуку і містить записи, що складаються з ключового значення і адреси логічного запису у файлі, що містить це ключове значення.Файл, що містить логічні записи, називається файлом даних, а файл, який містить індексні записи, - індексним файлом. Значення в індексному файлі впорядковані по полю індексування, яке зазвичай будується на базі одного атрибута.

Типи індексів

Для прискорення доступу до даних застосовується кілька типів індексів.

Первинний індекс - це такий спеціальний масив-покажчик порядку записів, коли файл даних послідовно впорядковується по полю ключа впорядкування, а на основі поля ключа впорядкування створюється поле індексації, яке гарантовано має унікальне значення в кожному записі.

Індекс кластеризації - це такий спеціальний масив-покажчик порядку записів, коли файл даних послідовно впорядковується за неключових полю, і на основі цього неключових поля формується поле індексації, тому в файлі може бути декілька записів, які відповідають значенням цього поля індексації. Неключових поле називається атрибутом кластеризації.

Вторинний індекс - це індекс, який визначено на поле файлу даних, відмінному від поля, по якому виконується впорядкування.

Файл може мати не більше одного первинного індексу або одного індексу кластеризації, але додатково до них може мати кілька вторинних індексів. Індекс може бути розрідженим (sparse) або щільним (dense). Розріджений індекс містить індексні записи тільки для деяких значень ключа пошуку в даному файлі, а щільний індекс має індексні записи для всіх значень ключа пошуку в даному файлі. Ключ пошуку для індексу може складатися з декількох полів.

Індексного-послідовні файли

Відсортований файл даних з первинним індексом називається індексованих послідовним файлом, або індексного-послідовним файлом. Ця структура є компромісом між файлами з повністю послідовної і повністю довільній організацією. У такому файлі записи можуть оброблятися як послідовно, так і вибірково, з довільним доступом, здійснюваним на основу пошуку по заданому значенню ключа з використанням індексу. Індексований послідовний файл має більш універсальну структуру, яка зазвичай включає наступні компоненти:

- первинна область зберігання;
- окремий індекс або кілька індексів;
- область переповнення.

Зазвичай велика частина первинного індексу може зберігатися в оперативній пам'яті, що дозволяє обробляти його набагато швидше. Для прискорення пошуку можуть застосовуватися

спеціальні методи доступу, наприклад метод бінарного пошуку. Основним недоліком використання первинного індексу (як і при роботі з будь-яким іншим відсортованим файлом) є необхідність дотримання послідовності сортування при вставці і видаленні записів. Ці проблеми ускладнюються тим, що потрібно підтримувати порядок сортування як у файлі даних, так і в індексному файлі. У подібному випадку може використовуватися метод, що полягає в застосуванні області переповнення і ланцюжки пов'язаних покажчиків, аналогічно методу, що використовується для вирішення конфліктів в хешірованних файлах.

вторинні індекси

Вторинний індекс також є впорядкованим файлом, аналогічним первинному індексу. Однак пов'язаний з первинним індексом файл даних завжди впорядкований по ключу цього індексу, тоді як файл даних, пов'язаний із вторинним індексом, не обов'язково повинен бути відсортований по ключу індексації. Крім того, ключ вторинного індексу може містити повторювані значення, що не допускається для значень ключа первинного індексу. Для роботи з такими повторюваними значеннями ключа вторинного індексу зазвичай використовуються перераховані нижче методи.

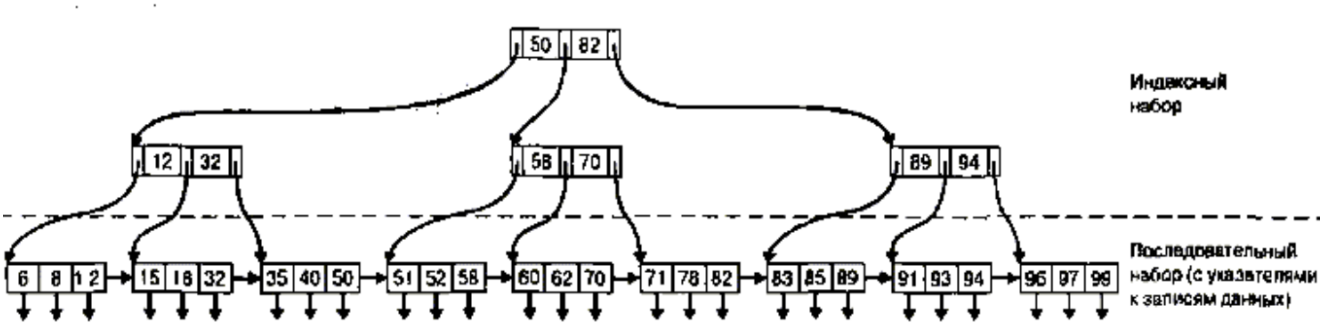
- Створення щільного вторинного індексу, який відповідає всіма записами файла даних, але при цьому в ньому допускається наявність дублікатів.
- Створення вторинного індексу зі значеннями для всіх унікальних значень ключа. При цьому покажчики блоків є багатозначними, оскільки кожне його значення відповідає одному з дублікатів ключа у файлі даних.
- Створення вторинного індексу зі значеннями для всіх унікальних значень ключа. Але при цьому покажчики блоків вказують не на файл даних, а на сегмент, який містить покажчики на відповідні записи файлу даних.

Вторинні індекси підвищують продуктивність обробки запитів, в яких для пошуку використовуються атрибути, відмінні від атрибута первинного ключа. Однак таке підвищення продуктивності запитів вимагає додаткової обробки, пов'язаної з супроводом індексів при оновленні інформації в базі даних. Це завдання вирішується на етапі фізичного проектування бази даних.

багаторівневі індекси

При зростанні розміру індексного файлу і розширенні його вмісту на велику кількість сторінок час пошуку потрібного індексу також значно зростає. Звернувшись до багаторівневого індексу, можна спробувати вирішити цю проблему шляхом скорочення діапазону пошуку. Дана операція виконується над індексом аналогічно тому, як це робиться в разі файлів іншого типу, тобто за допомогою розщеплення індексу на кілька субіндексів меншого розміру і створення індексу для цих субіндексів. На кожній сторінці файлу даних можуть зберігатися два записи. Крім того, в якості ілюстрації тут показано, що на кожній сторінці індексу також зберігаються дві індексні записи, але на практиці на кожній такій сторінці може зберігатися набагато більше індексних записів. Кожна індексна запис містить значення ключа доступу і адреса сторінки. Збережене значення ключа доступу є найбільшим на адресується сторінці.

Вдосконалені збалансовані деревовидні індекси



збалансоване дерево

дерево - це структура даних, яка використовується в багатьох СУБД для зберігання даних або індексів. Дерево складається з ієрархії вузлів (node), в якій кожен вузол, за винятком кореня (root), має батьківський (parent) вузол, а також один, кілька або жодного дочірнього (child) вузла. Корінь не має батьківського вузла. Вузол, який не має дочірніх вузлів, називається листом (leaf).

Глибина дерева - це максимальна кількість рівнів між коренем і листом. Глибина дерева може бути різною для різних шляхів доступу до листів.

Збалансоване дерево, B-дерево, B-Tree - це дерево, у якого глибина дерева однакова для всіх листів.

Ступінь (degree), порядок (order) дерева - це максимально допустима кількість дочірніх вузлів для кожного батьківського вузла. Великі ступеня зазвичай використовуються для створення більш широких і менш глибоких дерев.

Оскільки час доступу в структурі дерева залежить від глибини, а не від ширини, зазвичай прийнято використовувати більш "розгалужені" і менш глибокі дерева.

Бінарне дерево, binary tree - це дерево близько 2, в якому кожен вузол має не більше двох дочірніх вузлів.

Вдосконалені збалансовані деревовидні індекси визначаються за такими правилами.

- Якщо корінь не є лист-вузлом, то він повинен мати, принаймні, два дочірніх вузла.
- У дереві порядку n кожен вузол (за винятком кореня і листів) повинен мати від n / 2 до n покажчиків і дочірніх вузлів. Якщо число n / 2 не є цілим, то воно округлюється до найближчого більшого цілого.
- У дереві порядку n кількість значень ключа в листі повинно знаходитися в межах від (n-1) / 2 до (n-1). Якщо число (n-1) / 2 не є цілим, то воно округлюється до найближчого більшого цілого.
- Кількість значень ключа в нелістовом вузлі на одиницю менше кількості покажчиків.
- Дерево завжди має бути збалансованим, тобто всі шляхи від кореня до кожного листу повинні мати однакову глибину.
- Листи дерева пов'язані в порядку зростання значень ключа.

МЕТОДИЧНІ ВКАЗІВКИ
із самостійної роботи студентів

1. Тема: Захист інформації в Internet

2. Навчальна мета. Студент повинен:

знати: способи опису алгоритмів і їх типи алгоритмів.

уміти: складати алгоритми різних типів

3. Завдання для самопідготовки

Вивчити рекомендовану літературу, звернути увагу на такі питання:

1) Принципи захисту
2) Криптографія
3) Електронний цифровий підпис
4) Аутентифікація
5) Захист мереж

4. Результати самостійної позааудиторної діяльності, питання для самоконтролю

1) Чому необхідно захищувати інформацію в Інтернет
2) Засоби захисту

5.Контроль засвоєння теми

Різнорівневі запитання.

6. Рекомендована література

1. Конноли, Томас Бази даних. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание:Пер. с англ /Томас Конноли, Каролін Бег .–М.:Издательский дом ″Вильямс″, 2003.– 1440 с.: ил. .– ISBN 5-8459-0527-3 (рус)

2. Мамаев, Евгений. Microsoft SQL Server для профессионалов/ Евгений Мамаев, Лилия Шкарина.– СПб.: Питер, 2001.– 1088 с. ил.− ISBN 5-272-00339-X

3. https://translate.googleusercontent.com/translate_c?depth=1&hl=uk&prev=search&rurl=translate.google.com.ua&sl=ru&u=http://camafon.ru/informatsionnaya-bezopasnost/zashhita-v-internete&usg=ALkJrhjuDypm_zLhB_Y9t3Z65Xyd6xvcsw

Захист інформації в Інтернеті

Проведення фінансових операцій з використанням Інтернету, замовлення товарів і послуг, використання кредитних карток, доступ до закритих інформаційних ресурсів, передача телефонних розмов вимагають забезпечення відповідного рівня безпеки. Конфіденційна інформація, яка передається через мережу Інтернет, проходить через певну кількість маршрутизаторів і серверів, перш ніж досягне пункту призначення. Зазвичай маршрутизатори не відслідковують проходять крізь них потоки інформації, але можливість того, що інформація може бути перехоплена, існує. Більш того, інформація може бути змінена і передана адресату в зміненому вигляді. На жаль, сама архітектура мережі Інтернет завжди залишає можливість для несумлінного користувача здійснити подібні дії.

Завжди існує проблема вибору між необхідним рівнем захисту і ефективністю роботи в мережі. У деяких випадках користувачами або споживачами заходи щодо забезпечення безпеки можуть бути розцінені як заходи з обмеження доступу та ефективності. Однак такі засоби, як, наприклад, криптографія, дозволяють значно посилити ступінь захисту, не обмежуючи доступ користувачів до даних.

Принципи захисту інформації

Проблеми, що виникають з безпекою передачі інформації при роботі в комп'ютерних мережах, можна розділити на чотири основні типи:

- перехоплення інформації - цілісність інформації зберігається, але її конфіденційність порушена;
- модифікація інформації - вихідне повідомлення змінюється або повністю підміняється іншим і відсилається адресату;
- підміна авторства інформації;
- перехоплення повідомлення з його вилученням.

Дана проблема може мати серйозні наслідки. Наприклад, хтось може надіслати листа від вашого імені (цей вид обману прийнято називати Спуфінга) або Web-сервер може прикидатися електронним магазином, приймати замовлення, номери кредитних карт, але не висилати ніяких товарів.

У відповідності з перерахованими проблемами при обговоренні питань безпеки під самим терміном "безпека" мається на увазі сукупність трьох різних характеристик забезпечує безпеку системи:

–Аутентифікація - це процес розпізнавання користувача системи і надання йому певних прав і повноважень. Кожен раз, коли заходить мова про ступінь або як аутентифікації, під цим слід розуміти ступінь захищеності системи від зазіхань сторонніх осіб на ці повноваження.

–Цілісність - стан даних, при якому вони зберігають свій інформаційний зміст і однозначність інтерпретації в умовах різних впливів. Зокрема, в разі передачі даних під цілісністю розуміється ідентичність відправленого і прийнятого.

–Секретність - запобігання несанкціонованому доступу до інформації. У разі передачі даних під цим терміном звичайно розуміють запобігання перехоплення інформації.

Криптографія

Для забезпечення секретності застосовується шифрування, або криптографія, що дозволяє трансформувати дані в зашифровану форму, з якої витягти вихідну інформацію можна тільки при наявності ключа. В основі шифрування лежать два основних поняття: алгоритм і ключ.

Алгоритм - це спосіб закодувати вихідний текст, в результаті чого виходить зашифроване послання. Зашифроване послання може бути інтерпретовано тільки за допомогою ключа. Очевидно, щоб зашифрувати послання, досить алгоритму. Однак використання ключа для шифрування надає два істотних переваги. По-перше, можна використовувати один алгоритм з різними ключами для відправки послань різним адресатам. По-друге, якщо секретність ключа буде порушена, його можна легко замінити, не змінюючи при цьому алгоритм шифрування.

Таким чином, безпеку систем шифрування залежить від секретності використовуваного ключа, а не від секретності алгоритму шифрування. Багато алгоритмів шифрування є загальнодоступними. Кількість можливих ключів для даного алгоритму залежить від числа біт в ключі. наприклад, 8-бітний ключ допускає 256 (28) комбінацій ключів. Чим більше можливих комбінацій ключів, тим важче підібрати ключ, тим надійніше зашифровано послання. Так, наприклад, якщо використовувати 128-бітний ключ, то необхідно буде перебрати $^{2}128 \sim = 10^{40}$ ключів, що в даний час не під силу навіть найпотужнішим комп'ютерів.

Важливо відзначити, що зростаюча продуктивність техніки призводить до зменшення часу, потрібного для розтину ключів, і системам забезпечення безпеки доводиться використовувати дедалі довші ключі, що, в свою чергу, веде до збільшення витрат на шифрування.

Оскільки таке важливе місце в системах шифрування приділяється секретності ключа, то основною проблемою подібних систем є генерація і передача ключа. Існують дві основні схеми шифрування: симетричне шифрування (його також іноді називають традиційним або шифруванням з секретним ключем) і шифрування з відкритим ключем (іноді цей тип шифрування називають асиметричним).

При симетричному шифруванні відправник і одержувач володіють одним і тим же ключем (секретним), за допомогою якого вони можуть зашифровувати і розшифровувати дані. При симетричному шифруванні використовуються ключі невеликої довжини, тому можна швидко шифрувати великі обсяги даних. Симетричне шифрування використовується, наприклад, деякими банками в мережах банкоматів. Однак симетричне шифрування володіє декількома недоліками. По-перше, дуже складно знайти безпечний механізм, за допомогою якого відправник і одержувач зможуть таємно від інших вибрати ключ. Виникає проблема безпечного поширення секретних ключів. По-друге, для кожного адресата необхідно зберігати окремий секретний ключ. По-третє, в схемі симетричного шифрування неможливо гарантувати особу відправника, оскільки два користувача володіють одним ключем.

У схемі шифрування з відкритим ключем для шифрування послання використовуються два різних ключа. За допомогою одного з них послання зашифрована, а за допомогою другого - розшифровується. Таким чином, необхідної безпеки можна домогтися, зробивши перший ключ загальнодоступним (відкритим), а другий ключ зберігати тільки в одержувача (закритий, особистий ключ). У такому випадку будь-який користувач може зашифрувати послання за допомогою відкритого ключа, але розшифрувати послання здатний тільки володар особистого ключа. При цьому немає необхідності піклуватися про безпеку передачі відкритого ключа, а для того щоб користувачі могли обмінюватися секретними повідомленнями, досить наявності у них відкритих ключів один одного. Недоліком асиметричного шифрування є необхідність використання більш довгих, ніж при симетричному шифруванні, ключів для забезпечення еквівалентного рівня безпеки, що позначається на обчислювальних ресурсах, необхідних для організації процесу шифрування.

Електронний цифровий підпис

Навіть якщо послання, безпеку якого ми хочемо забезпечити, належним чином зашифровано, все одно залишається можливість модифікації вихідного повідомлення або підміни цього повідомлення іншим. Одним із шляхів вирішення цієї проблеми є передача користувачем одержувачу короткого представлення переданого повідомлення. Подібне короткий уявлення називають ***контрольною сумою,*** або ***дайджестом*** повідомлення.

Контрольні суми використовуються при створенні резюме фіксованої довжини для подання довгих повідомлень. Алгоритми розрахунку контрольних сум розроблені так, щоб вони були по можливості унікальні для кожного повідомлення. Таким чином, усувається можливість підміни одного повідомлення іншим зі збереженням того ж самого значення контрольної суми. Однак при використанні контрольних сум виникає проблема передачі їх одержувачу. Одним з можливих шляхів її вирішення є включення контрольної суми в так званий електронний підпис.

За допомогою електронного підпису одержувач може переконатися в тому, що отримане їм повідомлення надіслано НЕ сторонньою особою, а мають певні права відправником. Електронні цифрові підписи створюються шифруванням контрольної суми і додаткової інформації за допомогою особистого ключа відправника. Таким чином, будь-хто може розшифрувати підпис, використовуючи відкритий ключ, але коректно створити підпис може тільки власник особистого ключа. Для захисту від перехоплення і повторного використання підпис включає в себе унікальне число - порядковий номер.

Аутентифікація

Аутентифікація є одним з найважливіших компонентів організації захисту інформації в мережі. Перш ніж користувачеві буде надано право отримати той чи інший ресурс, необхідно переконатися, що він дійсно той, за кого себе видає.

При отриманні запиту на використання ресурсу від імені будь-якого користувача сервер, що надає даний ресурс, передає управління серверу аутентифікації. Після отримання позитивної відповіді сервера аутентифікації користувачеві надається запитуваний ресурс.

При аутентифікації використовується, як правило, принцип, який отримав назву "що він знає", - користувач знає деяке секретне слово, яке він посилає серверу аутентифікації у відповідь на його запит. Однією зі схем аутентифікації є використання стандартних паролів. Ця схема є найбільш вразливою з точки зору безпеки - пароль може бути перехоплений і використаний іншою особою. Найчастіше використовуються схеми з застосуванням одноразових паролів. Навіть будучи перехоплених, цей пароль буде марний при наступній реєстрації, а отримати наступний пароль з попереднього є вкрай важким завданням. Для генерації одноразових паролів використовуються як програмні, так і апаратні генератори, що представляють собою пристрої, що вставляються в слот комп'ютера. Знання секретного слова необхідно користувачу для приведення цього пристрою в дію. Однією з найбільш простих систем, які не потребують додаткових витрат на обладнання, але в той же час забезпечують хороший рівень захисту, є S / Key, на прикладі якої можна продемонструвати порядок подання одноразових паролів.

У процесі аутентифікації з використанням S / Key беруть участь дві сторони - клієнт і сервер. При реєстрації в системі, що використовує схему аутентифікації S / Key, сервер надсилає на клієнтську машину запрошення, що містить зерно, передане по мережі у відкритому вигляді, поточне значення лічильника ітерацій і запит на введення одноразового пароля, який повинен відповідати цьому значенню лічильника ітерації. Отримавши відповідь, сервер перевіряє його і передає управління серверу необхідного користувачем сервісу.

Захист мереж

Останнім часом корпоративні мережі все частіше включаються в Інтернет або навіть використовують його в якості своєї основи. З огляду на те, якої шкоди може принести незаконне вторгнення в корпоративну мережу, необхідно виробити методи захисту. Для захисту корпоративних інформаційних мереж використовуються брандмауери.

Брандмауер - це система або комбінація систем, що дозволяють розділити мережу на дві або більше частин і реалізувати набір правил, що визначають умови проходження пакетів з однієї частини в іншу. Як правило, ця межа проводиться між локальною мережею підприємства та Інтернетом, хоча її можна провести і всередині. Однак захищати окремі комп'ютери невигідно, тому зазвичай захищають всю мережу. Брандмауер пропускає через себе весь трафік і для кожного пакету приймає рішення - пропускати його або відкинути. Для того щоб брандмауер міг приймати ці рішення, для нього визначається набір правил.

Брандмауер може бути реалізований як апаратними засобами (тобто як окремий фізична пристрій), так і у вигляді спеціальної програми, запущеної на комп'ютері. Як правило, в операційну систему, під керуванням якої працює брандмауер, вносяться зміни, мета яких - підвищення захисту самого брандмауера. Ці зміни зачіпають як ядро ОС, так і відповідні файли конфігурації. На самому брандмауері не дозволено мати розділів користувачів, а отже, і потенційних дірок - тільки розділ адміністратора. Деякі брандмауери працюють тільки в режимі одного, а багато хто має систему перевірки цілісності програмних кодів.

Брандмауер зазвичай складається з декількох різних компонентів, включаючи фільтри або екрани, які блокують передачу частини трафіку. Всі брандмауери можна розділити на два типи: пакетні фільтри, які здійснюють фільтрацію IP-пакетів засобами фільтруючих маршрутизаторів; сервери прикладного рівня, які блокують доступ до певних сервісів в мережі. Таким чином, брандмауер можна визначити як набір компонентів або систему, яка розташовується між двома мережами і має такі властивості:

- Весь трафік з внутрішньої мережі в зовнішню і з зовнішньої мережі у внутрішню повинен пройти через цю систему;

- Тільки трафік, певний локальної стратегією захисту, може пройти через цю систему;

У такому випадку система надійно захищена від проникнення.

Кардинальним вирішенням захисту локальної мережі є її повна (фізична) ізоляція від інших мереж.

Підводні камені захисту інформації в інтернеті

Коли ви передаєте в мережах якісь дані, то, перш ніж вони досягнуть адресата, їм потрібно пройти через безліч інших серверів і маршрутизаторів. Ніякого відстеження або контролю шляхом при цьому немає. Тому з інформацією може статися що завгодно. Інтернет за своєю архітектурою дає можливість зловмисникам повну свободу дій.

Система інтернет зароджувалася як абсолютно корпоративна. Але вийшло так, що з часом вона стала оперувати не тільки мережами освітнього, комерційного, державного, відомчого, військового характеру, що самі по собі мають на увазі якийсь обмежений доступ, а й простими користувачами, які легко одержують прямий доступ в інтернет з будь-якого домашнього комп'ютера з допомогою звичайного модему і телефонної мережі загального користування. При цьому використовуються єдиний стек протоколів TCP / IP і єдиний адресний простір.

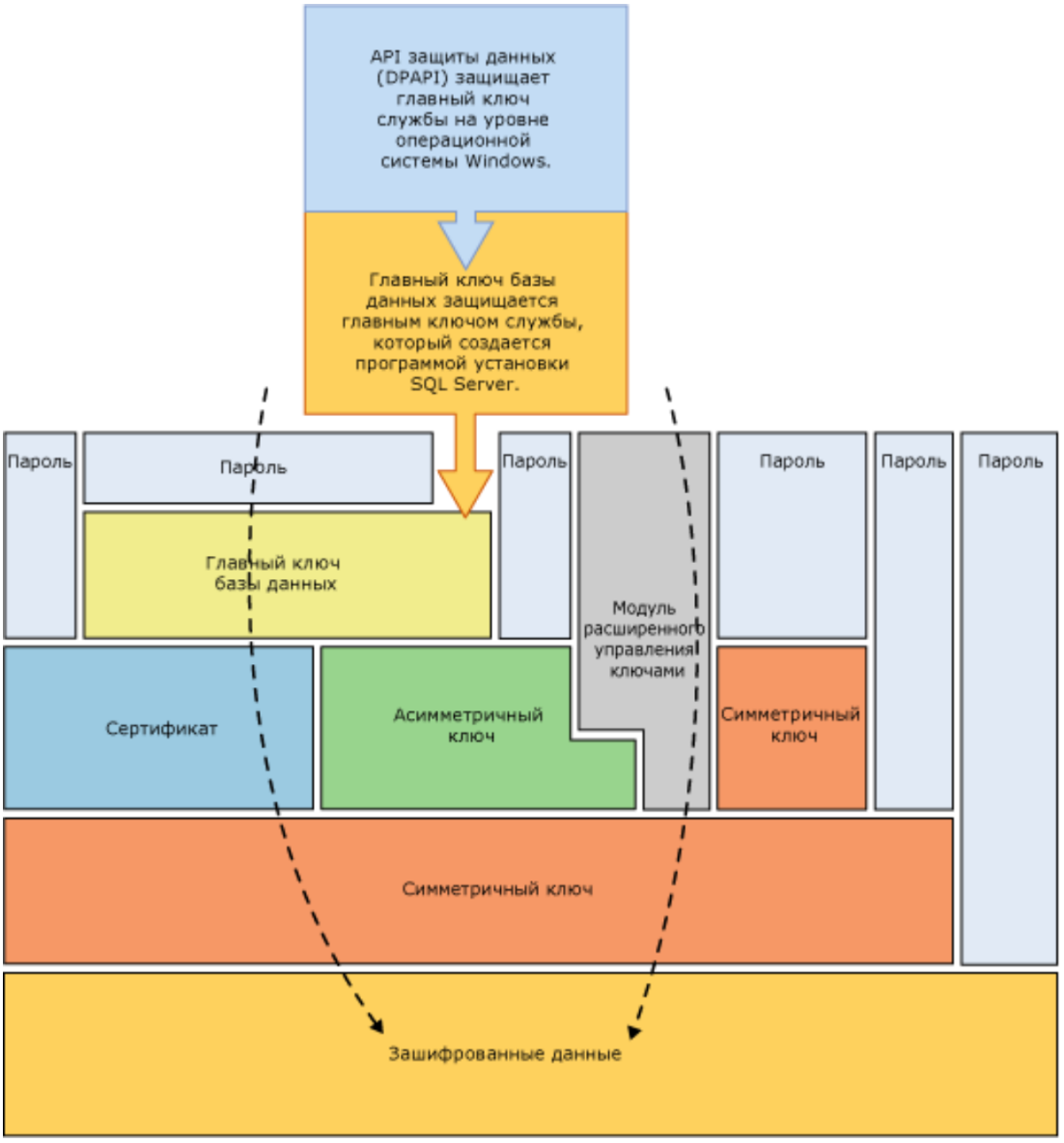
Така простота доступу в інтернет негативним чином позначається на безпеці інформації в мережах. А найгірше те, що ви можете навіть і не дізнатися, що ваші файли або програми були скопійовані, не кажучи вже про можливість їхнього псування і коректування.

Захист інформації в інтернеті є однією з головних проблем будь-якої великої компанії або організації. Ті, хто більш передбачливий, планують, як повинна здійснюється захист завчасно. Решта замислюються про це вже після першої неприємності.

Ієрархія засобів шифрування

SQL Server шифрує дані, використовуючи ієрархічну структуру коштів шифрування і управління ключамі.На кожному рівні дані нижчого рівня шифруються на основі комбінації сертифікатів, асиметричних ключів і симетричних ключей.Асімметрічние і симетричні ключі можна зберігати поза модуля розширеного управління ключами SQL Server.

На наступному малюнку показано, що на кожному рівні ієрархії засобів шифрування шифруються дані більш нижнього рівня і відображаються найбільш поширені конфігурації шіфрованія.Доступ до початку ієрархії, як правило, захищається паролем.



Слід враховувати наступні основні поняття.

• Для кращої продуктивності дані слід шифрувати за допомогою симетричних ключів, а не за допомогою сертифікатів та асиметричних ключів.

• Головні ключі бази даних захищені головним ключем служби.Главний ключ служби створюється при установці SQL Server і шифрується API-інтерфейсом захисту даних Windows (DPAPI).

• Можливі інші ієрархії шифрування з додатковими рівнями.

• Модуль розширеного управління ключами зберігає симетричні або асиметричні ключі поза SQL Server.

• Прозоре шифрування даних (TDE) має використовувати симетричний ключ, який називається ключем шифрування бази даних, захищений сертифікатом, який, в свою чергу захищається головним ключем бази даних master або асиметричним ключем, що зберігається в модулі розширеного керування ключами.

• Головний ключ служби і всі головні ключі бази даних є симетричними ключами.

На наступному малюнку показані ті ж відомості іншим чином.

Діаграма ілюструє такі додаткові основні поняття:
•На цьому малюнку стрілки вказують стандартні ієрархії шифрування.
• Симетричні і асиметричні ключі в модулі розширеного керування ключами захищають доступ до симетричним і асиметричним ключам, що зберігаються в SQL Server.Пунктірная лінія, пов'язана з модулем розширеного управління ключами, вказує, що ключі в модулі розширеного керування ключами можуть замінювати симетричні і асиметричні ключі, що зберігаються в SQL Server.

Механізми шифрування

SQL Server підтримує наступні механізми шифрування:
• функції Transact-SQL;

• асиметричні ключі;
• симетричні ключі;
• Сертифікати
• Прозоре шифрування даних

Функції Transact-SQL

Окремі елементи можна шифрувати по мірі того, як вони вставляються або оновлюються, за допомогою функцій Transact-SQL ..

Сертифікати

Сертифікат відкритого ключа, або просто сертифікат, являє собою підписану цифровим підписом інструкцію, яка пов'язує значення відкритого ключа з ідентифікатором користувача, пристрою або служби, що має відповідний закритий ключ.Сертіфікати поставляються і підписуються центром сертифікації (certification authority, CA) .Сущность, яка отримує сертифікат від центру сертифікації, є суб'єктом цього сертіфіката.Как правило, сертифікати містять такі відомості.

• Відкритий ключ суб'єкта.
• Ідентифікаційні дані суб'єкта, наприклад ім'я та адресу електронної пошти.
• Термін дії, тобто інтервал часу, протягом якого сертифікат буде вважатися дійсним.

Сертифікат дійсний тільки протягом зазначеного в ньому періоду, який задається в кожному сертифікаті за допомогою дат (Valid From і Valid To), що визначають початок і закінчення терміну действія.По закінчення терміну дії сертифіката його суб'єкт повинен запросити новий сертифікат.

• Ідентифікаційні дані постачальника сертифіката.
• Використання цифрового підпису може постачальника.

Цей підпис підтверджує дійсність зв'язку між відкритим ключем і ідентифікаційними даними суб'єкта. (В процесі створення цифрового підпису дані, разом з деякими секретними даними відправника, перетворюються в тег, званий підписом.)

Головна перевага сертифікатів в тому, що вони дають змогу не зберігати на вузлах сукупність паролів окремих суб'єктов.Вместо цього вузол просто встановлює довірчі відносини з постачальником сертифіката; після цього постачальник може підписати необмежену кількість сертифікатів.

Коли вузол (наприклад, захищений веб-сервер) вказує, що конкретний постачальник сертифіката є довіреною кореневих центром сертифікації, він неявно висловлює довіру до політиків, які постачальник використовував при визначенні зв'язків для виданих їм сертіфікатов.Такім чином, вузол висловлює впевненість в тому, що постачальник перевірив ідентифікаційні дані суб'єкта сертіфіката.Узел робить постачальника довіреною кореневих центром сертифікації, поміщаючи самостійно підписаний сертифікат постачальника, що містить відкритий ключ постачальника, в сховище сертифікатів довіреної кореневого центру сертифікації на головному компьютере.Промежуточние або підлеглі центри сертифікації є довіреними тільки в тому випадку, якщо до них веде правильний шлях від довіреної кореневого центру сертифікації.

Постачальник може відкликати сертифікат до закінчення терміну його действія.Прі цьому скасовується зв'язок відкритого ключа з ідентифікаційними даними, зазначеними в сертіфікате.Каждий постачальник веде список відкликаних сертифікатів, який можна використовувати для перевірки конкретного сертифіката.

Асиметричні ключі

Асиметричний ключ складається з закритого ключа та відповідного відкритого ключа.Каждий з цих ключів дозволяє дешифрувати дані, зашифровані іншим ключем.На виконання асиметричних операцій шифрування і дешифрування потрібно порівняно багато ресурсів, але вони забезпечують більш надійний захист, ніж симетричне шіфрованіе.Асімметрічний ключ можна використовувати для шифрування симетричного ключа перед його збереженням в базі даних.

Симетричні ключі

Симетричний ключ - це ключ, який використовується і для шифрування, і для дешифрування данних.Данние при використанні симетричного ключа шифруються і дешифруються швидко, і він цілком підходить для повсякденного захисту конфіденційних даних, що зберігаються в базі даних.

Прозоре шифрування даних

Прозоре шифрування даних (TDE) є особливим випадком шифрування з використанням симетричного ключа.TDE шифрує всю базу даних, використовуючи симетричний ключ, який називається ключем шифрування бази данних.Ключ шифрування бази даних захищена іншими ключами або сертифікатами, які, в свою чергу, захищаються головним ключем бази даних або асиметричним ключем, що зберігається в модулі розширеного керування ключами.

Організація баз даних та знань»[Текст] Методичні вказівки до самостійної роботи для здобувачів освітньо-кваліфікаційного ступеня фаховий молодший бакалавр спеціальності 122 Комп’ютерні науки денної форми здобуття освіти /уклад. Олександр ПРИСАДА. – Ковель: ВСП «КПЕФК ЛНТУ», 2025. – 115с.

Комп’ютерний набір і верстка: **Олександр ПРИСАДА**

Редактор: **Леся ПРОКОПЧУК**

Підп. до друку «____»_____2025. Формат 60х84/16. папірофіс.
Гарн.Таймс. Ум.друк. арк. ____Обл.-вид. арк. ___Зам. ___
Тираж ____прим.

ВСП «КПЕФК ЛНТУ»
45000 м. Ковель, вул. Заводська, 23
Друк – ВСП «КПЕФК ЛНТУ»