ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ
СПЕЦИАЛИЗИРАНО ЖУРИ ПО ЕЛЕКТРОННА И КОМПЮТЪРНА ТЕХНИКА

инж. Луей Насер Махмуд Немрауи

ПОДХОД ЗА МОДЕЛИРАНЕ НА ОБЕКТНИ БАЗИ ДАННИ

АВТОРЕФЕРАТ

НА ДИСЕРТАЦИОНЕН ТРУД
ЗА ПОЛУЧАВАНЕ НА ОБРАЗОВАТЕЛНАТА И НАУЧНА СТЕПЕН "**ДОКТОР**"

Научна специалност: „Комютърни системи, комплекси и мрежи"

София, 2012 г.

ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ
СПЕЦИАЛИЗИРАНО ЖУРИ ПО ЕЛЕКТРОННА И КОМПЮТЪРНА ТЕХНИКА

инж. Луей Насер Махмуд Немрауи

ПОДХОД ЗА МОДЕЛИРАНЕ НА ОБЕКТНИ БАЗИ ДАННИ

АВТОРЕФЕРАТ

НА ДИСЕРТАЦИОНЕН ТРУД
ЗА ПОЛУЧАВАНЕ НА ОБРАЗОВАТЕЛНАТА И НАУЧНА СТЕПЕН "**ДОКТОР**"

Научна специалност: „Комютърни системи, комплекси и мрежи"

Научен ръководител:
доц. д-р инж. Даниела Гоцева

Рецензенти:

София, 2012 г.

Дисертационният труд се състои от четири глави. Текстът е написан на 125 страници и съдържа 20 фигури и 8 таблици. Цитирани са 102 литературни източника и Internet страници. Номерата на фигурите, таблиците и уравненията в автореферата съвпадат с тези от дисертацията.

Дисертационният труд е обсъден и насочен за защита на разширено заседание на катедра „Компютърни системи" към Факултет по компютърни системи и управление на Технически университет – София, състояло се на 14 май 2012 год.

Защитата на дисертационния труд ще се състои на ..... г. от ..... часа в зала 1434 на I блок на Технически Университет–София на разширено заседание на .....

Материалите по защитата са на разположение на интересуващите се в канцеларията на Факултета по „Комютърни системи и управление", стая 1443а, I блок на Технически Университет-София.

Автор: Луей Махмуд Насер Нимрауи
Заглавие: Подход за моделиране на обектни бази данни.

ОБЩА ХАРАКТЕРИСТИКА НА ДИСЕРТАЦИОННИЯ ТРУД

Цел и задачи на изследването

Целта на дисертационният труд е да се създаде подход за обектно моделиране в бази от данни. За изпълнение на тази цел са дефинирани следните задачи:

1. Да бъде изследвано обектно-ориентираното моделиране на данни и концепциите за обектно-ориентирани бази от данни.
2. Да се проучат основите на обектно-релационната алгебра, обектно-релационния SQL, обектно-релационните системи за управление на бази от данни (ОРСУБД) и обектно-ориентираните СУБД. Да се направи сравнителен анализ на ОРСУБД и ООСУБД.
3. Да бъде изследвана производителността на реализараните връзки в ОРСУБД Oracle и да се предложи подход за дизайн на бази от данни с ОРСУБД и ООСУБД. Да се реализира примерно приложение с бази от данни чрез ОРСУБД и релационна СУБД, за да се изучи производителността на дизайна на бази от данни.

Структура на дисертационния труд

За да реши поставените задачи, дисертационният труд е разделен на 3 глави, справка за приносите и библиография. Всяка глава има въведение в конкретната проблематика, описание на решението на поставената задача и обобщение, в което са дадени направените изводи и препоръки за по-нататъшна работа. В края на дисертационния труд са дадени научните и научно-приложните приноси. Текстът на дисертацията е написан на 125 страници и съдържа 20 фигури и 8 таблици. Цитирани са 102 литературни източника и Internet страници. Номерата на фигурите и таблиците в автореферата съвпадат с тези от дисертацията. Ето и конкретното съдържание на всяка от главите:

1. ОСНОВИ НА ОБЕКТНО-ОРИЕНТИРАНИТЕ БАЗИ ОТ ДАННИ

Главата се състои от две части. В първата част „Object Oriented Data Modeling“ са описани основите на обектно-ориентираното бази от данни и моделирането им.

Във втората част „Object Oriented Database Concepts“ са описани основните понятия като паралелизми, транзакции, защита, които се използват в обектно-ориентираните бази от данни.

2. ОБЕКТНО-РЕЛАЦИОННИ БАЗИ ОТ ДАННИ

Втора глава се състои от три части. В първата част „Object Relational Algebra“ са разгледани основите на обектно-релационната алгебра.

Представени са основните операции, които се използват в обектно-релационните бази дани.

Във втората част „Object Relational SQL“ са представени операциите на обектно-релационния SQL, както и допълнителните операции в обектните бази данни, разработени от докторанта.

В третата част „Object Relational SQL3“ е представена ORDBMS и е показана сравнителна таблица, която съдържа основните характеристики на ORDBMS и OODBMS.

3. ПРОЕКТИРАНЕ НА БАЗИ ОТ ДАННИ С ОБЕКТНО МОДЕЛИРАНЕ

Главата се състои от две части. В първата част „Relationships Implementation Performance in ORDBMS“ е представено изследване на производителността на връзките в ORDBMS. Разгледани са наследяване, асоциация, агрегация и композиция в Oracle 10g. С цел сравнение е създадена обектно-релационна и релационна реализация. Направени са необходимите тестове и изводи.

Във втората част “Database Design and Implementation with ORDBMS” е показано моделиране на реална система собектно-релационнен подход в Oracle 10g. С цел сравнение на реализациите е създаден и изцяло релационен модел и негова реализация.

Публикации

Резултатите по дисертацията са публикувани в списание “Communication and Computer Engineering”, ТУ-София, в трудовете на международните конференции Computer Science’2011”, International Conference “Challenges in High Education and Research in 21st Centuary”. Списък на публикациите по дисертацията е даден в края на автореферата.

КРАТКО СЪДЪРЖАНИЕ НА ДИСЕРТАЦИОННИЯ ТРУД

Глава 1. Основи на обектно-ориентираните бази от данни

1. Обектно-ориентирано моделиране на данни

1.1 Въведение

Тази част разяснява някои базови концепции като обектно-ориентирано програмиране, обекти, системи с бази от данни и обектно-ориентирани бази от данни. Пояснени са и основите на обектно-ориентирано моделиране като сложни обекти, идентичност на обекти, класове, атрибути, поведение, инкапсулация, наследяване, припокриване, късно свързване и именоване.

1.2 Обектно-ориентирано програмиране

Обектно-ориентираното програмиране (ООР) е модел на програмен език, организиран повече около „обекти“, отколкото около „действия“. ООП е метод на създаване на програмни приложения от група програмни компоненти, наречени обекти като самото приложение се състои от съобщения, които се изпращат между тези обекти. ООП се различава от процедурното програмиране. В процедурното програмиране данните и функциите, които ги обработват се разделят помежду си, защото повечето данни са достъпни на множество функции. Това прави тестването сложно. Обикновено, когато приложението стане голямо, процедурния модел започва да пропада. В противовес при ООП данните и функциите (наречени методи в ООП) се съдържат в обекти и са по защитени, което позволява приложението да нараства почти без проблем. Обектите моделират и обекти от реалния свят по акуратно, така че концептуално дори сложни проблеми могат да се улеснят като програмна реализация.

1.3 Обект

Обектите са ключ към разбирането на обектно-ориентирата технология. Самите те съдържат програмни компоненти, които изграждат обектно-ориентирани приложения, тъй като обектите съдържат данни (означени обикновено като атрибути), методите, чрез които се работи с данните и факта, че програмните обекти се използват често за моделиране на обектни от реалния свят.

Друг термин, свързан с обекта е класа, който може да се счита като обобщение на обекта.

1.4 Система за бази от данни

Една система за бази от данни е компютъризирана система за запазване на записи. Базата данни от своя страна може да бъде електронен архив, т.е. тя е хранилище или контейнер на колекция компютъризирани файлове с данни [5]. Потребителите на системата могат да изпълняват различни операции върху тези файлове, като например:

- Добавяне на нови, празни файлове в базата от данни.
- Вмъкване на данни в съществуващи файлове.
- Извличане на данни от съществуващи файлове.
- Промяна на данни в съществуващи файлове.
- Изтриване на данни от съществуващи файлове.
- Изтриване на съществуващи файлове от базата от данни.

Така че базовото определение на система от бази данни е система за съхранение на компютъризирани записи, т.е. тя е компютъризирана система, чиято обща цел е да съхрани информацията и да позволи на потребителите да извличат и актуализират информация при поискване.

1.5 Обектно-ориентирани бази от данни

Обектно-ориентираните бази от данни се наричат още Системи за управление на обектни бази от данни (ODBMS). Технологията на обектно-ориентирани бази от данни се състои от обектно-ориентирано моделиране и технологии за бази от данни. Фигура 1.2 илюстрира как тези концепции за програмиране и бази от данни могат да се ползват съвместно за да се осигури това, което ние наричаме обектно-ориентирани бази от данни.

Една обектно-ориентирана база от данни е съвкупност от обектно-ориентирани модели и технологии за бази от данни	
Обектно-ориентирано моделиране	**Технологии за бази от данни**
Идентичност на обекта **Инкапсулация** **Припокриване** **Поведения** **Именоване** **Атрибути** **Наследяване** **Класове** **Сложни обекти**	**Транзакции** **Сигурност** **Цялост** **Запис на информацията на диска** **Заявка** **Възстановяване** **Паралелно изпълнение**

Фигура 1.2 Състав на обектно-ориентираните бази от данни

1.6 Основно обектно-ориентирано моделиране

Една обектно-ориентирана база от данни трябва да удовлетворява два стандарта: трябва да е система за управление на бази от данни (СУБД) и трябва да е обектно-ориентирана система, т.е. доколкото е възмжно тя трябва да е в съответствие с текущите тенденции в обектно-ориентираните езици за програмиране. Първият стандарт има десет характеристики:

- Атрибути, идентичност на обекта, обекти, класове, типове поведения, инкапсулация, припокриване и късно свързване, наследяване и именоване.

Вторият стандарт се свежда до няколко характеристики:

- Цялост, управление на паралелизма, възстановяване, транзакции, запис на информацията на диска и сигурност.

2. *Концепции на обектно-ориентираните бази от данни*

2.1 Въведение

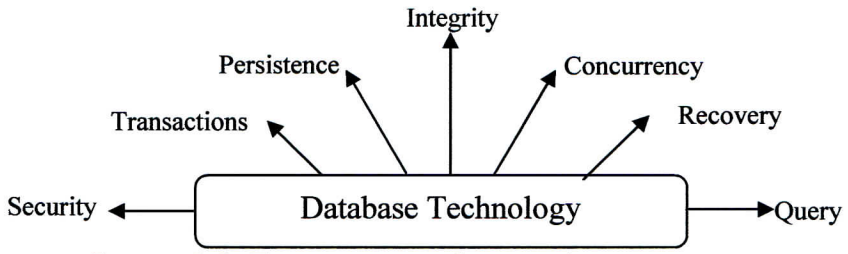
Този раздел обяснява как вторият стандарт се покрива от характеристики като цялост, управление на паралелизма, възстановяване, транзакции, запис на информацията на диска и сигурност.

Той обяснява и системите с обектно-ориентираните бази от данни (ООБД), двете архитектури на обектно-ориентирани системи за управление на бази от данни (ООСУБД), архитектурата клиент-сървър и съхранението на методи. Накрая е направено сравнение между обектно-релационни бази от данни (ОРСУБД) и ООСУБД.

2.2 Технология за бази от данни

Вторият стандарт от системите с обектно-ориентирани бази от данни има следните характеристики: Цялост, управление на паралелизма, възстановяване, транзакции, запис на информацията на диска и сигурност .

Фигура 2.1 илюстрира технологиите за бази от данни като Persistence, Concurrency, Recovery, Transactions, Integrity и други.



Фигура 2.1 Технологии за бази от данни

2.3 Перспективи на системи с ООБД

Системите с бази от данни са основния избор при създаване и поддръжка на големи дълготрайни колекции от данни. Модерните системи с бази от данни се характеризират с поддържането на следните характеристики:

- Модел на данните: начин за описание на данните, връзките между тях и ограниченията върху тях.
- Постоянство на данните: способността на данните да надживеят изпълнението на програмата, а вероятно и самата програма/
- Споделяне на данни: способността множество приложения (или инстанции на една и съща програма) да достъпят общи данни, едновременно.

- Надеждност: уверение, че данните в базата данни са защитени от хардуерни и софтуерни грешки.
- Мащабируемост: способността да се обработват огромно количество данни по лесни начини.
- Сигурност и цялост: защита на данните срещу неоторизиран достъп и уверение, че данните са в съответствие с правилата за коректност и съгласуваност.
- Разпространение: способността за физическо разпространение на логически взаимосвързани колекции от споделени данни в компютърната мрежа, като за предпочитане разпространението е да се направи прозрачно за потребителя.

За разлика от базите данни, традиционните езици за програмиране предоставят конструкции за процедурен контрол и за абстракции на ниво данни и функции, но не разполагат с вградена поддръжка за повечето от посочените по-горе характеристики. Докато всеки е полезен в своята област, съществува нарастващо множество приложения, които изискват функционалността на двете: и базите от данни и езиците за програмиране. Такива приложения се характеризират с нуждата им да съхраняват и извличат големи обеми споделени структурирани данни.

2.4 Архитектура

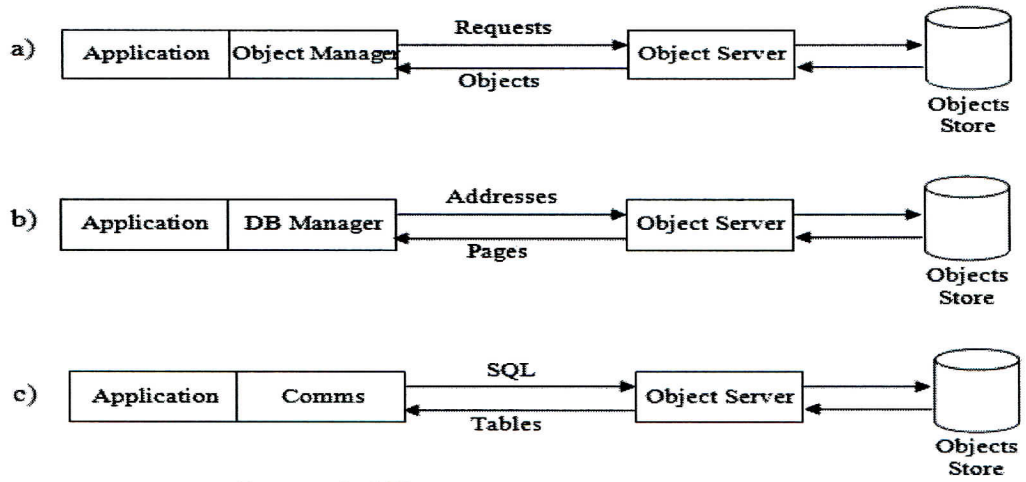
2.4.1 Клиент-сървър

Много комерсиални ООСУБД са базирани на клиент-сървър архитектури за предоставяне на данни на потребители, приложения и помощни програми за разпределени среди. Не всички системи, обаче, използва един и същ клиент-сървър модел. Съществуват три основни архитектури за клиент-сървър СУБД, които се различават по функционалността и компонентите си, както е показано на фигура 2.4.

Във всеки случай, сървърът се намиран на една и съща машина като физическа база от данни (БД). Клиентът може да се намира на същата или на различна машина. Ако клиентът се нуждае от достъп до бази от данни, разпределени на множество машини, то тогава той комуникира със сървъра на всяка от машините. Могат да съществуват и брой клиенти, комуникиращи с един сървър: например, един клиент за всеки потребител или приложение [3, 8].

2.4.2 Съхранение и изпълнение на методи

Съществуват два подхода за поддръжка на методи:



Фигура 2.4 Клиент-сървър архитектури:
(a) обектен сървър (b) страничен сървър (c) сървър на БД.

- Съхранение на методите във външни файлове, както е показано на фигура 2.4 (a).
- Съхранение на методите в базата от данни, както е показано на фигура 2.4 (b).

Първият подход е подобен на библиотека от функции или приложни програмни интерфейси (API). Той се среща в традиционните СУБД-та, в които производителя на СУБД предоставя приложния програмен интерфейс заедно със СУБД. С втория подход, методите се съхраняват в базата от данни и се свързват динамично с приложението по време на изпълнение на кода.

2.5 Сравнение между ОРСУБД и ООСУБД

Характеристика	**ОРСУБД**	**ООСУБД**
Ограничения върху целостта	Поддържа се напълно	Не се поддържа
Транзакции	Не се поддържа	Поддържа се, но степента е различна при различните продукти
Възстановяване	Поддържа се напълно	Поддържа се, но степента е различна при различните продукти
Сигурност	Поддържа се напълно	Поддържа се с ограничения
Изгледи	Поддържа се напълно	Поддържа се с ограничения
Цялост	Поддържа се напълно	Поддържа се с ограничения

Глава 2. Обектно-релационни бази от данни

3. Обектно-релационна алгебра

3.1 Въведение

Тази секция представя обектно-релационната алгебра и операциите й ограничение, проекция, произведение, обединение, сечение, разлика, присъединяване и делене.

3.2 Въведение в оригиналната алгебра

Традиционните операции с множества обединение, сечение, разлика и Декартово произведение (всички те се променят в известна степен, за да се изпълняват върху релации, вместо върху множества). Специалните операции върху релации са ограничение, проекция, присъединяване и деление:

- **Ограничение (Restrict)** : връща релация, съдържаща всички редове от зададена релация, които удовлетворяват специфично условие.
- **Произведение (Product)** : връща релация, съдържаща всички възможни редове от комбинацията на два реда, всеки от които специфицира различна релация.
- **Обединение (Union)** : връща релация, съдържаща всички редове, които се срещат в коя и от двете релации.
- **Сечение (Intersect)** : връща релация, съдържаща всички редове, които се срещат и в двете релации.
- **Разлика (Difference)**: връща релация, съдържаща всички редове, които присъстват в първата, но не присъстват във втората релация.
- **Присъединяване (Join)** : връща релация, съдържаща всички възможни редове, които са комбинации от два реда, всеки от зададена релация, такива че за всяка комбинация имат общи стойности за общите атрибути, които не се повтарят.
- **Делене (Divide)** : използва две унарни и една бинарна релации и връща релация, съдържаща всички редове от едната унарна релация, които се срещат в бинарната и съвпадат с всички редове в другата унарна релация [5].

3.3 Семантика

В тази секция е представено обяснение на предходните осем операции. Операциите са представени в реда обединение, сечение, разлика, произведение, ограничение, проекция, присъединяване и делене.

3.3.1 Обединение

Обединение на две множества в математиката е множество от всички елементи, принадлежащи към кое и да е от двете оригинални множества. Тъй като релацията е множество, наречено множество от редове, е възможно да се изгради обединение на две такива множества; резултатът ще е множество от редове, намиращи се в кое и да е от двете оригинални релации.

3.3.2 Сечение

Както обединението и по същата причина, сечението на релации изисква операнди от един и същи тип. За дадени две релации A и B от еднакъв тип, сечението им, A INTERSECT B, е релация от същия тип, която се състои от всички редове t, присъстващи и в A и в B.

3.3.3 Разлика

Както обединението и сечението, операцията релационна разлика също изисква операнди от един и същи тип. За дадени две релации A и B от един и същи тип, разликата между тях, A MINUS B (в този ред), е релация от същия тип и се състои от всички редове t, присъстващи в A и не присъстващи в B.

3.3.4 Произведение

Декартовото произведение в математиката (произведение за по-кратко) на две множества е множество от всички наредени двойки, такива че във всяка двойка първият елемент е от първото множество, а втория – от второто. Така, по аналогия, Декартовото произведение на две релации ще бъде множество от наредени двойки от редове.

3.3.5 Ограничение

Нека релацията A има атрибути X и Y (възможно е да има и други атрибути) и нека Θ е една типична операция като "=",">" и др., така че условието X Θ Y е добе дефинирано и за дадени конкретни стойности на X и Y, дава булев резултат (true или false). Булевият резултат е Θ-ограничение на релацията A върху атрибутите X и Y (в този ред).

3.3.6 Проекция

Нека релацията A има атрибути X,Y,…..Z (възможно е да има и други атрибути). Тогава проекция на релацията A по X,Y,…..Z, A{X,Y,……..,Z} е релация с:

- Заглавие, породено от заглавието на A, чрез премахване на всички атрибути, които не присъстват в множеството t{X,Y,……..Z} и

- Тяло, състоящо се от всички редове {X:x,Y:y,…..,Z:z}, такива че ред, присъстващ в A със стойност x за X , стойност y за Y, ... и стойност z за Z.

3.3.7 Присъединяване

Присъединяването съществува в няколко различни разновидности. Най-важното присъединяване е така нареченото естествено присъединяване (natural join). Нека релациите A и B имат заглавия {X1,X2,…..Xm,Y1,Y2,……Yn} и{Y1,Y2,……..Yn,Z1,Z2,……Zp} съответно, т.е. атрибутите Y1,Y2……..Yn се повтарят и в двете релации. Атрибутите X1,X2,…..Xm са останалите атрибути на A, а атрибутите Z1,Z2,…Zp са останалите атрибути на B. Считаме, че {X1,X2,……,Xm}, {Y1,Y2,……Yn} и {Z1,Z2,……Zp} са три съставни атрибута X,Y и Z, съответно. Тогава естественото присъединяване на A и B:

A JOIN B

Е една релация със заглавие {X,Y,Z} и тяло, състоящо се от множествата на всички редове {X:x, Y:y, Z:z}, такива че ред присъстващ в A със стойност x за X и стойност y за Y value, присъства и в B със стойност y за Y и стойност z за Z.

3.3.8 Делене

В [7] са дефинирани две различни операции делене: малко делене (Small Divide) и голямо делене (Great Divide), съответно. define two distinct "divide" operators that and the, respectively. Един <divide>, в който <per> се състои само от един <relational expression> е малко делене; <divide>, който се състои от списък <relational expression>, разделени със запетая и затворени в скоби, се нарича голямо делене. В труда е разгледано само малко делене, само заради ограничената форма на малкото делене.

3.4 Асоциатичност и комутативност

Лесно се проверява асоциативността на UNION, т.е. ако A, B и C са произволни релационни изрази, образуващи релации от един и същи тип, то изразите

(A UNION B) UNION C **и** **A UNION (B UNION C)**

Са логически еквивалентни. Следователно последователност от обединения може да бъде записана без обхващащи скоби за удобство, т.е. всеки от предходните изрази може да бъде еднозначно опростен като

A UNION B UNION C

Аналогични бележки могат да се приложат и към INTERSECT, PRODUCT и JOIN (но не и по отношение на MINUS).

Трябва да се отбележи, че UNION, INTERSECT, PRODUCT и JOIN (но не MINUS) са комутативни изрази.

Съществува и логическа еквивалентност и подобие за INTERSECT, PRODUCT и JOIN. В заключение, ако A и B нямат общи имена на атрибути, то A JOIN B е еквивалентно на A PRODUCT B, т.е. естественото присъединяване се свежда до Декартово произведение в този случай [5, 7].

4. Обектно-релационен SQL

4.1 Въведение

Този раздел представя някои основни концепции на Езика за дефиниране на обекти (Object Definition language, ODL) и Езика за заявки с обекти (Object Query Language, OQL) с приложение на функции, тригери (triggers) , изгледи (views), таблици, процедури, типове и операциите insert ,update ,delete и select.

4.2 Език за дефиниране на обекти (ODL)

Езикът за дефиниране на обекти (ЕДО) е език за дефиниране на спецификации на типове обекти за ODMG-съвместими системи. Основната му цел е да улесни преносимостта на схеми между съвместими системи като в същото време спомага за осигуряване на оперативна съвместимост между ООСУБД-та. ЕДО е еквивалент на езика за дефиниране на данни (data definition language, DDL) на традиционните СУБД-та. Той дефинира атрибутите и типовете връзки и определя сигнатурата на операциите. ЕДО не се отнася до изпълнението на сигнатурите. Синтаксисът на ЕДО разширява езика за дефиниране на интерфейси(Interface Definition Language, IDL) на Common Object Request Broker Architecture(CORBA). Групата Object Data Management Group (ODMG) се надяват, че ЕДО ще бъде основа за интегриране на схеми от различни източници и приложения [8].

Език за дефиниране на данни (ЕДД). Тези оператори на SQL дефинират структурата на базата от данни, редове, колони, таблици и специфики на базата от данни като местоположение на файла. Тези оператори са част от СУБД и има големи разлики между вариантите на SQL. CREATE; ALTER, DROP се наричат DDL COMMANDS [16].

- CREATE – създава обекти в базата от данни.
- ALTER – променя структурата на базата от данни.
- DROP – изтрива обекти от базата от данни.

4.3 Език за заявки с обекти (OQL)

Езикът за заявки с обекти (ЕЗО) предоставя декларативен достъп до обекти в базата от данни, ползвайки SQL-подобен синтаксис. Той не предоставя външни оператори за актуализация, но оставя това на операциите, дефинирани в обектния тип. Както при SQL, OQL може да се използва като самостоятелен език и като език вграден в друг, за който е дефинирано ODMG вграждане. Текущо поддържани езици са Smaltalk, C++ и Java. ЕЗО може да извиква и операции, програмирани в тези езици. Една заявка в ЕЗО е функция, която доставя обект, чийто тип може да бъде получен от оператор, допринесъл изразът на заявката. Преди дефиниране на заявката в ЕЗО, може да се изясни композицията от изрази [8].

Език за обработка на данни (DML, Data Manipulation Language). Тези SQL оператори се използват за извличане и обработка на данни. Тази категория обхваща най-основните команди включително DELETE, INSERT, SELECT и UPDATE. Операторите DML SQL имат някои малки разлики между вариантите на SQL. DML SQL командите включват:

- INSERT за добавяне на ред.
- UPDATE за промяна на данни в зададена колона.
- DELETE за изтриване на редове.
- SELECT за извличане на ред.

Командите DML не могат да се възстановяват, когато една DDL команда се изпълни непосредствено след една DML. DDL след DML означава "auto commit". Промените се записват на диска, а не във буфера. Ако промените са записани във буфер, може да бъде направено rollback, докато при диска това не е възможно [9].

5. *Обектно-релационен SQL3*

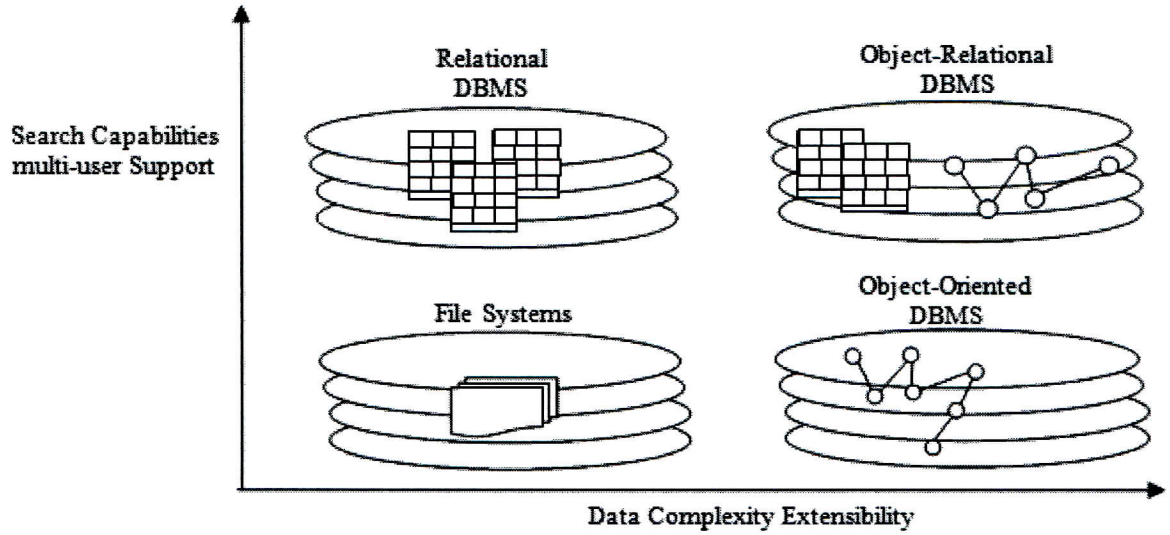
5.1 Въведение

Този раздел показва концептуално разширение на SQL, обектно-релационна система за управление на бази от данни като идентичност на обекта, типове редове, потребителски дефинирани типове (UDTs), потребителски дефинирани подпрограми, полиморфизъм, подтипове, постоянно съхранени модули и големи обекти.

5.2 Системи с обектно-релационни бази от данни

Доскоро изборът на СУБД изглеждаше между релационни СУБД (РСУБД) и обектно-ориентирани СУБД (ООСУБД) - фигура 5.1. Въпреки това продавачите на РСУБД продукти са още под заплаха от обещанията на ООСУБД. Те се съгласяват, че техните

системи в момента не са подходящи за напреднали приложения и че се изисква допълнителна функционалност. Въпреки това те отхвърлят твърдението, че разширените РСУБД няма да осигурят достатъчно функционалност или ще бъдат твърде бавни за да се справят адекватно с новата функционалност.



Фигура 5.1 Класификация на СУБД

Разглеждането на напредналите приложения с бази от данни, които се развиват, поради широкото използване на много обектно-ориентирани характеристики като потребителски разширяема система на типове, инкапсулация, наследяване, полиморфизъм, динамично вграждане на методи, сложни обекти включващи обекти, които не са в първа нормална форма и идентичност на обекти. Най-лесният начин за отстраняване на недостатъците на релационния модел е да се разшири модела с тези функции. Това е подходът, който е бил избран от много прототипи на разширени релационни системи, въпреки че всеки от тях реализира различна комбинация от характеристики. Следователно, няма един разширен релационен модел, а по-скоро има разновидности на тези модели. Всички модели обаче споделят едни и същи базови релационни таблици и езици за запитвания, всички вграждат някаква концепция за обект и някои имат възможност да съхраняват методи (или процедури, или тригери) така добре, както данните в бази от данни.

5.3 SQL3 (Какво е новото в SQL3 ?)

SQL3 е разширение на SQL и покрива:

- Конструктори на типове за тип на ред и тип референция.

- Потребителски дефинирани типове (типове разлики и структурирани типове), които могат да вземат участие във връзки супертип/подтип.
- Потребителски дефинирани процедури, функции и оператори.
- Типови конструктори на колекции (arrays, sets и lists).
- Поддръжка на големи оебкти Binary Large Objects (BLOBs) и Character Large Objects (CLOBs).

5.3.1 Идентичност на обекти

Всяка релация има неявно дефиниран атрибут OID, който съдържа уникален идентификатор на реда, като стойността на всеки OID се създава и поддържа от БД. Атрибутът OID може да се достъпва, но не и актуализира от потребителски заявки. Сред другите потребители OID може да се ползва като механизъм за симулиране на типове на атрибути, които референсират редове в други релации. Името на релацията може да се използва като име на тип, защото релациите, типовете и процедурите имат отделни пространства на имената [8].

5.3.2 Типове редове

Един тип на ред е поредица име на поле/тип данни, която осигурява тип данни, представляващи типовете редове в таблиците, така че целия ред да бъде съхранен в променлива, предадена като аргумент към подпрограми и върната като резултат от извикване на функция. Един тип ред може да бъде използван и за достъп до колона на таблица, съдържаща стойностите на реда [1, 8].

5.3.3 Потребителски дефинирани типове (UDTs)

Потребителски дефинираният тип се отнася до абстрактен тип данни (ADTs), който може да се използва по същия начин, както вградените типове (например, CHAR, INT, FLOAT). UDTs се разделят на две категории: различаващи се и структурирани типове. Най-простият тип UDT в SQL3 е различаващия тип, който позволява различаване на едни и същи основни типове. В по-общия случай дефиницията на UDT се състои от една или повече дефиниции на атрибути. Предложено е и дефиницията на UDT да включва допълнително и декларациите на подпрограмите.

5.3.4 Потребителски дефинирани подпрограми

Потребителски дефинираните подпрограми (UDRs) дефинират методи за всяка манипулация на данни и са важно допълнение към UDTs. Една ОРСУБД трябва да предоставя значима гъвкавост в тази област, така че допустимите UDRs да връщат сложни стойности,

които могат по нататък да се обработват (като таблици) и да поддържат предефиниране на имена на функции за улеснение на разработката на приложения. В SQL3, UDRs може да се дефинират като част от UDT или отделно, като част от една схема.

5.3.5 Подтипове и супертипове

SQL3допуска UDTs да участват в йерария подтип/супертип. Един тип може да има повече от един супертип (т.е. поддържа се множествено наследяване) и повече от един подтип. Един подтип наследява всички атрибути и поведения на неговите супертипове, може да да дефинира допълнителни атрибути и функции като всеки друг UDT и може да припокрива наследени функции [8].

5.3.6 Връзки и наследяване

Една релация наследява всички атрибути от своите родители, докато атрибут не се припокрие е дефиницията. Множественото наследяване се поддържа , но ако един и същи атрибут може да се наследи от повече от един родители и типовете на атрибута са различни, декларацията се забранява. Спецификациите на ключове също се наследяват [2, 8].

5.3.7.Постоянно съхранени модули

Добавени са редица нови типове оператори в SQL3, за да направят езика изчислително пълен, така че поведението на обектите (методите) да могат да се съхраняват и изпълняват в базата от данни като оператор на SQL. Операторите могат да се групират заедно в един съставен оператор (блок), със собствени локални променливи. Някои от допълнителните оператори предоставени в SQL3 са:

- Оператор за присвояване, който позволява резултата от израз на SQL да бъде присвоен на локална променлива, колона или атрибут от UDT.
- Оператор IF…THEN…ELSE…END IF, който позволява условни обработки.
- Оператор CASE, който позволява избор на изпълнима част, базирана на множество от алтернативи.
- Множество от оператори, което позволява циклично изпълнение на блок от SQL оператори. Итеративните оператори са: FOR,WHILE и REPEAT.

5.3.8 Полиморфизъм

Различни подпрограми може да имат едно и също име, т.е. подпрограмите могат да се предефинират, например да се позволи на

подтип UDT да предефинира метод, наследен от супертипа, трябва да са в сила следните ограничения:

- Няма две функции в една и съща схема, които имат една и съща сигнатура, т.е. еднакъв брой аргументи, еднакъв тип за всеки аргумент и един и същи тип на връщаната стойност.
- Няма две процедури в една и съща схема, които имат едно и също име и еднакъв брой параметри.

5.3.9 Големи обекти

Един голям обект е поле от таблица, което съхранява голямо количество данни като дълъг текстов или графичен файл. Съществуват три различни типа големи обекти, дефинирани в SQL3:

- Binary Large Object (BLOB), двоичен низ, който няма таблица на символите или свързана колация.
- Character Large Object (CLOB) и National Character Large Object (NCLOB), и двата са символни низове.

5.4 Сравнение на характеристиките на ОРСУБД и ООСУБД

Характеристика	ОРСУБД	ООСУБД
Връзки	Напълно се поддържа през потребителски дефинирани ограничения за референтната цялост	Поддържа се (например, използване на блиблиотеки от класове)
Наследяване	Поддържа се (отделни йерархии за UDTs и таблици)	Поддържа се
Полиморфизъм	Поддържа се (UDR извиквания, базирани на родови функции)	Поддържа се както в модела на обектно-ориентиран програмен език
Инкапсулация	Поддържа се през UDTs	Поддържани и разбити заявки

5.5 Заключение

SQL3 е разширение на SQL, което добавя конструктори на типове за тип на ред и тип референция, потребителски дефинирани типове (типове разлики и структурирани типове), които могат да вземат участие във връзки супертип/подтип, типови конструктори на колекции (arrays, sets и lists).

ОРСУБД е изградено на база РСУБД и ООСУБД.

Глава 3. Проектиране на бази от данни с обектно моделиране

6. Изследване на реализацията на връзките в ОРСУБД

6.1. Въведение

В този раздел е изследвана реализацията на наследяване, асоциативност, агрегация и композиция при връзките в БД Oracle 10g, за да се докаже конкурентоспособността на обектно-релационната технология. Използвана е посочената ОРСУБД за реализация на обектно-релационни и релационни БД. Причината за избор на Oracle 10g е защото той е лидер на пазара за бази от данни и включва много от SQL:2003 характеристики.

6.2. Case Study

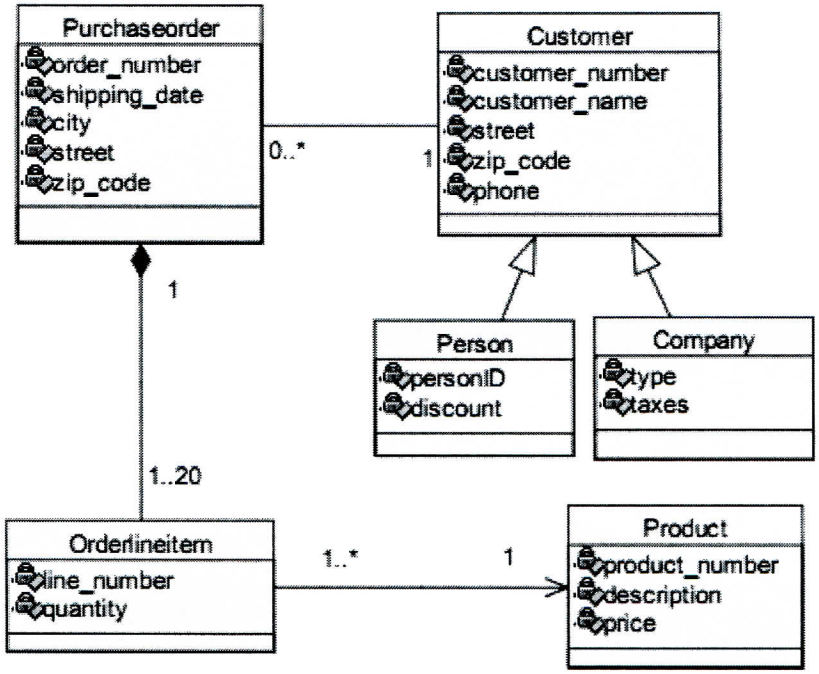
В сравнение с релационния стандарт SQL'92, обектно-релационните системи за управление на бази от данни (ORDBMS) базирани на стандарта SQL:2003 предлагат няколко нови възможности за реализиране на наследяване, асоциативност, агрегация и композиция при връзките между обекти. Тези нови възможности са базирани на потребителски-дефинираните типове (UDTs), референции и колекции. Референцията е логически указател към обект ред, който е конструиран от неговия идентификатор (OID). При обектно-релационният (O-R) подход връзките асоциация и агрегация могат да бъдат реализирани чрез една референция или колекция от референции, в зависимост кардиналността на връзката.

6.3. Изобразяване на слоевете в ОРСУБД

В [98] са дефинирани три слоя чрез трансформация на UML клас диаграма в ОРСУБД постоянни обекти. Първият кореспондира с UML клас диаграма, вторият е обектно-релационния слой, съставен от O-R елементи според стандарта SQL:2003 [102] - UDTs, масиви, multisets, референции, типове редове – и третият е обектно-релационен постоянен слой, съставен от типизирани таблици, които са дефинирани от елементите на втория слой и съдържат, освен останалите елементи, ключове, ограничения и OIDs. За разлика от релационния модел, допълнителния слой на обектно-релационния модел добавя по-голяма сложност.

6.4. Примерен модел на данните

Разгледан е примерен модел от литературата за администриране на продажби в бизнес компания, чиято UML клас диаграма, за реализация на схемата, е показана на фигура 6.2. Този модел съдържа много типове връзки, необходими за изпълнение на изследването.



Фигура 6.2. Клас диаграма на приложение за продажби

6.5. Дефиниране на релационната схема

Изобразяването на UML клас диаграмата в релационна схема е базирано на дефинициите, показнаи в [97]. Йерархията наследяване на класове, се изобразява по три начина, показани в [96, 97]: плосък, вертикален и хоризонтален. Тук е показан само плоския модел, чрез създаване на една таблица за всички класове (супер и подтипове) в йерархията. В йерархията е допуснато, че множествата Person и Company са разчленени и трите типа обекти (customers, persons и companies) трябва да бъдат представени и след това в таблицата, където тези атрибути нямат съответствие с типа на реда, се записва стойност NULL.

6.6. Дефиниране на обектно-релационната схема

O-R схемата се генерира чрез използване на референции, масиви и multisets и/или комбинация от тях, съгласно дефинициите в [98]. Трябва да се отбележи, че връката композиция Purchaseorder-Orderlineitem е реализирана по четири възможни начина:

- Първият (1.) дефинира тип масив Orderlineitem от 20 елемента в таблица Purchaseorder. Това е най-естествената реализация, съгласно връзката, дефинирана в UML клас диаграма. Обектите са включени като „част“ в „цялото“ поради силната връзка, където времето на живот на частта зависи от цялото време на живот. Типът масив Orderlineitem е използван в типа таблица

Purchaseorder, защото множествеността на частите е известна и максималната ѝ стойност е 20.

- Вторият (2.) дефинира масив от референции към обекти Orderlineitem в типа таблица Purchaseorder, реализиран чрез orderline_va attribute. Тази реализация е направена за да се използва референция в връзка композиция, така че да може да се третира като връзка агрегация. Важно е да се отбележи, че ако „цялото“ изтрие някоя процедура, за да елиминира „частта“, то трябва да се реализира съгласно поддръжка на цялостта на връзките. Това не е естествена реализация на връзка композиция; тук е направено за да се развие този тип връзка. Независимо от това, в някои случаи и зависимости от природата на връзката, това може да бъде алтернатива на една композиция.
- Третата (3.) и четвъртата (4.) алтернативни реализации включват multiset от референции и multiset от обекти съответно. Разликата между тези две реализации и предходните две е, че при multiset не се знае максималната големина на колекцията. it is not known the maximum size of the collection. Съображенията за използването на референции или обекти са същите, както при предходните параграфи.

6.7. Реализация на базата от данни

За да се направи подходящо изследване при различните реализации, таблиците на базата от данни са попълнени с хиляди обекти/редове чрез съхранени процедури, написани на програмния език, предоставен от ОРСУБД и съдържат случайни стойности, генерирани с функция.

В обектно-релационната схема все обект ред в обекта таблица (type table) има един свързан OID, който уникално го идентифицира. Той позволява на съответния обект ред да бъде референсиран от други обекти. В този случай се използва вграден тип данни, наречен REF. За да се ограничи, че само референции към зададения обект таблица могат да се реализират, е използван scoped REF. Той се съхранява по-ефективно отколкото unscoped REFs. В релационната схема всеки ред в една таблица има първичен ключ, който уникално я идентифицира, позволявайки присъединяване на таблици.

Броят на генерираните инстанции на всеки клас е показан в

клас	**Брой инстанции**
Company	427
Customer	900
Orderlineitem	58 245
Purchaseorder	9 900
Person	280
Product	15 000

Таблица 6.3. Брой обекти от всеки клас в модела на данните

таблица 6.3.

6.8. Оценка на тестовете между схемите

Дефинирани са няколко релационни и обектно-релационни заявки за сравнение на производителността на предложените схеми. Както може да се забележи, избраните заявки проучват използването на колекции (масив и/или multiset) от обекти и референции, една референция и йерархия при наследяване. Тези заявки са избрани за да се покаже разликата между O-R и релационния подход и за да се сравни производителността.

Всяка заявка е изпълненан 20 пъти в различни моменти и е изчислено средното време за изпълнението й. При реализацията не е направен запис на буфера на БД между стартиранията, защото в реалния живот потребителите изпълняват множество приложения по едно и също време като консумират системни ресурси. Хардуерът, който е използван за реализация и тестване е Intel® Core™ 2 Duo CPU T8300 @ 2.4 GHz, с 3 GB памет, като е стартирана Windows 7 Ultimate операционна система.

Целта на сравнението между заявките е да се направи сравнителна оценка на предложените реализации и анализ на използването на референции, масиви и multiset от обектно-релационната технология срещу операцията присъединяване от релационния подход. В този анализ се счита, че времето за отговор и плана на изпълнение е дефиниан от оптимизатор.

Заявка 1. Да се определят номерата на поръчките и детайлите номер на ред и поръчано количество.

В тази заявка е анализирано поведението на четирите реализации на връзка композиция, за да се намери най-подходящата алтернатива по отношение на времето за отговор.

1.1 Масив от обекти

```
SELECT p.order_number, o.line_number, o.quantity
FROM purchaseorder_t p, TABLE(p.orderline_va) o;
```

1.2 Масив от референции

```
SELECT p.order_number, o.column_value.line_number,
o.column_value.quantity
FROM purchaseorder2_t p, TABLE(p.orderline_va) o;
```

1.3 Multiset от референции

```
SELECT p.order_number, o.column_value.line_number,
o.column_value.quantity
```

```
FROM purchaseorder3_t p, TABLE(p.orderline_tab) o;
```

1.4 Multiset от обекти

```
SELECT p.order_number, o.line_number, o.quantity
FROM purchaseorder4_t p, TABLE(p.orderline_tab) o;
```

1.5 Релационен модел

```
SELECT p.order_number, o.line_number, o.quantity
FROM purchaseorder p, orderlineitem o
WHERE p.order_number = o.order_number;
```

Получените резултати са показани в таблица 6.4.

Заявка	Избрани редове	Време за отговор (mm:ss)
1.1	58245	00:04
1.2		00:09
1.3		00:09
1.4		00:03
1.5		00:03

Таблица 6.4. Резултати от заявка 1

Заявка 2. Да се определят клиентите, техните номера на поръчките заедно с номер на ред и поръчано количество.

Тази заявка е подобна на заявка 1, но в този случай се започва от тип таблица customer и се добавят допулнителни multiset от референции. Когато се реализира композицията като една агрегация, са наети две hop референции. В този случай е използван пакет от две колекции.

2.1 Multiset от референции + Масив от обекти

```
SELECT c.customer_number, c.customer_name,
p.column_value.order_number, o.line_number, o.quantity
FROM customer_t c, TABLE(c.purchase_tab) p,
TABLE(p.column_value.orderline_va) o;
```

2.2 Multiset от референции + Масив от референции

```
SELECT c.customer_number, c.customer_name,
p.column_value.order_number, o.column_value.line_number,
o.column_value.quantity
FROM customer2_t c, TABLE(c.purchase_tab) p,
TABLE(p.column_value.orderline_va) o;
```

2.3 Multiset от референции + Multiset от референции
SELECT c.customer_number, c.customer_name,
p.column_value.order_number, o.column_value.line_number,
o.column_value.quantity
FROM customer3_t c, TABLE(c.purchase_tab) p,
TABLE(p.column_value.orderline_tab) o;

2.4 Multiset от референции + Multiset от обекти
SELECT c.customer_number, c.customer_name,
p.column_value.order_number, o.line_number, o.quantity
FROM customer4_t c, TABLE(c.purchase_tab) p, TABLE(p.
column_value.orderline_tab) o;

2.5 Релационен модел
SELECT c.customer_number, c.customer_name, p.order_number,
o.line_number, o.quantity FROM customer_plano c, purchaseorder p,
orderlineitem o
WHERE c.customer_number = p.customer_number AND p.order_number
= o.order_number;

Получените резултати са показани в таблица 6.5.

Заявка	Избрани редове	Време за отговор (mm:ss)
2.1	58245	00:08
2.2		00:12
2.3		00:17
2.4		00:17
2.5		00:05

Таблица 6.5. Резултати от заявка 2

Заявка 3. Да се определят продуктите, поръчани от клиентите.

В тази заявка а използвани две колекции плюс единична референция за извличане на информация, т.е. наети са три hop референции.

3.1 Multiset от референции + Масив от обекти + Единична референция
SELECT c.customer_number, c.customer_name,
p.column_value.order_number, o.reftoproduct.product_number
FROM customer_t c, TABLE(c.purchase_tab) p,
TABLE(p.column_value.orderline_va) o;

3.2 Multiset от референции + Масив от референции + Единична референция

```
SELECT c.customer_number, c.customer_name,
p.column_value.order_number,
o.column_value.reftoproduct.product_number
FROM customer2_t c, TABLE(c.purchase_tab) p,
TABLE(p.column_value.orderline_va) o;
```

3.3 Multiset от референции + Multiset от референции + Единична референция

```
SELECT c.customer_number, c.customer_name,
p.column_value.order_number, o.column_value.reftoproduct.description
FROM customer3_t c, TABLE(c.purchase_tab) p,
TABLE(p.column_value.orderline_tab) o;
```

3.4 Multiset от референции + Multiset от обекти + Единична референция

```
SELECT c.customer_number, c.customer_name,
p.column_value.order_number, o.reftoproduct.product_number
FROM customer4_t c, TABLE(c.purchase_tab) p,
TABLE(p.column_value.orderline_tab) o;
```

3.5 Релационен модел

```
SELECT c.customer_number, c.customer_name, p.order_number,
pr.description
FROM customer_plano c, purchaseorder p, orderlineitem o, product pr
WHERE c.customer_number = p.customer_number AND p.order_number
= o.order_number AND o.product_number = pr.product_number;
```

Получените резултати са показани в таблица 6.6.
Може да се отбележи, че заявките с трета добавена hop имат същата производителност, както при заявка 2.

Заявка	Избрани редове	Време за отговор (mm:ss)
3.1	58245	00:12
3.2		00:14
3.3		00:16
3.4		00:13
3.5		00:09

Таблица 6.6. Резултати от заявка 3

Заявка 4. Да се определи информацията за всички клиенти от тип person.

В тази заявка се ползва йерайрията при наследяване на Customer, за да се получи информацията на супертипа за подтипа person. Свойството заменяемост позволява съхранение на произволен подтип в таблицата на супертипа.

4.1 Обектно-релационен

```
SELECT p.customer_number, p.customer_name, p.street, p.city
FROM customer_t p WHERE VALUE(p) IS OF (person_ob);
```

4.2 Релационен

```
SELECT customer_number, customer_name, street, city
FROM customer_plano WHERE type = 'P';
```

Заявка	Избрани редове	Време за отговор (mm:ss)
4.1	280	< 1 sec
4.2		< 1 sec

Таблица 6.7. Резултати от заявка 4

Получените резултати са показани в таблица 6.7.

6.9. Заключение

В този раздел бе представена оценка на реализацията на връзки от различни типове в обектно-релационната схема и бе направено сравнение между тях и с релационния подход.

Тези задачи са по сложни отколко релационния модел, който предоставя директно изобразяване. Няколко O-R схеми бяха дефинирани, използвайки различни алтернативи за реализацията на връзка композиция. Масиви и multiset от референции и обекти беше използвано за тази цел. Предложен е графичен елемент за поддържане на обектно-релационното разширение на релационните схеми.

Оценката на O-R реализациите спрямо релационния подход е направена на база множество заявки, тяхното време за отговор и плана им за изпълнение. В резултат от това изследване е направен сравнителен анализ на използването на масиви, multiset, обекти и референции, използвани за реализация на връзки композиция и агрегация. Не може да бъде направено общо заключение, кой подход е най-добър. Всеки случай може да бъде анализиран как да бъде реализиран съгласно бизнес правилата, съществуват множество

алтернативи и най-добрата може да се избере след направа на тестове. Производителността на йерархията при наследяване е една и съща при двете анализирани технологии, като O-R дава по-голяма гъвкавост при оценка на типовете.

7. Дизайн и реализация на бази от данни с ОРСУБД

7.1 Въведение

Този раздел представя моделирането с SQL оператори на приложение за търговско дружество. Навсякъде има сравнение за броя на атрибутите в SQL операторите с и без използване на обекти.

7.2 Приложение с обектно-релационна БД

Приложението се фокусира върху достъпа и промяната на корпоративни данни, съхранение в таблици, съставени от вградени в SQL типове данни като integer, date ,number и CHAR.

Както бе обяснено в предходната глава, обектният тип има атрибути от различни типове, аналогично на колони на таблица. В схемата на базата от данни присъстват таблици с обекти и без тях:

Employee таблица без Object

Emp No	Fname	Mname	Lname	DoB	PoB	City	Street	Salary	Emp sex	Dept No	Nationality

Employee таблица с Object

Emp No	Emp -Name	Emp-Birth	Emp Address	Salary	Emp Sex	Dept No	Nationality

Project таблица без Object

Proj-No	Proj-Name	City	Street	Manager-No	Dept-No

Project таблица с Object

Proj-No	Proj-Name	Proj-Address	Manager-No	Dept-No

Family таблица без Object

Emp -No	Fname	Mname	Lname	DOB	POB	Sex

Family таблица с Object

Emp-No	Family-Name	Family-Birth	Sex

Department таблица без Object

Dept-No	Dept-Name	Emp-No	Fmanager	Mmanager	Lmanager

Department таблица с Object

Dept-No	Dept-Name	Manager-Name	Emp-No

Използването на тези обекти при създаване на структурата на таблиците намалява размера на кода, времето за проектиране и времетъо за изпълнение на приложението. Самото то става по четимо, използваемо и разбираемо.

След реализацията на приложенията с и без обекти е направено сравнение за броя на атрибутите в операторите на SQL:

DML	**Без използване на обекти**	**С използване на обекти**
INSERT	6	3
UPDATE	4	2
DELETE	3	1
SELECT	7	4

7.5 Заключение

Моделирано е софтуерно приложение по два подхода: с обектна база от данни и с релационна база от данни. Направено е измерване на на броя на атрибутите при двата подхода и са анализирани времето за разработка на проекта, читаемостта, разбираемостта и използваемостта на кода. От направените тестове може да се заключи, че използване на ОРСУБД прави приложението по четимо, използвамо и намалява размера на кода.

АСПЕКТИ НА БЪДЕЩИ РАЗРАБОТКИ, ПРОИЗТИЧАЩИ ОТ ДИСЕРТАЦИОННИЯ ТРУД

Получените резултати стимулират за по-нататъшно развитие на обектно-релационен подход. Ето някои аспекти в тази посока:

- Добавяне на паралелно изпълняващи се конструкции в с цел ускоряване на изчислителния процес;
- Използване на получените резултати в учебния процес на студентите от образователна степен „магистър", специалност „КСТ".

ПРИНОСИ НА ДИСЕРТАЦИОННИЯ ТРУД

1. Изучени са обектно-ориентираното моделиране на данни и концепциите на обектно-ориентираните бази от данни и са дискутирани основите на обектно-релационната алгебра и обектно-релационния SQL.

2. Предложено е разширение на граматиката на логическите операции и специфични заявки за оптимизация работата на ОРСУБД, реализирани посредством тригери и сървърно-базирани процедури.

3. Направен е сравнителен анализ на ОРСУБД и ООСУБД и е изследвана производителността на реализациите на връзки в ОРСУБД Oracle.

4. Разработено е графично представяне на структурите в ОРСУБД посредством диаграми.

5. Създаден е модел на бази от данни с ОРСУБД и РСУБД като са използвани разработените графични диаграми..

6. Проектирана е и реализирана софтуерна система в ОРСУБД и РСУБД. Направен е сравнителен анализ на производителността и са доказани предимствата на ОРСУБД в количеството код, който се създава при разработка на приложението.

ИЗПОЛЗВАНА ЛИТЕРАТУРА (ИЗВАДКА)

[1] Database Management Systems Revisited, An Updated DACS State-of-the-Art Report. Prepared by: Gregory McFarland, Andres Rudmik, and David Lange Modus Operandi, Inc, 1999.

[2] Cattell, R.G.G., Object Data Management: Object-Oriented and Extended Relational Database Systems. Massachusetts: Addison-Wesley, 1997.

[3] Atkinson, Malcolm, et al., "The Object-Oriented Database System Manifesto." Building An Object-Oriented Database System, California: Morgan Kaufmann, 1999.

[5] C. J DATE An Introduction to Database System – 7th ed, Addison Wesley, 2005.

[7] Hugh Darwen and C.J.Date: " Into the Great Divide," in C.J.Date and hugh Drawen, Relational Database Writings 1989-1991. Reading,, mass.: Addison-Wesley 1992.

[8] Thomas Connolly and Carolyn Begg, Database systems of particular approach to Design, Implementation and management. Addison Welley 2001.

[9] Introduction to Oracle9i_SQL-Instructor Guide Volume 2.

[16] Stonebraker M. (2005). Object-Relational DBMS: The Next Great Wave. San Francisco, CA: Morgan Kaufmann Publishers Inc.

[96] Carey, M., Chamberlin, D., Narayanan, S., Vance, B., Doole, D., Rielau, S., Swagerman, R. and Mattos, N.: O-O, What Have They Done to DB2?. In Proc. of 25th International Conference on Very Large Data Bases. Edinburgh, Scotland (1999).

[97] Elmasri, R. and Navathe S.: Fundamentals of Database Systems, Third Edition. Addison Wesley. (2000).

[98] Golobisky, M.F. and Vecchietti, A.: Mapping UML Class Diagrams into Object-Relational Schemas. In Proc. of the Argentinian Symposium of Software Engineering (ASSE 2005). ISSN: 1666-1087. Pág. 65-79. 34 JAIIO, Rosario, Santa Fe, Argentina (2005).

СПИСЪК НА ПУБЛИКАЦИИТЕ ПО ТЕМАТА НА ДИСЕРТАЦИОННИЯ ТРУД

1. Gotseva, D. **Loie Naser Mahmoud Nimrawi**, COMPARISON BETWEEN OBJECT-RELATIONAL AND OBJECT-ORIENTED DATABASES. CASE STYDI, IX International Conference "Challenges in High Education and Research in 21st Centuary", 2011.

2. Gotseva, D. **Loie Naser Mahmoud Nimrawi**, AN APPROACH OF OBJECT DATABASE MODELING, IX International Conference "Challenges in High Education and Research in 21st Centuary", 2011.

3. Gotseva, D. **Loie Naser Mahmoud Nimrawi**, An Approach to Object Oriented Database Perspective, "Computer Science'2011", 2011.

4. Gotseva, D. **Loie Nimrawi**, Object Relational Database Concepts, Computer and Communication Engineering, 2012. (under print)

5. **Nimrawi, L.**, Investigation Of Performance In Object-Relational Databases, X International Conference "Challenges in High Education and Research in 21st Centuary", 2012. (under print).

APPROACH OF OBJECT DATABASE MODELING

ANOTATION

The aim of the thesis is to create an approach for object data modeling in databases. To fulfill this objective have been formulated the following tasks:

- To explore object-oriented data modeling, and object-oriented database concepts.
- To examine the fundamentals of object-relational algebra, object relational SQL, ORDBMS, and OODBMS. A comparative analysis of ORDBMS and OODBMS to be done.
- Be investigated relationship implementation performance in ORDBMS Oracle and proposed approach to database design with ORDBMS and OODBMS. Be realized a sample database application through ORDMBS and RDBMS in order to study the productivity of database design.

It's studied object-oriented data modeling, and object-oriented database concepts, and discussed the fundamentals of object-relational algebra, and object relational SQL.

It's proposed an extension of the grammar of logical operations and specific queries for optimization of working in ORDBMS, implemented through triggers and server-based procedures.

A comparative analysis of ORDBMS and OODBMS is done and examined the relationship implementation performance in ORDBMS Oracle.

It's developed a graphical representation of ORDBMS structure using diagrams.

It's created database models with ORDBMS and RDBMS using graphical diagrams.

After implementation of the system in RDBMS and ORDBMS a comparative analyses of the performance is done, and it's proved the benefit of ORDBMS in the amount of code that was created when developing application.