**Лабораторная работа № 6. Процесс проектирования БД. Нормализация.**

|  |
| --- |
| Настройка SQL\*Plus. Необходимо включить режим ECHO и вывести протокол лаб. работы в файл *<Фамилия студента>.txt*. Этот файл является отчетом о проделанной лаб. работе.  SQL> set echo on  SQL> spool d:\ф\_и\_о. txt  …………………………………….  Завершить протокол лаб. работы (команда spool off) и направить результаты преподавателю по дистанционной системе Microsoft Teams.  SQL> spool off |

Нормализация баз данных — очень важное понятие, рассматриваемое в процессе проектирования работоспособной схемы базы данных. Идея нормализации данных была предложена Е. Ф. Коддом в 1972 году, и с тех пор, она стала камнем преткновения при проектировании любой реляционной базы данных. По факту, любая схема, если она работоспособна, должна придерживаться определенных правил. Реляционная схема рассматривается как серия операций, которые проводятся над данными. Таким образом, сводятся к минимуму избыточность данных и аномалии при вставке, обновлении и удалении, путем более детального рассмотрения и определения отношений между различными сущностями.

Например. Cпроектируем фрагмент складского учета (документ - накладная на отпущенный товар – рис. 1):

|  |  |
| --- | --- |
| 107_1.gif (3473 b) | 107_2.gif (2166 b) |

|  |  |
| --- | --- |
| 107_5.gif (3188 b) | 107_6.gif (3468 b) |
| 107_7.gif (3883 b) | 107_8.gif (4879 b) |
| В этом примере проектирования структуры базы данных использовался метод НФ (нормальных форм), который является итерационным и заключается в последовательном переводе отношения из первой нормальной формы (1НФ) в нормальные формы более высокого порядка по определенным правилам.   * В нормализованной структуре базы данных можно производить сложные выборки данных относительно простыми SQL-запросами. * Нормализованная база данных позволяет надежно хранить данные (целостность). * Нормализация предотвращает появление избыточности хранимых данных. Данные всегда хранятся только в одном месте, что делает легким процесс вставки, обновления и удаления данных. Есть исключение из этого правила. Ключи, сами по себе, хранятся в нескольких местах потому, что они копируются как внешние ключи в другие таблицы. * Масштабируемость – это возможность системы справляться с будущим ростом. Для базы данных это значит, что она должна быть способна работать быстро, когда число пользователей и объемы данных возрастают. | |

**Задание 1. Составить SQL код создания таблицы/таблиц, определить первичный ключ, ввести данные (c учетом 1НФ).**

SQL>CREATE …………………………………………….;

SQL>INSERT …………………………………………….;

SQL>ALTER .…………………………………………….;

*Первая нормальная форма (1НФ) - выполняется, если все значения атрибутов атомарны, то есть неделимы. Например, дана таблица:*

Фирма Модели

**BMW M5, X5M, M1**

**Nissan GT-R**

Нарушение нормализации 1НФ происходит в моделях BMW, т.к. в одной ячейке содержится список из 3 элементов: M5, X5M, M1, т.е. он не является атомарным. Преобразуем таблицу к 1НФ:

Фирма Модели

**BMW M5**

**BMW X5M**

**BMW M1**

**Nissan GT-R**

*Правило: каждая таблица имеет первичный ключ, состоящий из наименьшего возможного количества полей.*

*Правило: поля не имеют дубликатов в каждой записи и каждое поле содержит только одно значение.  
Правило: порядок записей таблицы не должен иметь значения.*

**Задание 2. Составить SQL код создания/изменения таблицы/таблиц, определить первичный ключ, установить связи, ввести данные (c учетом 1НФ, 2НФ).**

*Для того, чтобы база данных была нормализована согласно 2НФ, она должна быть нормализована согласно 1НФ. 2НФ связана с избыточностью данных. Следование второй нормальной форме – это вопрос нахождения данных, которые часто дублируются в записях таблицы и которые могут принадлежать другой сущности. Например, дана таблица:*

Модель Фирма Цена Скидка

**M5 BMW 5500000 5%**

**X5M BMW 6000000 5%**

**M1 BMW 2500000 5%**

**GT-R Nissan 5000000 10%**

Таблица находится в первой нормальной форме, но не во второй. Цена машины зависит от модели и фирмы. Скидка зависят от фирмы, то есть зависимость от первичного ключа неполная. Исправляется это путем декомпозиции на два отношения, в которых не ключевые атрибуты зависят от ПК.

Модель Фирма Цена

**M5 BMW 5500000**

**X5M BMW 6000000**

**M1 BMW 2500000**

**GT-R Nissan 5000000**

Фирма Скидка

**BMW 5%**

**Nissan 10%**

*Правило: поля с не первичным ключом не должны быть зависимы от первичного ключа.*

**Задание 3. Составить SQL код создания/изменения таблицы/таблиц, определить первичный ключ, установить связи, ввести данные (c учетом 1НФ, 2НФ, 3НФ).**

*Третья нормальная форма связана с транзитивными зависимостями. Транзитивные зависимости между полями базы данных существует тогда, когда значения не ключевых полей зависят от значений других не ключевых полей. Чтобы база данных была в третьей нормальной форме, она должна быть во второй нормальной форме. Например, дана таблица:*

Модель Магазин Телефон

**BMW Риал-авто 87-33-98**

**Audi Риал-авто 87-33-98**

**Nissan Некст-Авто 94-54-12**

Таблица находится во 2НФ, но не в 3НФ. В отношении атрибут «Модель» является первичным ключом. Личных телефонов у автомобилей нет, и телефон зависит исключительно от магазина. Таким образом, в отношении существуют следующие функциональные зависимости:

Модель → Магазин,

Магазин → Телефон,

Модель → Телефон.

Зависимость Модель → Телефон является транзитивной, следовательно, отношение не находится в 3НФ. В результате разделения исходного отношения получаются два отношения, находящиеся в 3НФ:

Магазин Телефон

**Риал-авто 87-33-98**

**Риал-авто 87-33-98**

**Некст-Авто 94-54-12**

Модель Магазин

**BMW Риал-авто**

**Audi Риал-авто**

**Nissan Некст-Авто**

*Правило: не может быть транзитивных зависимостей между полями в таблице.*

*Отдельные выдержки из предыдущих работ:*

|  |  |
| --- | --- |
| * вести файл протокола spool. txt   SQL> set echo on  SQL> spool d:\spool. Txt | * завершить протокол лаб. работы   SQL> spool off |
| * создать пользователя   SQL> create user user1 identified by user1  default tablespace system  quota unlimited on system;   * изменить пароль   SQL> alter user user1 identified by test;   * разблокировать   SQL> alter user user1 account unlock; | - просмотреть пользователей  SQL>select username from sys.dba\_users;  -предоставить роль DBA  SQL> grant dba to user1;  -уничтожить пользователя  SQL> drop user user1 cascade;  -показать пользователя  SQL>show user |
| - просмотреть названия таблиц в схеме USER1  SQL>select table\_name from sys.all\_tables where owner ='USER1'; | - просмотреть из system структуру таблицы regions схемы HR  SQL>desc HR.regions  - просмотреть из system содержимое таблицы regions схемы HR  SQL>select \* from HR.regions |
| * создать таблицу   SQL> CREATE TABLE dolj  (kod NUMBER(10) NOT NULL PRIMARY KEY, naimen VARCHAR2(50)); | - заполнить таблицу  SQL> INSERT INTO dolj VALUES (1, ‘Программист’);  - выбрать запись  SQL> SELECT \* FROM dolj WHERE kod=1; |
| SQL> select r.region\_name, c.country\_name from regions r, countries c where r.region\_id=c.region\_id;  REGION\_NAME COUNTRY\_NAME  ------------------------- ----------------------------------------  Americas Argentina  Asia Australia  Europe Belgium  Americas Brazil  Americas Canada  Europe Switzerland  Asia China  Europe Germany  Europe Denmark  Middle East and Africa Egypt  Europe France | SQL> select r.region\_name, c.country\_name from regions r, countries c where r.region\_id=c.region\_id order by r.region\_id;  REGION\_NAME COUNTRY\_NAME  ------------------------- ----------------------------------------  Europe Netherlands  Europe France  Europe United Kingdom  Europe Denmark  Europe Belgium  Europe Switzerland  Europe Italy  Europe Germany  Americas United States of America  Americas Canada  Americas Mexico |

|  |  |
| --- | --- |
| Логическая модель | Физическая модель |
| Сущность | Таблица |
| Атрибут | Столбец |
| Логический тип  (текст, число, дата, blob) | Физический тип (корректный тип, зависящий от выбранной СУБД) |
| Первичный ключ | Первичный ключ, индекс PK |
| Внешний ключ | Внешний ключ, индекс FK |
| Альтернативный ключ | AK-индекс, уникальный,  непервичный индекс |
| Правило бизнес-логики | Триггер или сохраненная  процедура |
| Взаимосвязи | Взаимосвязи, определяемые использованием FK-атрибутов |

|  |  |  |
| --- | --- | --- |
| HR_ER_DIAGRAM_01_CORR | | |
| CREATE TABLE regions  ( region\_id NUMBER  CONSTRAINT region\_id\_nn NOT NULL  , region\_name VARCHAR2(25)  );  CREATE UNIQUE INDEX reg\_id\_pk  ON regions (region\_id);  ALTER TABLE regions  ADD ( CONSTRAINT reg\_id\_pk  PRIMARY KEY (region\_id)  ) ; | CREATE TABLE countries  ( country\_id CHAR(2)  CONSTRAINT country\_id\_nn NOT NULL  , country\_name VARCHAR2(40)  , region\_id NUMBER  , CONSTRAINT country\_c\_id\_pk  PRIMARY KEY (country\_id)  )  ORGANIZATION INDEX;  ALTER TABLE countries  ADD ( CONSTRAINT countr\_reg\_fk  FOREIGN KEY (region\_id)  REFERENCES regions(region\_id)  ) ; | CREATE TABLE locations  ( location\_id NUMBER(4)  , street\_address VARCHAR2(40)  , postal\_code VARCHAR2(12)  , city VARCHAR2(30)  CONSTRAINT loc\_city\_nn NOT NULL  , state\_province VARCHAR2(25)  , country\_id CHAR(2)  ) ;  CREATE UNIQUE INDEX loc\_id\_pk  ON locations (location\_id) ;  ALTER TABLE locations  ADD ( CONSTRAINT loc\_id\_pk  PRIMARY KEY (location\_id)  , CONSTRAINT loc\_c\_id\_fk  FOREIGN KEY (country\_id)  REFERENCES countries(country\_id) ) ; |