МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)
Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«14» 12 2021 г.

Методология проектирования баз данных

методические указания к практическим занятиям для магистров направления 02.04.03 Математическое обеспечение и администрирование информационных систем

Курск 2021

УДК 004.65
Составитель: Ю.А. Халин

Рецензент
Кандидат технических наук, с.н.с., доцент А.В. Ткаченко

Методология проектирования баз данных: методические указания к практическим занятиям / Юго-Зап. гос. ун-т; сост. Ю.А. Халин. Курск, 2021. 16 с. Библиогр.: с. 16.

В работе рассматриваются модели данных и общая методология роектирования баз данных. Изложены краткие теоретические сведения, приведены примеры проектирования баз данных, а также задания для самостоятельного решения.

Методические рекомендации предназначены для студентов, обучающихся по направлению подготовки 02.04.03 Математическое обеспечение и администрирование информационных систем.

Текст печатается в авторской редакции.

Подписано в печать 14.12.2021 . Формат 60x84 1/16.
Усл.печ. л. 0,98 п.л . Уч.-изд. л. 0,81 . Тираж 100 экз. Заказ.1719 . Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Практическая работа №1
SQL. Язык определения данных.

Общепринятый термин «язык запросов» не совсем точно отражает возможности языка SQL, поскольку слово «запрос» подразумевает лишь выборку. В то время как язык SQL предназначен для манипулирования данными в реляционных базах данных, определения структуры баз данных и для управления правами доступа к данным в многопользовательской среде.

Существуют и используются две формы языка SQL: интерактивный SQL и встроенный SQL. Интерактивный SQL используется для задания SQL-запросов пользователем и получения результата в интерактивном режиме. Встроенный SQL состоит из команд SQL, встроенных внутрь программ, обычно написанных на каком-то другом языке (Паскаль, C, C++ и др.). Это делает программы, использующие такие языки, более мощными, гибкими и эффективными, обеспечивая их применение для работы с данными, хранящимися в реляционных базах. При этом, однако, требуются дополнительные средства интерфейса SQL с языком, в который он встраивается.

В язык SQL в качестве составных частей входят:

1. Язык определения данных – DDL (Data Definition Language) – дает возможность создания, изменения и удаления различных объектов базы данных (таблиц, индексов, пользователей, привилегий и т.д.).
2. Язык манипулирования данными – DML (Data Manipulation Language) – предоставляет возможность выборки информации из базы данных и ее преобразования.
3. Язык управления данными – DCL (Data Control Language).

Подчеркнем, что это не отдельные языки, а различные команды одного языка. Такое деление проведено только лишь с точки зрения различного функционального назначения этих команд.

DDL – язык определения данных.

Язык определения данных используется для создания и изменения структуры базы данных и ее составных частей – таблиц, индексов, представлений (виртуальных таблиц), а также триггеров и сохраненных процедур. Основными его командами являются:

CREATE DATABASE	(создать базу данных)
CREATE TABLE	(создать таблицу)
CREATE INDEX	(создать индекс)
ALTER DATABASE	(модифицировать базу данных)
ALTER TABLE	(модифицировать таблицу)
ALTER INDEX	(модифицировать индекс)
DROP DATABASE	(удалить базу данных)
DROP TABLE	(удалить таблицу)
DROP INDEX	(удалить индекс)

Создание таблицы.

Таблицы создаются командой CREATE TABLE. Эта команда создает пустую таблицу – таблицу без строк. Значения вводятся с помощью команды INSERT. Команда CREATE TABLE в основном определяет имя таблицы, в виде описания набора имен столбцов, указанных в определенном порядке. Она также определяет типы данных и размеры столбцов. Каждая таблица должна иметь по крайней мере один столбец.

Синтаксис команды CREATE TABLE:

CREATE TABLE table_name (field1 type1 [nullable] [default_value][pk] [,field2 type2 [nullable] [default_value][pk]] …)

Обозначения, использованные в команде CREATE TABLE, представлены в таблице.

table_name	Имя создаваемой таблицы.	Имя таблицы не должно содержать пробелов и должно содержать только латинские символы и цифры. Запрещено давать таблицам названия, начинающиеся с цифр.
field1, field2	Названия полей таблицы.	Правила по именованию полей такие же, как и у таблиц.
type1, type2	Типы полей данных.	Допустимые типы данных и их обозначение определяются конкретными СУБД. Наиболее поддерживаемые типы данных приведены ниже.
nullable		Может принимать значение NOT NULL, что означает – поле обязательно должно иметь значение.
default_value		Может принимать значение DEFAULT const, где const – константа, значение которой будет добавляться в поле, если при вставке не будет указано явно других значений.
pk	Первичный ключ.	Может принимать значение PRIMARY KEY.

Наиболее поддерживаемые типы данных.

CHAR(длина) CHARACTER(длина)	Строки символов постоянной длины.
VARCHAR(длина) CHAR VARYING(длина) CHARACTER VARYING(длина)	Строки символов переменной длины.
INTEGER INT	Целые числа.
SMALLINT	Малые целые числа.
NUMERIC(точность, степень) DECIMAL(точность, степень) DEC(точность, степень)	Десятичные числа.
FLOAT(точность)	Числа с плавающей запятой.
REAL	Числа с плавающей запятой низкой точности.
DOUBLE PRECISION	Числа с плавающей запятой высокой точности.
DATE	Дата.
TIME(точность)	Время
TIMESTAMP(точность)	Дата и время.
INTERVAL	Временной интервал.

Пример запроса на создание таблицы:

CREATE TABLE Knigi (Code NUMBER UNIQUE PRIMARY KEY, Name CHAR(100) NOT NULL, Author_name CHAR(100) NOT NULL)

Если в таблице используется составной первичный ключ, то необходимо использовать следующую конструкцию PRIMARY KEY:

PRIMARY KEY(field1 [, field2 …])

Пример запроса на создание таблицы с использованием составного первичного ключа:

CREATE TABLE Students (Family VARCHAR(50) NOT NULL, Name VARCHAR(50) NOT NULL, Address VARCHAR(100) NOT NULL, PRIMARY KEY(Family, Name))

Для создания связей между таблицами (внешних ключей) используется следующая конструкция:

FOREIGN KEY (field1 [,field2 …]) REFERENCES master_table (key_field1 [, key_field2 …]) [ON UPDATE update_rule] [ON DELETE delete_rule]

Обозначения, использованные в команде FOREIGN KEY, представлены в таблице.

field1, field2	Поля создаваемой таблицы, участвующие в связи.
master_table	Имя таблицы, являющейся главной в создаваемой связи.
key_field1, key_field2	Соответствующие полям field1, field2 поля главной таблицы (обязательно должны входить в состав первичного ключа).
ON UPDATE	Задает правила поведения подчинённой таблицы при изменении записей в главной таблице.
ON DELETE	Задает правила поведения подчинённой таблицы при удалении записей в главной таблице.
update_rule	Правила для изменения. Поддерживаемые значения приведены ниже.
delete_rule	Правила для удаления. Поддерживаемые значения приведены ниже.

Правила изменения и удаления могут принимать следующие значения.

CASCADE	Каскадное обновление (при изменении значения ключевых полей в главной таблице автоматически меняются поля в подчиненной) или удаление (при удалении записи в главной таблице автоматически удаляются все записи в дочерней таблице, соответствующие значению первичного ключа удаляемой записи).
SET NULL	Установка значения поля в NULL при модификации ключевого поля или удаления записи в главной таблице (при объявлении поля не должно присутствовать NOT NULL).
SET DEFAULT	Установка значения по умолчанию, предусмотренного для поля, при модификации ключевого поля или удаления записи в главной таблице (при объявлении поля должно присутствовать выражение DEFAULT).

Пример запроса на создание таблицы, связанной с уже имеющейся таблицей:

CREATE TABLE Author (ID NUMBER UNIQUE PRIMARY KEY, Author_name CHAR(100) NOT NULL, Author_address CHAR(100) NOT NULL, Biografia VARCHAR(100), FOREIGN KEY (Author_name) REFERENCES Knigi (Author_name))

Создание индекса.

Индексы создаются командой CREATE INDEX.

Синтаксис команды CREATE INDEX:

CREATE [UNIQUE] INDEX index_name ON table_name (field1 [ASC|DESC] [, field2 [ASC|DESC] …])

Обозначения, использованные в команде CREATE INDEX, представлены в таблице.

index_name	Имя создаваемого индекса.
table_name	Имя таблицы, для которой индекс создается.
field1, field2	Поля таблицы, включаемые в индекс.
ASC	Значения включаемых полей будут отсортированы по возрастанию (это значение задается по умолчанию).
DESC	Значения включаемых полей будут отсортированы по убыванию.
UNIQUE	В индекс будут включены только неповторяющиеся значения полей.

Пример запроса на создание индекса:

CREATE UNIQUE INDEX Index1 ON Author (ID ASC)

Модификация таблицы.

Модификация (редактирование структуры) существующей таблицы проводится командой ALTER TABLE.

Синтаксис команды ALTER TABLE:

ALTER TABLE table_name ADD|REMOVE field_definition [,ADD|REMOVE field_definition …]

Обозначения, использованные в команде ALTER TABLE, представлены в таблице.

table_name	Имя таблицы, структуру которой следует изменить.
ADD	Позволяет добавить поле в таблицу.
REMOVE	Позволяет удалить поле из таблицы.
field_definition	Выражение вида field1 type1 [nullable] [default_value] [pk] в случае ADD или имя удаляемого поля в случае REMOVE.

Пример запроса на модификацию таблицы:

ALTER TABLE Knigi ADD Comment VARCHAR(100)

Удаление таблицы.

Удаление существующей таблицы из базы данных проводится командой DROP TABLE.

Синтаксис команды DROP TABLE:

DROP TABLE table_name

Если удаляемая таблица является главной по отношению к некоторым другим, то раннее должны быть удалены эти таблицы или удалены связь между ними.

Пример запроса на удаление таблицы:

DROP TABLE Students

Удаление индекса.

Удаление индекса проводится командой DROP INDEX.

Синтаксис команды DROP INDEX:

DROP INDEX index_name ON table_name

Пример запроса на удаление индекса:

DROP INDEX Index1 ON Author

Задания к практической работе №1

Вариант 1.

Создать базу данных «Кафедра» состоящую из следующих таблиц «Список преподавателей», «Информация о преподавателях», «Список студентов», «Информация о студентах», «Список дисциплин».

1. Создать таблицу «Список преподавателей» (Spisok_prep).

Таблица Spisok_prep содержит поля: Prep_id (номер удостоверения преподавателя); FIO_prep (ФИО преподавателя); Zvanie (ученое звание преподавателя (доцент, профессор и пр.)); Stepen (ученая степень преподавателя (к.т.н., д.т.н. и пр.)). Поле FIO_prep является первичным ключом таблицы. Поля Prep_id и Discip должны обязательно иметь значения. Таблица Spisok_prep связана с таблицей Discipliny по полю FIO_prep.

2. Создать таблицу «Информация о преподавателях» (Info_prep).

Таблица Info_prep содержит поля: FIO_prep (ФИО преподавателя); Data_rozhd (день рождения преподавателя); Address_prep (адрес преподавателя); Phone_prep (телефон преподавателя). Поле FIO_prep является первичным ключом. Таблица Info_prep связана с таблицей Spisok_prep по полю FIO_prep, предусмотрено каскадное обновление.

3. Создать таблицу «Список студентов» (Spisok_stud).

Таблица Spisok_stud содержит поля: Stud_id (номер студенческого); FIO_stud (ФИО студента); Group (группа, в которой обучается студент); Forma_ob (форма обучения); Semestr (номер семестра). Поле FIO_stud является первичным ключом таблицы. Поля Stud_id и Semestr должны обязательно иметь значения. Таблица Spisok_stud связана с таблицей Discipliny по полю Semestr.

4. Создать таблицу «Информация о студентах» (Info_stud).

Таблица Info_stud содержит поля: FIO_stud (ФИО студента); Den_rozhd (день рождения студента); Address_ stud (адрес студента); Phone_ stud (телефон студента). Поле FIO_ stud является первичным ключом. Таблица Info_ stud связана с таблицей Spisok_ stud по полю FIO_ stud, предусмотрено каскадное обновление и каскадное удаление.

5. Создать таблицу «Список дисциплин» (Discipliny).

Таблица Discipliny содержит поля: Disc_id (код дисциплины); Disc_name (название дисциплины); Chasy (количество часов); FIO_prep (ФИО преподавателя, ведущего дисциплину); Semestr (номер семестра, в котором преподается дисциплина). Составной первичный ключ включает поля FIO_prep и Semestr.

6. Создать индекс Index1 для таблицы Spisok_prep. В индекс включаются поля Prep_id и FIO_prep. Значения полей отсортированы по возрастанию.

7. Изменить таблицу Discipliny. Удалить поле Chasy (количество часов) и добавить поля: Lek (количество лекционных часов); Prakt (количество часов практических занятий).

Вариант 2.

Создать базу данных «Фирма» состоящую из следующих таблиц «Служащие», «Отделы», «Регионы», «Клиенты».

1. Создать таблицу «Служащие» (Sluzh).

Таблица Sluzh содержит поля: ID (идентификатор служащего); Last_name (фамилия служащего); First_name (имя служащего); Data_rab (дата начала работы); Zarpl (зарплата служащего); Otd_id (номер отдела); Comment (комментарии). Поле ID является первичным ключом таблицы. Поля Last_name и Otd_id должны обязательно иметь значения. Таблица Sluzh связана с таблицей Otdels по полю Otd_id.

2. Создать таблицу «Отделы» (Otdels).

Таблица Otdels содержит поля: Otd_id (номер отдела); Name (название отдела); Reg_id (номер региона, в котором находится отдел). Поле Otd_id является первичным ключом. Поле Reg_id должно обязательно иметь значение. Таблица Otdels связана с таблицей Regions по полю Reg_id. Предусмотрено каскадное обновление.

3. Создать таблицу «Регионы» (Regions).

Таблица Regions содержит поля: Reg_id (номер региона); Reg_name (название региона); Reg_info (дополнительная информация о регионе). Поле Reg_id является первичным ключом таблицы. Поле Reg_name должно обязательно иметь значения.

4. Создать таблицу «Клиенты» (Client).

Таблица Client содержит поля: Client_id (номер клиента); Client_name (название фирмы-клиента); Address (адрес клиента); Reg_id (номер региона, в котором располагается фирма-клиент); ID (идентификатор служащего, который обслуживает фирму-клиента). Составной первичный ключ включает поля Client_id и Client_name. Таблица Client связана с таблицей Sluzh по полю ID и с таблицей Regions по полю Reg_id.

5. Создать индекс Index1 для таблицы Regions. В индекс включаются поля Reg_id и Reg_name. Значения полей отсортированы по возрастанию.

6. Изменить таблицу Otdels. Добавить поля Data_otkr (дата открытия отдела); Rukovod (руководитель отдела).

7. Изменить таблицу Sluzh. Удалить поле Comment (комментарии) и добавить поля: Dolzhn (должность); Phone (телефон).

Вариант 3.

Создать базу данных «Оптовый магазин» состоящую из следующих таблиц «Покупатели», «Продажи», «Продукция», «Срок годности», «Стоимость».

1. Создать таблицу «Покупатели» (Pokupateli).

Таблица Pokupateli содержит поля: Kod_pokupat (код покупателя); Nazv (название фирмы-покупателя); Director (ФИО директора); Address (адрес фирмы-покупателя). Поле Kod_pokupat является первичным ключом таблицы. Все поля должны обязательно иметь значения.

2. Создать таблицу «Продажи» (Prodazha).

Таблица Prodazha содержит поля: Kod_oper (код операции); Kod_prod (код продукции); Kod_pokupat (код покупателя); Kolich_prod (количество продукции); Stoim_pokupki (стоимость покупки). Поле Kod_oper является первичным ключом. Все поля должны обязательно иметь значения. Таблица Prodazha связана с таблицей Produktciya по полю Kod_prod. Таблица Prodazha связана с таблицей Pokupateli по полю Kod_pokupat, предусмотрено каскадное обновление и каскадное удаление.

3. Создать таблицу «Продукция» (Produktciya).

Таблица Produktciya содержит поля: Kod_prod (код продукции); Nazv_prod (название продукции); Post (информация о поставщике). Поле Kod_prod является первичным ключом таблицы. Все поля должны обязательно иметь значения.

4. Создать таблицу «Срок годности» (Srok_godn).

Таблица Srok_godn содержит поля: Kod_prod (код продукции); Data_pokupki (день покупки); Data_realiz (последний день для реализации. Поле Kod_prod является первичным ключом. Таблица Srok_godn связана с таблицей Produktciya по полю Kod_prod.

5. Создать таблицу «Стоимость» (Stoimost).

Таблица Stoimost содержит поля: Kod_prod (код продукции); Stoim (стоимость); Nadbavka (надбавка). Поле Kod_prod является первичным ключом. Таблица Stoimost связана с таблицей Produktciya по полю Kod_prod.

6. Создать индекс Index1 для таблицы Pokupateli. В индекс включаются поля Kod_pokupat и Nazv. Значения полей отсортированы по возрастанию.

7. Изменить таблицу Produktciya. Добавить поля: Info_prod (описание продукции); Prakt (количество часов практических занятий).

Контрольные вопросы:

1. Что входит в язык SQL в качестве составных частей?
2. Что такое DDL?
3. Какой командой создаются таблицы в SQL?
4. Как связываются между собой таблицы в SQL?
5. Как происходит модификация таблицы?

Практическая работа №2
SQL. Язык манипулирования данными.

Общепринятый термин «язык запросов» не совсем точно отражает возможности языка SQL, поскольку слово «запрос» подразумевает лишь выборку. В то время как язык SQL предназначен для манипулирования данными в реляционных базах данных, определения структуры баз данных и для управления правами доступа к данным в многопользовательской среде.

Существуют и используются две формы языка SQL: интерактивный SQL и встроенный SQL. Интерактивный SQL используется для задания SQL-запросов пользователем и получения результата в интерактивном режиме. Встроенный SQL состоит из команд SQL, встроенных внутрь программ, обычно написанных на каком-то другом языке (Паскаль, C, C++ и др.). Это делает программы, использующие такие языки, более мощными, гибкими и эффективными, обеспечивая их применение для работы с данными, хранящимися в реляционных базах. При этом, однако, требуются дополнительные средства интерфейса SQL с языком, в который он встраивается.

В язык SQL в качестве составных частей входят:

4. Язык определения данных – DDL (Data Definition Language) – дает возможность создания, изменения и удаления различных объектов базы данных (таблиц, индексов, пользователей, привилегий и т.д.).
5. Язык манипулирования данными – DML (Data Manipulation Language) – предоставляет возможность выборки информации из базы данных и ее преобразования.
6. Язык управления данными – DCL (Data Control Language).

Подчеркнем, что это не отдельные языки, а различные команды одного языка. Такое деление проведено только лишь с точки зрения различного функционального назначения этих команд.

DML – язык манипулирования данными.

Язык манипулирования данными используется для добавления, удаления и редактирования записей в базе данных Основными его командами являются:

INSERT	(добавление новых строк в таблицу)
UPDATE	(обновление данных в таблице)
DELETE	(удаление строк из таблицы)

Добавление новых строк в таблицу.

Добавление записей в таблицу происходит с использованием команды INSERT.

Синтаксис команды INSERT:

INSERT INTO table_name [(field1 [, field2…])] VALUES (val1 [,val2 …])

Обозначения, использованные в команде INSERT, представлены в таблице.

table_name	Название таблицы.
field1, field2	Список полей, значения которых в добавляемой записи будут соответственно равны val1, val2. Если список полей опущен, то значения определяются в порядке, заданном при создании таблицы.

Пример запроса на добавление новой строки в таблицу:

INSERT INTO Knigi VALUES (12345, “Турецкий гамбит”, “Борис Акунин”)

Обновление данных в таблице.

Обновление записей в таблице происходит с использованием команды UPDATE.

Синтаксис команды UPDATE:

UPDATE table_name SET field1=value1 [, field2=value2 …] [WHERE conditional]

Обозначения, использованные в команде UPDATE, представлены в таблице.

table_name	Название таблицы.
field1, field2	Поля таблицы.
value1, value2	Значения полей.
conditional	Определяет условие, удовлетворяющие которому записи будут модифицированы. Может состоять из одного или нескольких подусловий, объединённых командами AND (логическое И) или OR (логическое ИЛИ). Возможные виды условий приведены ниже.

Возможные виды условий.

field1 < | > | = | <> | >= | <= value1	Значение поля field1 меньше, больше, равно, не равно, больше или равно, меньше или равно значения value1.
field1 BETWEEN value1 AND value2	Значение поля field1 между value1 и value2.
field1 LIKE value1	Значение поля field1 «похоже» на value1, используется при проверке соответствия строковых полей некоторой маске.
field1 IN (value1, value2 [,…])	Значение поля field1 находится в списке value1, value2.
field1 IS NULL	Значение поля field1 отсутствует.
field1 IS NOT NULL	Значение поля field1 есть.
NOT conditional	Выбирать только те записи, когда условие conditional не выполняется.

Также допустимо использование в условиях вложенных запросов на выборку SELECT.

Если выражение WHERE отсутствует, то редактируются все записи таблицы.

Пример запроса на обновление данных в таблице:

UPDATE Knigi SET Author_name = ”Борис Акунин” WHERE Code IN 12345, 23456, 34567)

Удаление строк из таблицы.

Удаление записей из таблицы происходит с использованием команды DELETE.

Синтаксис команды DELETE:

DELETE FROM table_name [WHERE conditional]

Обозначения, использованные в команде DELETE, представлены в таблице.

table_name	Название таблицы.
conditional	Условия отбора удаляемых записей (строится аналогично UPDATE).

Пример запроса на удаление записей из таблицы:

DELETE FROM Knigi WHERE Author_name = ”Борис Акунин”

Задание к практической работе №2

1. Заполнить каждую из таблиц базы данных, созданной в практической работе №1, 10 значениями.
2. Обновить значения таблиц с использованием различных условий для каждой таблицы.
3. Удалить из каждой таблицы хотя бы одно значение с использованием различных условий отбора удаляемых записей в каждой таблице.

Контрольные вопросы:

1. Опишите интерактивный язык SQL.
2. Опишите встроенный язык SQL.
3. Что такое DML?
4. Какая команда применяется при добавлении новых строк в таблицу?
5. Как происходит обновление данных в таблице?
6. Какой командой удаляются строки из таблицы?

Лабораторная работа №3
SQL. Запросы на выборку данных.

Запрос – команда, которую дают программе базы данных, и которая сообщает ей чтобы она вывела определенную информацию из таблиц в память. Эта информация обычно посылается непосредственно на экран компьютера или терминала которым вы пользуетесь, хотя, в большинстве случаев, ее можно также послать принтеру, сохранить в файле (как объект в памяти компьютера), или представить как вводную информацию для другой команды или процесса.

Все запросы в SQL состоят из одиночной команды. Эта команда называется – SELECT.

Синтаксис команды SELECT:

SELECT [DISTINCT] field_list FROM table_list [WHERE conditional] [GROUP BY group_field_list] [ORDER BY order_field_list]

Обозначения, использованные в команде SELECT, представлены в таблице.

DISTINCT	Запрещает вывод повторяющихся записей в результате запроса.
field_list	Содержит список извлекаемых полей, разделённых запятой. Помимо названий полей этот список может содержать различные выражения по преобразованию информации (сложение, вычитание и т.д.). Если необходимо извлечь значения всех полей, то используется символ *.
table_list	Содержит одну или несколько таблиц, разделённых запятой с возможным указанием связей между таблицами, значения выбранных полей которых будут результатами запроса.
conditional	Задаёт условия отбора записей из списка таблиц.
group_field_list	Содержит список полей, по значениям которых будет группироваться результат запроса. Строится по тем же правилам, что и field_list.
order_field_list	Содержит список полей, по которым результат будет отсортирован. Строится по тем же правилам, что и field_list.

Пример запроса на выборку данных:

SELECT DISTINCT Code, Author_name FROM Knigi ORDER BY Author_name

Построение условий запроса аналогично построению условий в команде UPDATE.

Объединение таблиц.

Одна из наиболее важных особенностей запросов SQL – это их способность определять связи между многочисленными таблицами и выводить информацию из них в терминах этих связей, всю внутри одной команды. Этот вид операции называется – объединением, которое является одним из видов операций в реляционных базах данных.

При объединении, таблицы, представленные списком в предложении FROM запроса, отделяются запятыми. Предикат запроса может ссылаться к любому столбцу любой связанной таблицы и, следовательно, может использоваться для связи между ними. Обычно, предикат сравнивает значения

в столбцах различных таблиц, чтобы определить, удовлетворяет ли WHERE установленному условию.

Полное имя столбца таблицы фактически состоит из имени таблицы, сопровождаемого точкой и затем именем столбца.

Пример запроса на объединение таблиц:

SELECT Knigi.Code, Knigi.Author_name, Author.ID, Author.Biografia FROM Knigi, Author

Операция INNER JOIN.

Объединяет записи из двух таблиц, если связующие поля этих таблиц содержат одинаковые значения.

Синтаксис операции INNER JOIN:

FROM table_name1 INNER JOIN table_name2 ON table_name1.field1 = | <= | >= | < | > | <> table_name2.field2

Обозначения, использованные в команде INNER JOIN, представлены в таблице.

table_name1, table_name2	Имена таблиц, из которых объединяются записи.
field1, field2	Имена связываемых полей. Если поля не содержат числовых данных, они должны относиться к одному типу данных и содержать некоторые данные. Имена этих полей могут быть разными.

Операцию INNER JOIN можно использовать в любом предложении FROM. Это самый распространенный тип объединения. С его помощью происходит объединение записей из двух таблиц по связующему полю, если оно содержит одинаковые значения в обеих таблицах.

Как правило, запросы, содержащие операцию INNER JOIN, выполняются быстрее, чем при использовании этих же условий в WHERE.

Пример запроса на объединение таблиц с использованием операции INNER JOIN:

SELECT Knigi.Code, Knigi.Name, Knigi.Author_name, Author.ID, Author.Author_address, Author.Biografia FROM Knigi INNER JOIN Author ON Knigi.Author_name =Author.Author_name

Операции LEFT JOIN, RIGHT JOIN.

Синтаксис операций LEFT JOIN, RIGHT JOIN:

FROM table_name1 [LEFT | RIGHT] JOIN table_name2 ON table_name1.field1 = | <= | >= | < | > | <> table_name2.field2

Для того, чтобы создать левое внешнее объединение, необходимо использовать LEFT JOIN. С помощью левого внешнего объединения выбираются все записи первой (левой) таблицы, даже если они не соответствуют записям во второй (правой) таблице.

RIGHT JOIN позволит создать правое внешнее объединение. С помощью правого внешнего объединения выбираются все записи второй (правой) таблицы, даже если они не соответствуют записям в первой (левой) таблице.

Пример запроса на объединение таблиц с использованием операции LEFT JOIN:

SELECT Knigi.Code, Knigi.Name, Knigi.Author_name, Author.ID, Author.Author_address, Author.Biografia FROM Knigi LEFT JOIN Author ON Knigi.Author_name =Author.Author_name

Обобщение данных с помощью агрегатных функций.

Рассмотрим, как использовать значения, полученные путём выполнения простых запросов, чтобы получить из них информацию. Это делается с помощью агрегатных или общих функций, которые берут группы значений из поля и сводят их до одиночного значения. Изучим, как использовать эти функции, как определить группы значений, к которым они будут применяться, и как определить какие группы выбираются для вывода. Рассмотрим, при каких условиях можно объединить значения поля с этой полученной информацией в одиночном запросе.

Запросы могут производить обобщенное групповое значение полей точно также как и значение одного поля. Это делает с помощью агрегатных функций. Агрегатные функции производят одиночное значение для всей группы таблицы. Список агрегатных функций представлен в таблице.

COUNT	Производит номера строк или ненулевые значения полей, которые выбрал запрос.
SUM	Производит арифметическую сумму всех выбранных значений данного поля.
AVG	Производит усреднение всех выбранных значений данного поля.
MAX	Производит наибольшее из всех выбранных значений данного поля.
MIN	Производит наименьшее из всех выбранных значений данного поля.

Агрегатные функции используются подобно именам полей в предложении SELECT запроса, но с одним исключением, они берут имена поля как аргументы. Только числовые поля могут использоваться с SUM и AVG. COUNT, MAX, и MIN, могут использоваться с числовыми или символьными полями. Когда они используются с символьными полями, MAX и MIN будут транслировать их в эквивалент ASCII, который должен сообщать, что MIN будет означать первое, а MAX последнее значение в алфавитном порядке.

Это конечно, отличается от выбора поля при котором возвращается одиночное значение, независимо от того сколько строк находится в таблице. Из-за этого, агрегатные функции и поля не могут выбираться одновременно, пока предложение GROUP BY (описанное далее) не будет использовано.

Функция COUNT несколько отличается от всех. Она считает число значений в данном столбце, или число строк в таблице. Чтобы подсчитать общее число строк в таблице, используйте функцию COUNT со звездочкой вместо имени поля.

Пример запроса с использованием агрегатных функций:

SELECT COUNT (*) FROM Knigi

Предложение GROUP BY.

Предложение GROUP BY позволяет определять подмножество значений в особом поле в терминах другого поля, и применять функцию агрегата к подмножеству. Это дает вам возможность объединять поля и агрегатные функции в едином предложении SELECT.

GROUP BY применяет агрегатные функции независимо от серий групп, которые определяются с помощью значения поля в целом. Значение поля, к которому применяется GROUP BY, имеет, по определению, только одно значение на группу вывода, также как это делает агрегатная функция. Результатом является совместимость, которая позволяет агрегатам и полям объединяться таким образом.

Можно также использовать GROUP BY с многочисленными полями.

Предложение HAVING.

Предложение HAVING определяет критерии, используемые чтобы удалять определенные группы из вывода, точно также как предложение WHERE делает это для индивидуальных строк.

Аргументы в предложении HAVING следуют тем же самым правилам, что и в предложении SELECT, состоящем из команд использующих GROUP BY. Они должны иметь одно значение на группу вывода.

Задание к практической работе №3

Написать 15 запросов с использованием различных операций, агрегатных функций и предложений к базе данных, выполненной в практических работах 1 и 2.

Контрольные вопросы:

Что такое запрос? Из какой команды они состоят?
Как происходит объединение таблиц?
Опишите операцию INNER JOIN.
Опишите операции LEFT JOIN, RIGHT JOIN.
Как происходит обобщение данных с помощью агрегатных функций?
Для чего предназначено предложение GROUP BY?
Для чего предназначено предложение HAVING?

Список литературы:

1. Шилин, А.С. Перспективные методы проектирования реляционных баз данных : учебное пособие / А.С. Шилин. – Москва ; Берлин : Директ-Медиа, 2021. – 136 с. –URL: https://biblioclub.ru/index.php?page=book&id=602240 (дата обращения: 04.02.2022). – Режим доступа : по подписке. – Текст : электронный..

2. Аврунев, О. Е. Модели баз данных : учебное пособие / О. Е. Аврунев, В. М. Стасышин. – Новосибирск : Новосибирский государственный технический университет, 2018. – 124 с. –URL: 43 http://biblioclub.ru/index.php?page=book&id=575324 (дата обращения: 15.09.2020). – Режим доступа : по подписке. – Текст : электронный.

3. Кугаевских, А.В. Проектирование информационных систем. Системная и бизнес-аналитика : учебное пособие / А.В. Кугаевских. – Новосибирск : Новосибирский государственный технический университет, 2018. – 256 с. – URL: https://biblioclub.ru/index.php?page=book&id=573827 (дата обращения: 04.02.2022). – Режим доступа : по подписке. – Текст : электронный.

4. Управление данными : учебник / Ю. Ю. Громов, О. Г. Иванова, А. В. Яковлев, В. Г. Однолько. – Тамбов : Тамбовский государственный технический университет (ТГТУ), 2015. – 192 с. –URL: http://biblioclub.ru/index.php?page=book&id=444642 (дата обращения: 15.09.2022). – Режим доступа : по подписке. – Текст : электронный.

5. Лапина, Т. И. Управление данными : учебное пособие / Т. И. Лапина ; Юго-Зап. гос. ун-т. - Курск : ЮЗГУ, 2011. - 255 с. - Текст : электронный.