APLANA
QUALITY SERVICES

Сергей Воробьёв
Ведущий инженер-тестировщик

SQL. Базовый курс

Содержание

APLANA
QUALITY SERVICES

SQL. Базовый курс

Часть 1. Введение в SQL

Введение в SQL

SQL (англ. Structured Query Language – «язык структурированных запросов») – универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных.

Введение в SQL

- **База данных** – список или множество связанных списков с информацией

- **Система управления базами данных (СУБД)** – специальный софт, управляющий этими списками

Реляционные и нереляционные БД

Реляционная БД – база данных, основанная на реляционной модели данных:

- Данные в базе представляют собой набор таблиц;
- Данные удовлетворяют определенным условиям целостности;
- Поддерживает операторы манипулирования таблицами

(например, выборка или копирование таблицы).

Нереляционные базы данных – иерархические, сетевые, объектно-ориентированные, NoSQL.

Чем БД отличаются от электронных таблиц

1. Хранение большого количества строк

- В электронных таблицах количество строк ограничено.
- В БД хранятся миллионы строк.

2. Одновременное обслуживание многих пользователей

3. Безопасность.

- Пользователям предоставляются привилегии только на определенные таблицы и действия.

4. Реляционные свойства.

- Данные хранятся в разных таблицах, между таблицами существуют связи.

5. Ограничения, гарантирующие качество данных.

Таблица (table)

Строка(row) – горизонтальный ряд ячеек, отведенный для каждого объекта таблицы.

Запись (record) – данные в строке.

Столбец(column) – содержит информацию одного типа.

Поле(field) – пересечение столбца и строки.

PEOPLE_ID	FIRST_NAME	LAST_NAME	BIRTHDAY	PHONE_NUMBER
1	Иванов	Иван	17-FEB-1983	99-99-999
2	Петров	Петр	23-APR-1987	22-22-222
3	Сидорова	Ульяна	02-SEP-1995	33-33-333

Основы интерфейса SQL

СУБД	Название	Расшифровка
InterBase/FireBird	PSQL	Procedural SQL
IBM DB2	SQL PL	SQL Procedural Language
Ms SQL Server/Sybase ASE	T-SQL	Transact-SQL
MySQL	SQL/PSM	SQL/Persistent Stored Module
Oracle	PL/SQL	Procedural Language/SQL (основан на языке Ada)
PostgreSQL	PL/pgSQL	Procedural Language/PostgreSQL (очень похож на Oracle PL/SQL)

Различия синтаксиса функций СУБД

MSSQL	Oracle	DB2
NUMERIC	NUMBER	NUMERIC
DATEADD	MONTH_ADD	(Date + 10 DAYS)
EOMONTH	LAST_DAY	LAST_DAY
DATEDIFF	MONTH_BETWEEN	TIMESTAMPDIFF
UPPER	UPPER	UCASE/UPPER
CONVERT	TO_CHAR	TO_CHAR
-	INITCAP	INITCAP

Синтаксис SQL

- Функции и названия объектов нечуствительны к регистру: SELECT = sELeCt.
- Однако при поиске по текстовым полям регистр учитывается
- SQL не чувствителен к переносу строк
- Отсутствуют обязательные символы, завершающие строки
- Поддерживаются --однострочные комментарии и /*многострочные */
- Каждую транзакцию принято завершать точкой с запятой, но при выполнении отдельных команд их употребление не обязательно

Типы данных

CHAR(n) – строки постоянной длины (до 256 байтов в MS SQL Server), т.е. ввели меньше данных в строку – размер не изменится

VARCHAR(n) – строки переменной длины, т.е. требует памяти столько, сколько данных

INTEGER – число без десятичной точки

NUMERIC (m,n) – используется для хранения нуля и положительных или отрицательных чисел с фиксированной и плавающей точкой. M-ТОЧНОСТЬ (общее число цифр), n – МАСШТАБ (число цифр справа от десятичной точки). m/n –необязательные параметры

DATE - дата в формате yyyy-mm-dd (ISO), dd/mm/yyyy (ANSI), dd-MON-yy.

BOOLEAN – логический тип данных: true/false или 1/0.

Также значением поля может быть ***NULL*** – означает отсутствие значений – пустую ячейку.

Тип DATE

По умолчанию можно представлять в базе данных дату в *формате DD-MON-YYYY* (например, ‘01-FEB-1900’):

INSERT INTO table1 (id, date_work) values (1, ‘01-FEB-1900’).

Также можно использовать ключевое слово DATE. При этом уже для формата даты *YYYY-MM-DD* (например, ‘1900-02-01’):

INSERT INTO table1 (id, date_work) values (1, DATE ‘1900-02-01’).

Также альтернативно можно использовать тип даты + время TIMESTAMP для задания уже не только даты, но и времени:

INSERT INTO table1 (id, date_work) values (1, ‘01-FEB-1900-10.50.01’),

т.е.

1 февраля 1900 10 часов 50 минут и 1 секунда (формат dd-MON-yy -hh.mm.ss.nnnnn)

Преобразование типов данных в MSSQL

CONVERT(тип данных, строка, стиль) – преобразование одного формата данных в другой

В символы:

CONVERT(VARCHAR(20), GETDATE())

В дату

CONVERT(DATETIME, '14-11-2015', 105)

В число

CONVERT(NUMERIC, ‘1234657890')

Преобразование типов данных в Oracle

TO_CHAR(входное значение, формат) – преобразование даты, числа, времени в строку.
Формат: ’MONTH DD’, ‘MONTH DD, YYYY’, ‘DD/MM/YYYY’, ‘DAY MON, YY AD’

YEAR
$9,999.00
И т. д.

TO_CHAR(SYSDATE, 'MONTH DD')

TO_DATE(входное значение, формат) – преобразование строки в дату. Формат: DD-MON-YYYY

Month dd, YYYY , HH:MI p.m.
И т. д.

TO_DATE('02-JAN-2012', 'DD-MON-YYYY')

TO_NUMBER(входное значение, формат) – преобразование строки в число.

TO_NUMBER('123')

Другие объекты базы данных

Представление (view) – это объекты БД, которые не содержат собственных таблиц, но их содержимое берется из других таблиц или представлений посредством выполнения запроса.

Схема (schema) – поименованная группа связанных объектов БД.

Индекс (index) – объект, создаваемый для повышения производительности Поиска. Скрытая таблица, содержащая один или несколько важных столбцов таблицы и указатели на строки таблицы.

Ограничение (constraint) – условия, которым должны удовлетворять введенные пользователем записи.

Другие объекты базы данных

Хранимая процедура (stored procedure) – объект базы данных, представляющий собой набор SQL-инструкций. Хранится в БД. Вызов процедуры приводит к выполнению содержащихся в ней инструкций.

Функция (function) – похожа на хранимую процедуру, но возвращает значение, которое может быть использовано в более крупном операторе.

Триггер (trigger) – процедура, которая выполняется автоматически, когда происходит некоторое заданное событие.

Курсор (cursor) – ссылка на контекстную область памяти. Используя курсор, можно отдельно обрабатывать каждую строку связанного с ним SQL-оператора.

Разделы языка SQL

1. **DDL** - Data Defenition Language (язык определения объектов БД). CREATE, ALTER, DROP и тд
2. **DCL** - Data Control Language (язык управления данными). GRANT, REVOKE
3. **DML** - Data Manipulation Language (язык манипулирования данными). INSERT, UPDATE, DELETE
4. **Data Retrieval** - выборка данных SELECT
5. **Transaction Control** (язык поддержания процесса транзакций). COMMIT, ROLLBACK, SAVEPOINT.

APLANA
QUALITY SERVICES

SQL. Базовый курс

Часть 2. Data Defenition Language

Data Defenition Language

1. *CREATE TABLE (создание таблиц)*

Общий синтаксис:
CREATE TABLE имя_таблицы (
поле1 Тип поля1,
поле2 Тип поля2,
…, полеN Тип поляN);

CREATE TABLE person_info (
person_id **INTEGER NOT NULL**,
first_name **VARCHAR(15) NOT NULL**,
last_name **VARCHAR(20) NOT NULL**,
gender **CHAR(1)**,
birthday **DATE**,
salary **NUMERIC(7,2));**

Data Defenition Language

2. ***ALTER TABLE*** *(изменение таблиц)*

ALTER TABLE имя_таблицы {**ADD** <имя столбца> <определение столбца>}| {**MODIFY** <имя столбца> <Определение столбца>}| {**DROP COLUMN** <имя столбца>}

3. ***DROP TABLE*** *(удаление таблиц)*

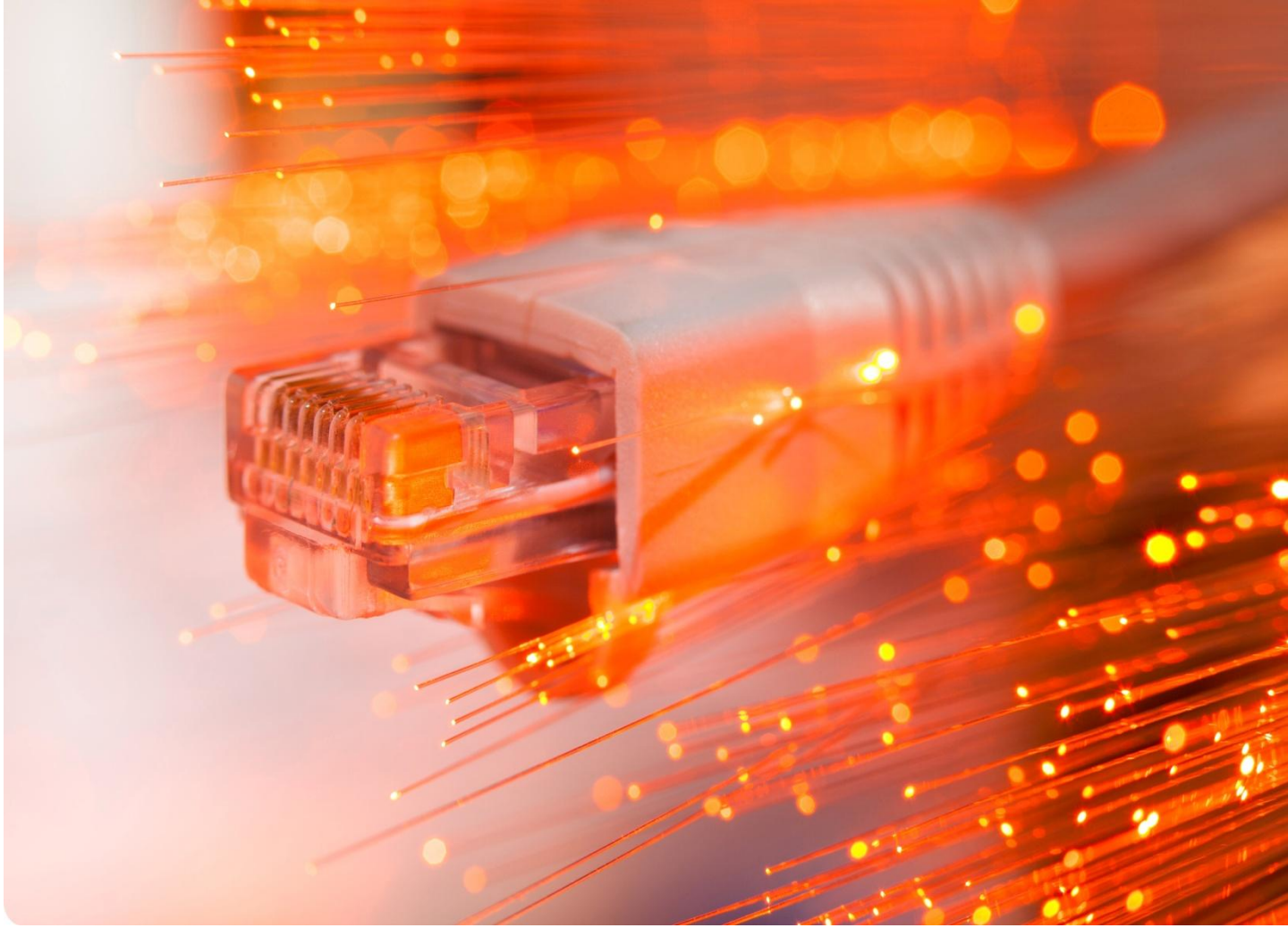
DROP TABLE имя_таблицы {**CASCADE CONSTRAINTS**};

4. ***TRUNCATE TABLE*** *(очистка таблиц)*

TRUNCATE TABLE имя_таблицы

APLANA
QUALITY SERVICES

SQL. Базовый курс



Часть 3. Data Manipulation Language

Data Manipulation Language

1. ***INSERT*** *- Вставка отдельной записи.*

- **INSERT INTO** имя_таблицы **VALUES** (значение поля1, значение поля2,.., значение поляN);
- **INSERT INTO** имя_таблицы (поле1, поле3,…) **VALUES** (значение поля1, значение поля2,..., значение поляN);

INSERT INTO person_info **VALUES** (1, 'John', 'Smith', 'M', '15-OCT-1973', 45568.56);
INSERT INTO person_info (person_id, first_name, last_name) **VALUES** (5, Sarah', ‘Connor');
Успешно.

INSERT INTO person_info **VALUES** (NULL, 'Jane', 'Smith', 'F', '8-AUG-1987', NULL);
Ошибка, т.к person_id не может быть NULL.

Data Manipulation Language

Вставка группы записей

INSERT INTO имя_таблицы
SELECT…;

CREATE TABLE t2 (
first_1 **VARCHAR(15),**
last_1 **VARCHAR(20)**,
birthday_1 **DATE**);

INSERT INTO t2
SELECT first_name, last_name, birthday
FROM person_info;

Data Manipulation Language

INSERT INTO person_info **VALUES** (2, 'Sara', 'Doe', 'F', '9-OCT-1986', 29789.56);

INSERT INTO person_info **VALUES** (2, 'Rita', 'Blow', 'F', '9-OCT-1975', 29789.56);

Успешно.

INSERT INTO person_info **VALUES** (3, 'Sara', 'Doe', 'F', '9-OCT-1986', 29789.56);

Теперь удалим вторую запись с person_id=2

DELETE FROM person_info **WHERE** person_id = 2

Data Manipulation Language

Целостность данных

Целостность сущностей - определяет строку таблицы как уникальный экземпляр некоторой сущности.

Первичный ключ (primary key) - столбец или группа столбцов уникально идентифицирующий каждую запись.

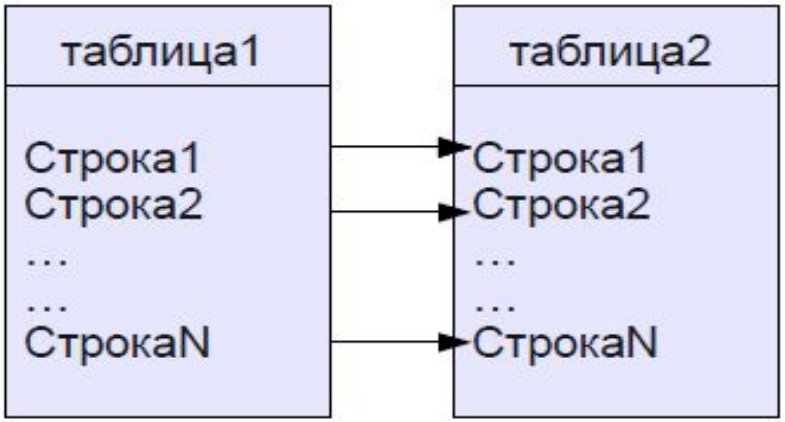
Внешний ключ (foreign key) – отражение связей между таблицами. Подчиненная таблица должна иметь идентичный столбец (или группу столбцов) для хранения значений, уникально идентифицирующих главные записи.

Ссылочная целостность – в подчиненных таблицах не должно быть записей, ссылающихся на несуществующие записи главных таблиц.

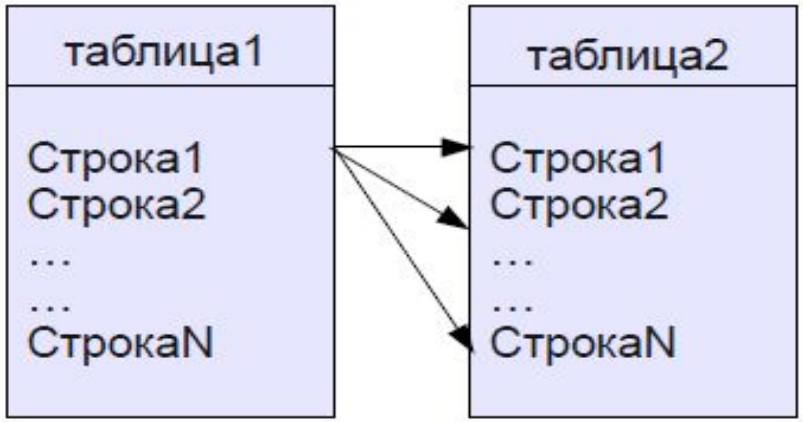
Data Manipulation Language

Данные хранятся в разных таблицах,
между таблицами существуют связи (relationships).

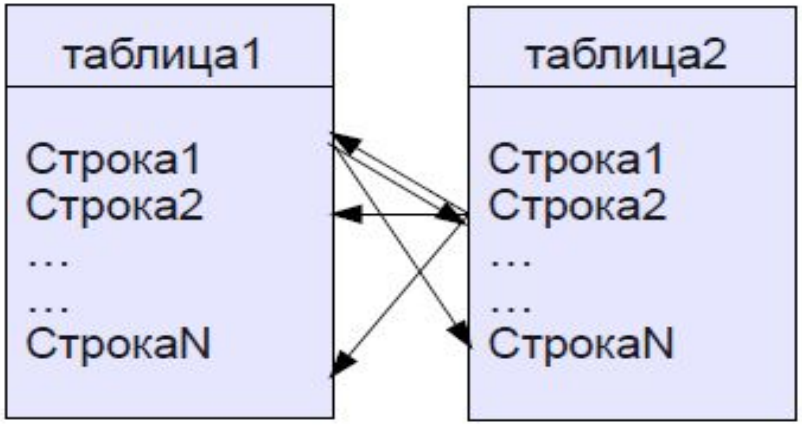
1 : 1



1 : n



m : n



APLANA QUALITY SERVICES

Data Manipulation Language

Первичный ключ

ALTER TABLE имя_таблицы
ADD PRIMARY KEY (имя_столбца);

ALTER TABLE person_info
ADD PRIMARY KEY(person_id);

Значения первичного ключа подразумевают уникальную идентификацию записи, соответственно, значения не могут повторяться.

И опять попытаемся добавить запись с person_id=2:
INSERT INTO person_info **VALUES** (2, 'Rita', 'Blow', 'F', '9-OCT-1975', 29789.56);

Внешний ключ

ALTER TABLE имя_подчиненной_таблицы
ADD CONSTRAINT имя_ограничения **FOREIGN KEY** (имя_столбца подчиненной таблицы) **REFERENCES** имя_главной_таблицы;

CREATE TABLE person_address (
person_id **INTEGER**,
address **VARCHAR(200)**);

ALTER TABLE person_address
ADD CONSTRAINT person_fk_address
FOREIGN KEY (person_id)
REFERENCES person_info;

Data Manipulation Language

INSERT INTO person_address **VALUES** (1, 'Moscow, Arbat street, 67-14');
INSERT INTO person_address **VALUES** (2, 'Moscow, Arbat street, 67-14');
Успешно.

INSERT INTO person_address **VALUES** (4, 'Zelenograd,Green street, 23');
Ошибка. Попытка вставить подчиненную запись при отсутствии соответствующей главной записи.

INSERT INTO person_address **VALUES** (3, 'Zelenograd,Green street, 23');

Связывание таблиц при создании

Как мы уже рассмотрели ранее, широко используется создание первичного (*PRIMARY KEY*) и внешнего (*FOREIGN KEY*) ключей через команды изменения структуры существующих таблиц. Также можно добавлять эти конструкции и при создании таблицы:

```
CREATE TABLE tab1(
id integer PRIMARY KEY,
…..
```

Data Manipulation Language

2. UPDATE - Изменение значений столбцов таблицы

А)Изменение всех значений столбца таблицы

UPDATE <table_name>
SET <column_name> = <value>

UPDATE person_address
SET address = 'Volgograd, First street, 15-20'

Б)Изменение конкретных значений таблицы

UPDATE <table_name>
SET <column_name> = <value>
WHERE <column_name> = <value>

UPDATE person_address **SET** address = 'Volgograd, First street, 15-20'
WHERE person_id = 3;

UPDATE <table_name>
SET <column_name> = <value>
WHERE <column_name> = <column_name> [оператор]

UPDATE person_info **SET** salary = salary * 2
WHERE person_id = 3;

Data Manipulation Language

3. DELETE - Удаление строк из таблицы

А) Удаление всех значений столбца таблицы

DELETE FROM <table_name>

Б) Удаление конкретных значений таблицы

DELETE FROM <table_name>
WHERE <column_name> = <value>

Практическое задание № 1

1. Создать БД, изображенную на рис.1 (создать таблицы и внешний ключ)
2. Внести в таблицы следующие данные.

Dept: (1, 'Marketing'), (2, 'RD')

Emp: (1, 1, 'James', 1000), (2, 2, 'Smith', 2000)

3. Создать таблицу dept_arch с такой же структурой, как и у таблицы dept.
4. Вставить в таблицу dept_arch все данные из таблицы dept.

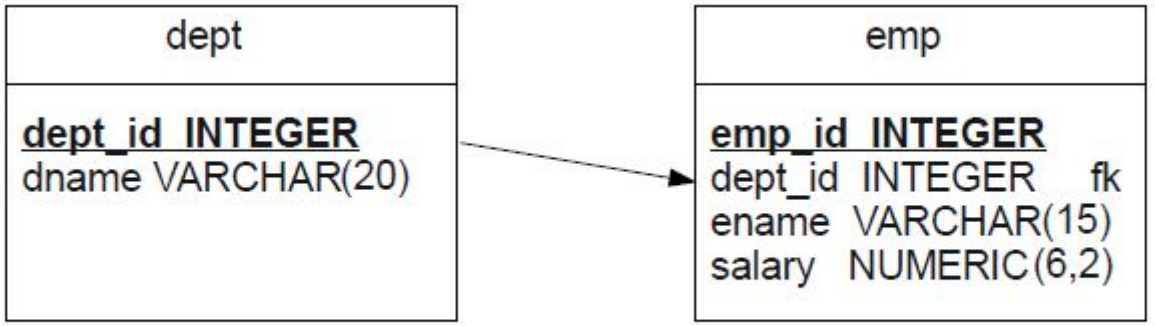


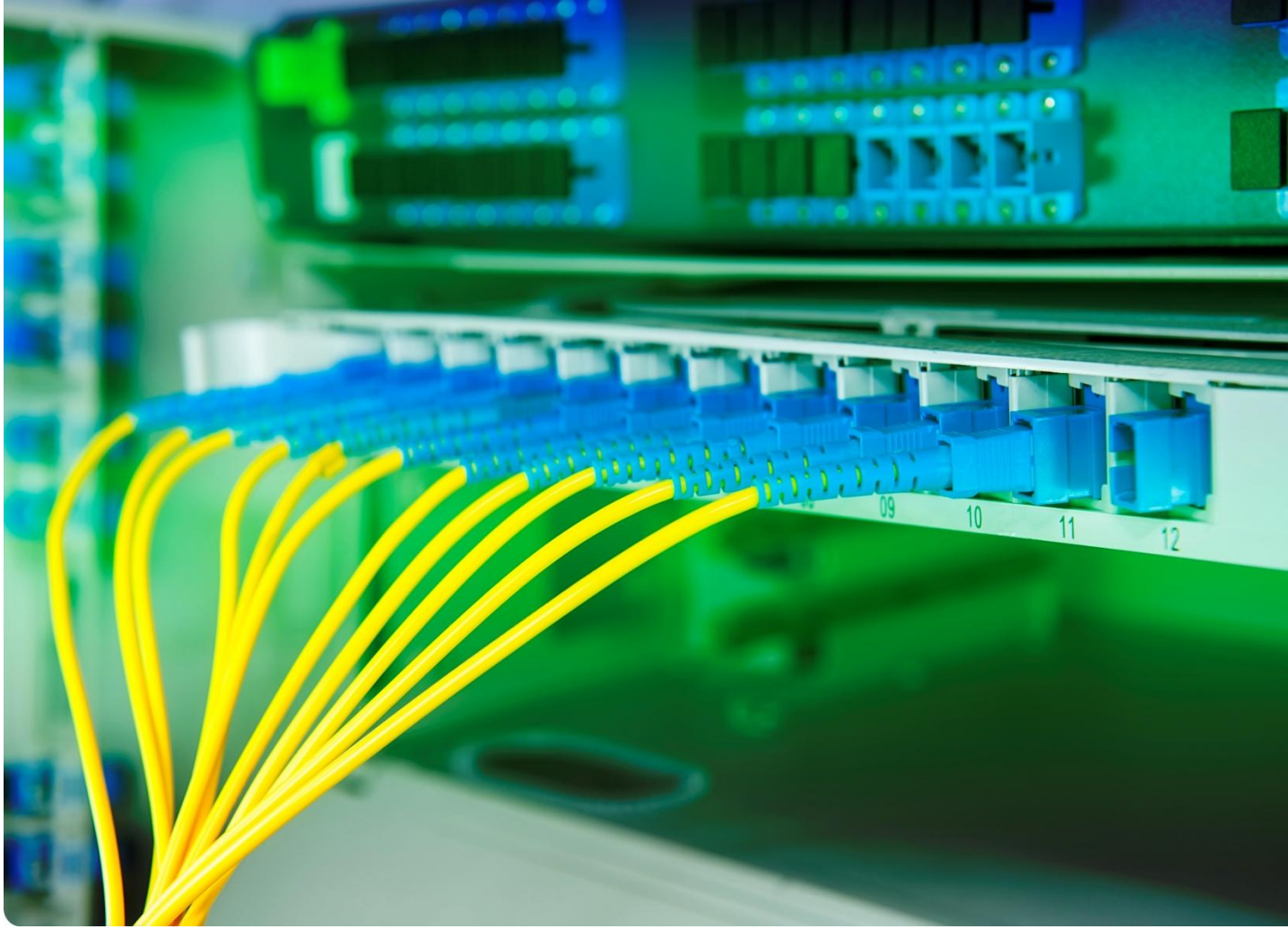
Рис. 1

Практическое задание № 1 (продолжение)

5. Увеличьте на 15% зарплату сотруднику Smith.
6. Убедитесь, что в таблицу dept нельзя вставить такую запись: (2, 'Sales'). Почему?
7. Убедитесь, что в таблицу emp нельзя вставить такую запись: (3, 4, 'Black', 3000, 'Active'). Почему?
8. Измените название отдела RD на RandD (таблица dept).
9. Удалите из таблицы emp запись с emp_id = 1.
12. Удалите из таблицы emp все записи.
13. Удалите таблицу emp.

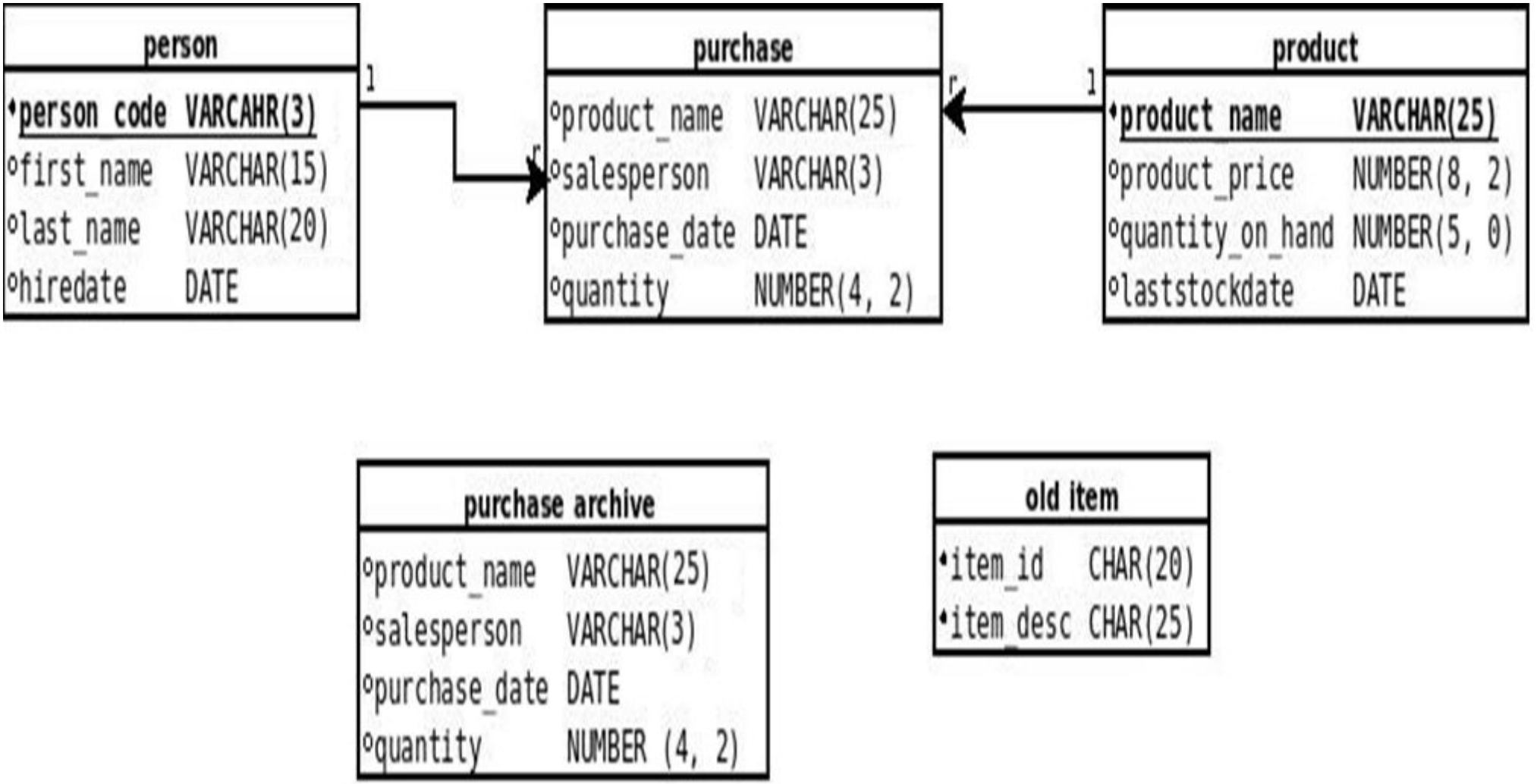
APLANA
QUALITY SERVICES

SQL. Базовый курс



Часть 4. DRL. Простые запросы

Наша учебная БД



APLANA QUALITY SERVICES

Data Retrieval Language

SELECT – выборка данных. Этот раздел является обязательным в запросе и позволяет:

SELECT поле1,…полеN ***FROM*** таблица1, .., таблицаN ***WHERE*** условие

- Определить список выходных столбцов
- Включить вычисляемые столбцы
- Включить константы
- Переименовать выходные столбцы
- Указать принцип обработки дублей строк
- Включить агрегатные функции

Data Retrieval Language

1. *Определение списка выходных столбцов*

Список выходных столбцов может быть указан несколькими способами:

- Указать символ *, обозначающий включение в результаты запроса всех колонок запроса в естественной последовательности.
- Перечислить в желательном порядке только нужные <имена столбцов>.

SELECT person_code, first_name, last_name **FROM** person;

--Можем менять порядок столбцов

SELECT first_name, last_name, person_code **FROM** person;

Конкатенация

Соединение двух и более частей текста.

SELECT product_name + ' was sold by ' + salesperson **FROM** purchase;

column1
Small Widget was sold by CA
Medium Widget was sold by BB
Chrome Phoobar was sold by GA
Small Widget was sold by GA
Medium Widget was sold by LB
Round Chrome Snaphoo was sold by CA

2. *Включение вычисляемых столбцов*

В качестве вычисляемых столбцов запроса могут выступать:

- Результаты простейших арифметических выражения (+, -, /, *_ или конкатенации строк (+).

- Результаты функций агрегирования {AVG|SUM|MAX|MIN|COUNT}

3. ***Включение констант***

В качестве столбцов могут выступать константы числового и символьного типов.

```
SELECT 'Есть такой код',person_code, 'для', first_name, last_name
    FROM person
```

Data Retrieval Language

4. Переименование выходных столбцов

Вычисляемым, а также любым другим столбцам, при желании,
можно присвоить уникальное имя с помощью ключевого слова
AS: <выражение> AS <новое имя>

```
SELECT product_name + ' was sold by ' + salesperson AS SOLDBY
FROM purchase;
```

Можно задавать псевдонимы и без использования ключевого слова, но с ограничениями:

```
SELECT product_name + ' was sold by ' + salesperson SOLDBY
FROM purchase;
SELECT product_name + ' was sold by ' + salesperson "Sold By"
FROM purchase;
```

5. *Указывание принципа обработки дублей*

- **DISTINCT** – запрещает появление строк-дублей в выходном множестве. Его можно задавать один раз для оператора SELECT. На практике первоначально формируется выходное множество, упорядочивается, а затем из него удаляются повторяющиеся значения. Обычно это занимает много времени и не следует этим злоупотреблять.

SELECT DISTINCT * FROM *person*

- **ALL** (действует по умолчанию) – обеспечивает включение в результаты запроса и повторяющихся значений

6. Включение агрегатных функций

Функции агрегирования (функции над множествами, статистические или базовые) предназначены для вычисления некоторых значений для заданного множества строк. Используются следующие агрегатные функции:

AVG|SUM(<выражение) – подсчитывает среднее значение | сумму от <выражение>.

MIN|MAX(<выражение>) – находит максимальное | минимальное значение.

COUNT(*|[DISTINCT] <имя столбца>) – подсчитывает число строк

Но об этом далее

Data Retrieval Language

WHERE – выборка данных, которые удовлетворяют определенным условиям.

SELECT поле1,…полеN ***FROM*** таблица1, .., таблицаM ***WHERE*** условие1,…условиеY

Data Retrieval Language

Примеры:

- **SELECT** * **FROM** product **WHERE** laststockdate **IS NULL**;

- **SELECT** * **FROM** product **WHERE** laststockdate **IS NOT NULL**;

- **SELECT** product_name, product_price, quantity_on_hand
FROM product **WHERE** quantity_on_hand > 150;

- **SELECT** product_name, product_price **FROM** product
WHERE product_name <> 'Square Zinculator';

Data Retrieval Language

Есть и более сложные условия:

Попадания во множество

<конструктор значений строки> [**NOT**] **IN** (<подзапрос>|<набор конструкторов значений строки>)

Определяется множество значений, которому объект сравнения, записанный до ключевого слова IN, может принадлежать или не принадлежать. Если подзапрос не возвращает строк, то предикат принимает значение FALSE.

Примеры на работу со множествами:

SELECT * FROM purchase **WHERE** salesperson **IN** ('CA', 'BB');

SELECT * FROM purchase **WHERE** salesperson **NOT IN** ('CA', 'BB');

SELECT * FROM purchase **WHERE** (salesperson + product_name) **in** (('CA' + 'Small Widget'), ('GA' + 'Chrome Phoobar'))

Data Retrieval Language

Принадлежности диапазону

<конструктор значений строки> [**NOT**] **BETWEEN** <конструктор значений строки 1> AND <конструктор значений строки 2>

Предикат **BETWEEN** сходен с предикатом IN, но вместо элементов множества он задает включающие границы, в которые [не] должно попадать проверяемое значение.

SELECT product_name, product_price **FROM** product **WHERE** product_price **NOT BETWEEN** 1 **AND** 80;

Data Retrieval Language

Булевы операторы

<предикат> {**AND**|**OR**|**NOT**} <предикат>

Примечания: булевы оператора связывают один или несколько предикатов, образуя единственное логическое значение **TRUE**|**FALSE**. Используя предикаты с булевыми операторами, можно значительно увеличить и избирательную способность по отбору строк в результат запроса.
При использовании булевых операторов, особенно оператора **NOT**, следует применять круглые скобки для правильного составления условий (**AND** выполняется раньше **OR**).

SELECT product_name, product_price **FROM** product **WHERE** product_name **LIKE** '%Widget' **OR** product_price < 20;

SELECT product_name, product_price **FROM** product **WHERE** product_name **LIKE** '%Widget' **AND** product_price < 20;

Data Retrieval Language

Оператор примерного поиска LIKE

SELECT список полей **FROM** список таблиц **WHERE** проверяемое значение **LIKE** (шаблон) (**ESCAPE** (имя пропуска));

✔ *Один любой символ* - _

SELECT person_code, first_name, last_name **FROM** person **WHERE** person_code **LIKE** '_A';

✔ *Любая подстрока* - %

SELECT product_name **FROM** product **WHERE** product_name **LIKE** '%Chrome%';

Если надо найти текст с символом % (например, название продукта ab%cdef):

WHERE product_name **LIKE** 'ab$%c%' **ESCAPE** '$';

Первый % читается как символ в названии, второй – как любая строка.

Data Retrieval Language

Оператор примерного поиска LIKE

… where отчество like ‘%ов%’

id	Имя	Отчество	Фамилия
1	Иван	Иванович	Иванов
2	Роман	Ибрагимович	Шведов
3	Евгений	Сидорович	Сидоров
4	Марий	Петровна	Шпак

… where отчество like ‘И%’

id	Имя	Отчество	Фамилия
1	Иван	Иванович	Иванов
2	Роман	Ибрагимович	Шведов

… where отчество like ‘%вич’

id	Имя	Отчество	Фамилия
1	Иван	Иванович	Иванов
2	Роман	Ибрагимович	Шведов
3	Евгений	Сидорович	Сидоров

… where Фамилия like ‘___ов’

id	Имя	Отчество	Фамилия
1	Иван	Иванович	Иванов
2	Роман	Ибрагимович	Шведов

Data Retrieval Language

Оператор примерного поиска LIKE

```
select product_name from purchase
```

PRODUCT_NAME
Small%Widget
Medium Widget
Chrome Phoobar
Small%Widget
Medium Widget
Round Chrome Snaphoo

```
select product_name from purchase
where product_name like '%Widget'
```

PRODUCT_NAME
Small%Widget
Medium Widget
Small%Widget
Medium Widget

```
select product_name from purchase
where product_name like '%$%Widget'
escape '$'
```

PRODUCT_NAME
Small%Widget
Small%Widget

Сортировка

SELECT список столбцов **FROM** список таблиц **WHERE** условие **ORDER** BY список столбцов **ASC (DESC)**;

По убыванию:

SELECT product_name, product_price **FROM** product
ORDER BY product_price **DESC**;

По возрастанию:

SELECT product_name, product_price **FROM** product
ORDER BY product_name **ASC**;

Практическое задание № 2

1. Напишите запрос, полностью показывающий таблицу purchase.
2. Напишите запрос, выбирающий столбцы product_name и quantity из таблицы Purchase.
3. Напишите запрос, выбирающий эти столбцы в обратном порядке.
4. Напишите запрос, выводящий для каждой строки таблицы person следующий текст:

<**first_name**> <**last_name**> started work <**hiredate**>*. Получаемому столбцу присвоить псевдоним “Started Work”.

5. Напишите запрос,выводящий наименование продуктов product_name (таблица product), для которых цена не определена (NULL).
6. Напишите запрос, выводящий наименование продуктов product_name (таблица purchase), которых продали от 3 до 23 штук.

* MSSQL не поддерживает объединение столбцов с типами данных varchar и date. Используйте оператор конветации: CONVERT(VARCHAR, hiredate)

APLANA QUALITY SERVICES

Практическое задание № 2 (продолжение)

7. Напишите запрос, выводящий фамилии сотрудников, которых приняли на работу 1го, 15го и 28го февраля 2010 года.

8. Напишите запрос, выводящий наименование продуктов product_name (таблица purchase), проданных сотрудниками, фамилии которых начинаются на “B”.

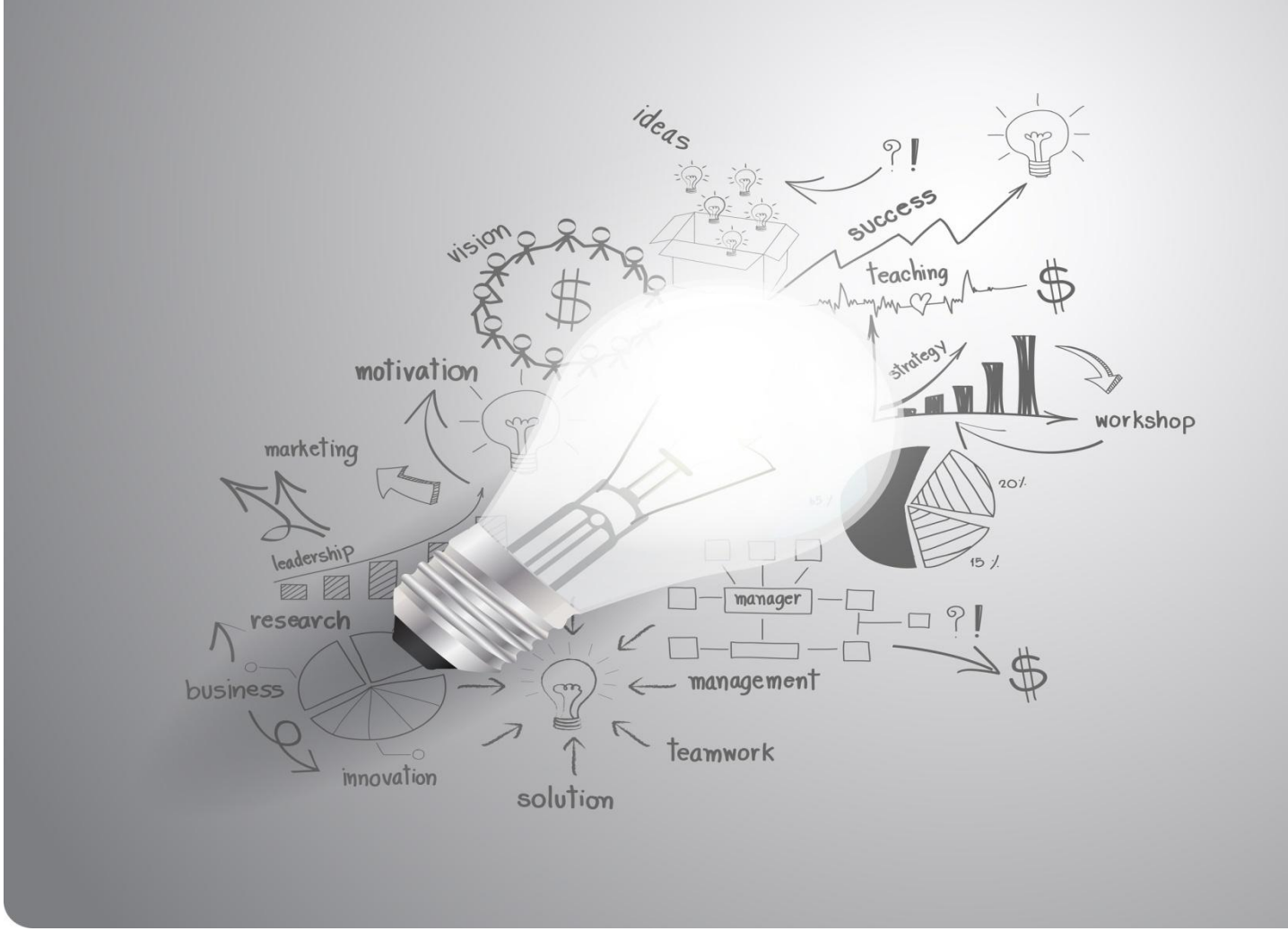
9. Напишите запрос, выводящий наименование продуктов product_name (таблица purchase), проданных сотрудниками, фамилии которых не начинаются на “B”.

10. Напишите запрос, выводящий фамилии и дату приема на работу сотрудников, фамилии которых начинаются на “B” и которых приняли на работу раньше 1 марта 2010 года.

11. Напишите запрос, выводящий наименование продуктов product_name и дату последней поставки laststockdate (таблица product), наименование которых Small Widget, Medium Widget и Large Widget или те, для которых не указана дата последней поставки. Отсортируйте по убыванию даты последней поставки.

APLANA
QUALITY SERVICES

SQL. Базовый курс



Часть 5. Выборка данных из нескольких таблиц

Выборка данных из нескольких таблиц

SELECT *имя_таблицы_1*.имя_столбца, *имя_таблицы_2*.
имя_столбца
FROM *имя_таблицы_1, имя_таблицы_2*;

SELECT purchase.product_name, person.first_name, person.last_name
FROM purchase, person;

Декартово произведение (Cartesian product) - соединение без конструкции WHERE, в результате которого каждая строка одной таблицы комбинируется с каждой строкой другой таблицы.

Выборка данных из нескольких таблиц с условием

SELECT имя_таблицы_1.имя_столбца, имя_таблчцы_2. имя_столбца
FROM имя_таблицы_1, имя_таблицы_2
WHERE имя_главной_таблицы.первичный_ключ =
имя_подчиненной_таблицы.внешний_ключ;

SELECT purchase.product_name, person.first_name, person.last_name
FROM purchase, person
WHERE person.person_code = purchase.salesperson;

Типы соединения

Существуют также иные способы соединения таблиц по ключам:

- <таблица A> [<тип соединения>] **JOIN** <таблица B> **ON** <предикат>

- <тип соединения> представляет собой один из аргументов:
  INNER|{**LEFT**|**RIGHT**|**FULL**[**OUTER**]}

- **INNER** – включает строки, в которых есть столбцы с совпадающими данными объединяемых таблиц. Используется по умолчанию.
- **LEFT**[**OUTER**] – включает все строки таблицы A (левая таблица) и все совпадающие значения из таблицы B. Столбцы несовпадающих строки заполняются NULL-значениями.
- **RIGHT**[**OUTER**] – включает все строки таблицы B (правая таблица) и все совпадающие значения таблицы A. обратный вариант для левого объединения.
- **FULL**[**OUTER**] – включает все строки обеих таблиц. Столбцы совпадающих строк заполнены реальными значениями, а несовпадающих строк – NULL-значениями.
- **OUTER** (внешний) – уточняющее слово, означающее, что несовпадающие строки из ведущей таблицы включаются вместе с совпадающими.

Варианты соединения таблиц

address

ClientID	Address
1	aaa
2	bbb
3	ccc
5	eee
8	hhh

phone

ClientID	phone
1	1111111
3	333333
4	333333
6	6666666
8	888888

INNER JOIN

SELECT * FROM address INNER JOIN phone ON address.ClientID=phone.ClientID

	ClientID	Address	ClientID	phone
1	1	aaa	1	1111111
2	3	ccc	3	333333
3	8	hhh	8	888888

Варианты соединения таблиц

address

ClientID	Address
1	aaa
2	bbb
3	ccc
5	eee
8	hhh

phone

ClientID	phone
1	1111111
3	333333
4	333333
6	6666666
8	888888

SELECT * FROM address, phone WHERE address.clientID=phone.ClientID

	ClientID	Address	ClientID	phone
1	1	aaa	1	1111111
2	3	ccc	3	333333
3	8	hhh	8	888888

Варианты соединения таблиц

address

ClientID	Address
1	aaa
2	bbb
3	ccc
5	eee
8	hhh

phone

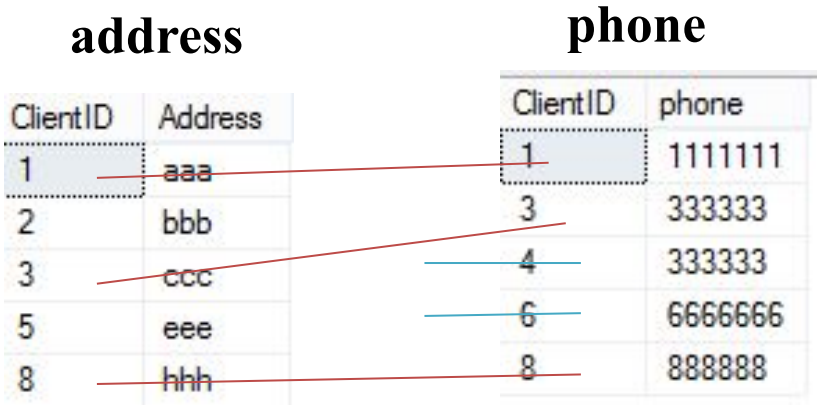
ClientID	phone
1	1111111
3	333333
4	333333
6	6666666
8	888888

LEFT JOIN

SELECT * FROM address LEFT JOIN phone ON address.ClientID=phone.ClientID

ClientID	Address	ClientID	phone
1	aaa	1	1111111
2	bbb	NULL	NULL
3	ccc	3	333333
5	eee	NULL	NULL
8	hhh	8	888888

Варианты соединения таблиц

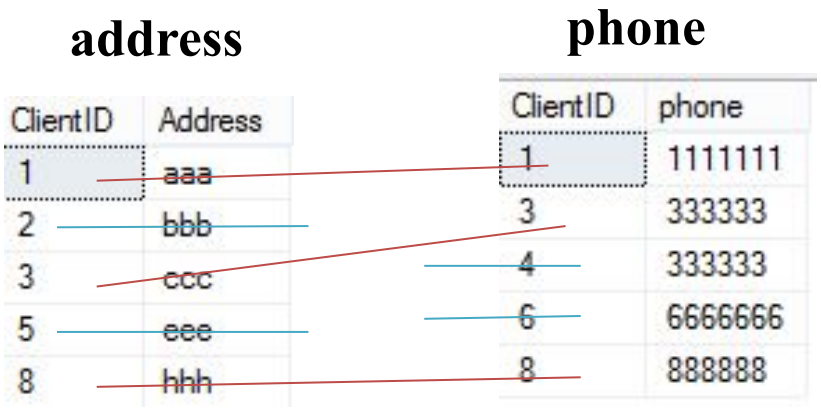


RIGHT JOIN

SELECT * FROM address RIGHT JOIN phone ON address.ClientID=phone.ClientID

ClientID	Address	ClientID	phone
1	aaa	1	1111111
3	ccc	3	333333
NULL	NULL	4	333333
NULL	NULL	6	6666666
8	hhh	8	888888

Варианты соединения таблиц

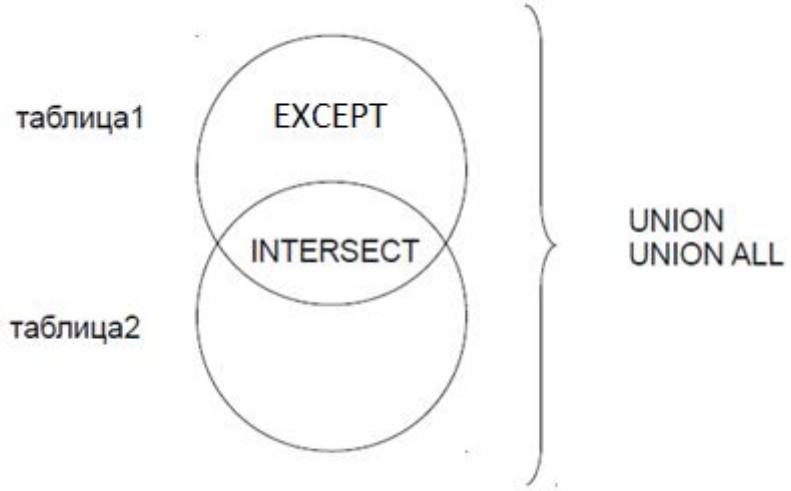


FULL JOIN

SELECT * FROM address FULL JOIN phone ON address.ClientID=phone.ClientID

ClientID	Address	ClientID	phone
1	aaa	1	1111111
2	bbb	NULL	NULL
3	ccc	3	333333
NULL	NULL	4	333333
5	eee	NULL	NULL
NULL	NULL	6	6666666
8	hhh	8	888888

Операторы соединения



UNION возвращает все строки из обоих операторов SELECT; повторяющиеся значения удаляются.

UNION ALL возвращает все строки из обоих операторов SELECT; повторяющиеся значения показываются.

INTERSECT возвращает строки, которые возвращены и первым, и вторым оператором SELECT.

EXCEPT возвращает строки, которые возвращены первым оператором SELECT, исключая те, которые возвращены вторым оператором.

Количество и порядок столбцов, возвращаемых SELECT из обеих таблиц, должны совпадать.

Операторы соединения

```
SELECT product_name
FROM purchase
ORDER BY product_name
```

product_name
Chrome Phoobar
Medium Widget
Medium Widget
Round Chrome Snaphoo
Small Widget
Small Widget

```
SELECT product_name
FROM purchase_archive
ORDER BY product_name
```

product_name
Chrome Phoobar
Large Harf linger
Medium Wodget
Round Snaphoo
Small Widget
Small Widget

```
SELECT product_name
FROM purchase
UNION
SELECT product_name
FROM purchase_archive
ORDER BY product_name
```

product_name
Chrome Phoobar
Large Harf linger
Medium Widget
Medium Wodget
Round Chrome Snaphoo
Round Snaphoo
Small Widget

Операторы соединения

```
SELECT product_name
FROM purchase
UNION ALL
SELECT product_name
FROM purchase_archive
ORDER BY 1
```

product_name
Chrome Phoobar
Chrome Phoobar
Large Harf linger
Medium Widget
Medium Widget
Medium Wodget
Round Chrome Snaphoo
Round Snaphoo
Small Widget
Small Widget
Small Widget
Small Widget

```
SELECT product_name
FROM purchase
EXCEPT
SELECT product_name
FROM purchase_archive
ORDER BY 1
```

product_name
Medium Widget
Round Chrome Snaphoo

```
SELECT product_name
FROM purchase
INTERSECT
SELECT product_name
FROM purchase_archive
ORDER BY 1
```

product_name
Chrome Phoobar
Small Widget

Псевдоним в области FROM

- При использовании больших баз со схемами принято использование псевдонимов:

```
SELECT purc.product_name, prod.laststockdate, pers.first_name,
pers.last_name
FROM purchase as purc,
        Person as pers,
        Product prod
WHERE pers.person_code = purc.salesperson AND
prod.product_name = purc.product_name;
```

Практическое задание № 3

1. Напишите запрос, выводящий декартово произведение таблиц product и purchase.

2. Напишите запрос, выводящий наименование проданного товара product_name, количество quantity (таблица purchase) и quantity_on_hand (таблица product).

3.Напишите запрос, выводящий наименование товара product_name (таблица purchase), дату последней поставки laststockdate (таблица product) и фамилию продавца last_name (таблица person).

4. Напишите запрос, выводящий столбцы product_name, first_name, last_name внешнего объединения таблиц purchase и person. Используйте для таблиц короткие псевдонимы.

Практическое задание № 3 (продолжение)

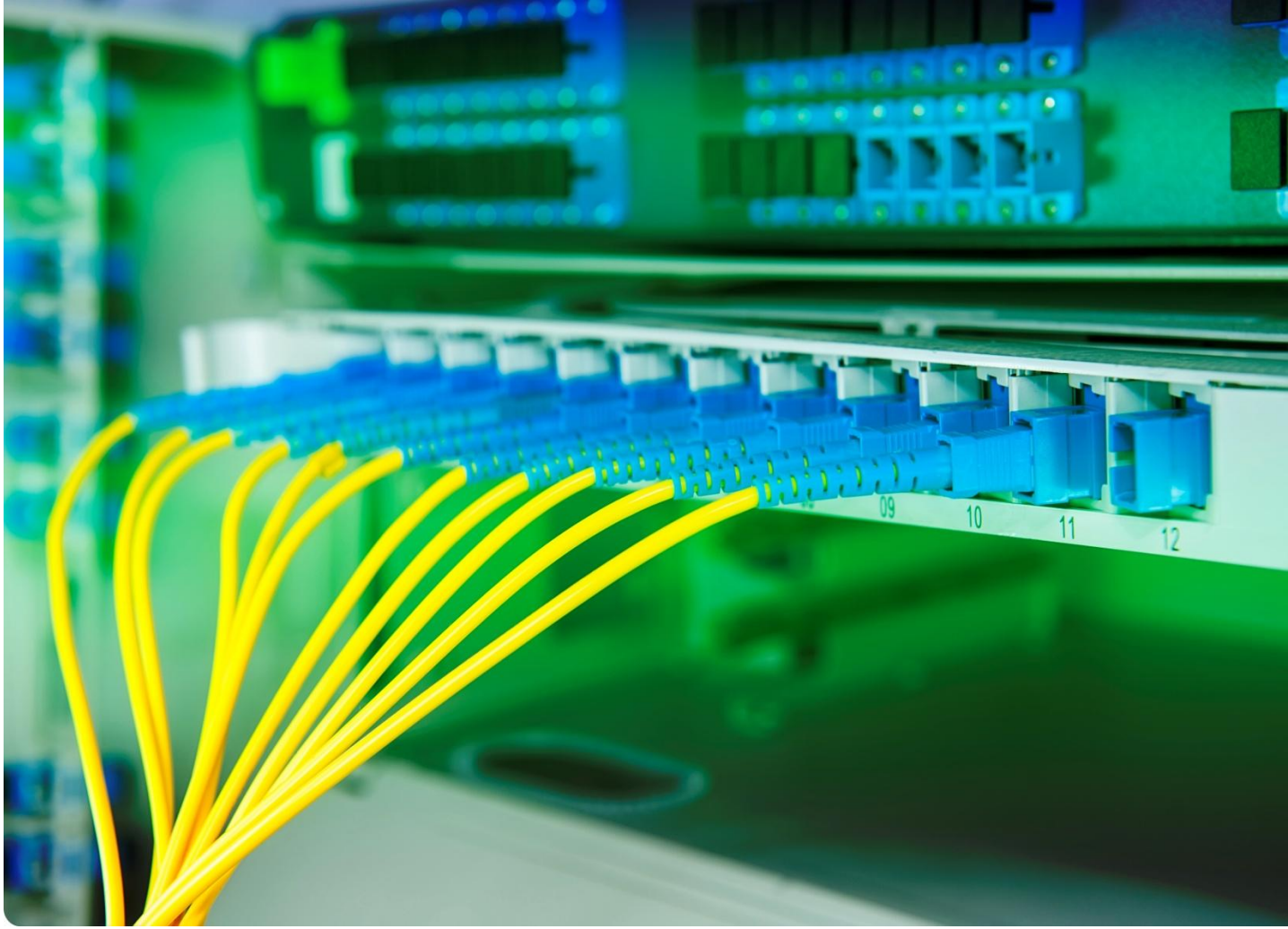
5. Напишите запрос, который выводит все неповторяющиеся в purchase коды продавцов
salesperson из таблицы purchase_archive.

6. Напишите запрос, который выводит коды только тех продавцов salesperson из таблицы purchase, которые так же содержаться в таблице purchase_archive.

7. Напишите запрос, который выводит все (в том числе повторяющиеся) коды продавцов salesperson из таблиц purchase и purchase_archive.

APLANA
QUALITY SERVICES

SQL. Базовый курс



Часть 6. Агрегатные функции. Группирование данных.

Математические операторы

Математический оператор – символы, обозначающие операции (+, -,*, /)

Вычисления с использованием данных из таблиц.

SELECT product_name, product_price * 1.07 **FROM** product;

SELECT product_name, product_price * quantity_on_hand
FROM product;

SELECT product_name, product_price * 1.07 * quantity_on_hand -
product_price * quantity_on_hand
FROM product;

SELECT product_name, product_price * (quantity_on_hand + 10)
FROM product;

Математические операторы

Функции агрегирования (функции над множествами, статистические или базовые) предназначены для вычисления некоторых значений для заданного множества строк.

1. **SUM** - суммирует значения и возвращает итог.

```
SELECT SUM(quantity)
FROM purchase;
```

2. **AVG** – возвращает среднее значение по указанному столбцу.

```
SELECT AVG(product_price)
FROM product;
```

Математические операторы

3. **MIN** – возвращает минимальное значение из указанного столбца.

SELECT MIN(product_price)
FROM product;

4. **MAX** - возвращает максимальное значение из указанного столбца.

SELECT MAX(product_price)
FROM product;

Математические операторы

5. **COUNT** – подсчитывает записи.

SELECT COUNT(*)

FROM purchase; --число строк с учетом NULL значений

SELECT COUNT(product_name)

FROM purchase;--значений в столбце, игнорируя NULL

GROUP BY

Этот раздел предназначен для объединения результатов запроса в группы и расчета для каждой из них статистических значений. Иногда используют термин «сгруппированная таблица».

SELECT product_name, **SUM**(quantity)
FROM purchase
GROUP BY product_name;

В оператор SELECT можно включить несколько групповых функций.

SELECT product_name, **SUM**(quantity) "Total Sold", **COUNT**(quantity) Transactions
FROM purchase
GROUP BY product_name;

HAVING

HAVING – является подразделом предназначенным для ограничения числа строк в сгруппированной таблице и является частью раздела GROUP BY. Предикат этого раздела строится по тем же семантическим правилам, что и в разделе WHERE, однако напрямую в предикате могут участвовать только те столбцы, которые указаны в раздел GROUP BY. Остальные можно использовать только внутри функций агрегирования. Этот раздел ограничивает состав групп (подгрупп) строк, на которые разбивается результат запроса. В группы (подгруппы) включаются только те из множества возможных строк, для значений которых выполняются условия предиката раздела HAVING. Внутри раздела HAVING можно использовать вложенные запросы с функциями агрегирования, а также связанные подзапросы.

HAVING

T.e., подведя итог выше описанного, можно сузить
назначение подраздела до:

С помощью конструкции HAVING можно

фильтровать группы.

HAVING работает для групп так же, как и WHERE для отдельных записей.

```
SELECT product_name, SUM(quantity) "Total Sold",
COUNT(quantity) Transactions
FROM purchase
GROUP BY product_name
HAVING SUM(quantity) < 5;
```

Практическое задание № 4

1. Напишите запрос, показывающий, какой будет цена продукта product_price после увеличения на 15%.

2. Напишите запрос, показывающий, сколько всего имеется товаров в таблице product.

3.Напишите запрос, показывающий, для какого количества товаров (таблица product) не указана цена.

4. Напишите запрос, выводящий минимальную и максимальную цену товаров product_price.

5. Напишите запрос, показывающий, какая сумма была выручена с продаж товаров каждого наименования.

6. Напишите запрос, показывающий, какая сумма была выручена с продаж товаров каждого наименования. Вывести только те записи, для которых сумма продаж больше 125.

APLANA
QUALITY SERVICES

SQL. Базовый курс

Часть 7. Подзапросы

Подзапросы

Подзапрос — это обычный запрос SELECT, вложенный в оператор SELECT, UPDATE или DELETE.

Он используется в качестве источника данных для раздела FROM или WHERE родительского оператора.

Подзапросы

Есть некие ограничения использования подзапросов:

- Подзапрос должен выбирать только один столбец (за исключением подзапроса с предикатом EXISTS), и тип данных его результата должен соответствовать типу данных значения, указанному в предикате.
- В ряде случаев можно использовать ключевое слово DISTINCT для гарантии получения единственного значения.
- Во вложенном запросе нельзя включать раздел ORDER BY и UNION.
- Подзапрос может находиться и слева и справа от условия поиска.
- В подзапросах могут использоваться функции агрегирования без раздела GROUP BY

Однострочные подзапросы

Однострочный подзапрос – это подзапрос, который возвращает лишь 1 значение. Используются символы сравнения с результатом вложенного запроса (=, <>, <, <=, >, >=)

```
SELECT * FROM product
WHERE laststockdate = (SELECT laststockdate
FROM product WHERE product_name = 'Small Widget');
```

Пример (использование агрегатной функции в однострочном подзапросе):

```
SELECT * FROM product WHERE product_price >
 (SELECT AVG(product_price) FROM product);
```

Многострочные подзапросы

Многострочный подзапрос – это подзапрос, который возвращает лишь >=1 значение.

Для таких подзапросов нельзя выполнять сравнение с помощью знаков равенства/неравенства; необходимо использовать функцию [**NOT**] ***IN***.

```
SELECT * FROM product
WHERE product_name IN
 (SELECT DISTINCT product_name FROM purchase);
```

```
UPDATE product SET product_price = product_price * 0.9
WHERE product_name NOT IN (SELECT DISTINCT product_name
FROM purchase);
```

EXISTS

EXISTS использует подзапрос в качестве аргумента и оценивает его как истинный, если в подзапросе есть выходные данные, а в противном случае как ложный. Выполняется подзапрос один раз и может содержать несколько столбцов, поскольку их значения не проверяются, а просто фиксируется результат наличия строк.

Примечания по предикату EXISTS:

- EXISTS – предикат, возвращающий значение TRUE или FALSE, и его можно применять отдельно или вместе с другими булевыми выражениями.

EXISTS

```
SELECT * FROM product
WHERE EXISTS
 (SELECT * FROM purchase
WHERE product.product_name = purchase.product_name);
```

Групповые условия (операторы сравнения).

ALL - сравнение будет производиться со всеми записями, которые возвращает подзапрос (или просто со всеми значениями в наборе). True вернется только в том случае, если все записи, которые возвращает подзапрос, будут удовлетворять указанному вами условию.

```
SELECT * FROM product
WHERE product_price >= ALL (SELECT product.product_price
FROM purchase, product
WHERE purchase.product_name = product.product_name
AND purchase.salesperson = 'GA');
```

Запрос вернёт все товары из таблицы product, цена которых больше или равна цене каждого товара, проданного сотрудником с кодом 'GA'.

Групповые условия (операторы сравнения).

ANY — сравнение вернет **true**, если условию будет удовлетворять хотя бы одна запись из подзапроса (или набора).

```
SELECT * FROM product WHERE product_price > ANY (SELECT
    product.product_price
FROM purchase, product
WHERE purchase.product_name = product.product_name
AND purchase.salesperson = 'GA');
```

Запрос вернет все записи из таблицы product, для которых цена продукта больше цены какого-либо продукта, проданного сотрудником с кодом 'GA'.

SOME — делает то же самое, что ANY. Полностью взаимозаменяемы.

Практическое задание № 5

1. Напишите запрос, который возвращает всех сотрудников, которых взяли на работу в то же день, что и сотрудника John Smith.

2. Напишите запрос, который возвращает все товары, цена которых ниже средней цены.

3. Напишите запрос, который возвращает все товары, которые продавались более одного раза.

4. Выведите увеличенную на 15% цену товаров, которые продавались более одного раза.

5. Используя условие EXISTS, напишите запрос, который возвращает всех сотрудников, которые хотя бы один раз что-либо продали.

6. Напишите запрос, который возвращает все товары из таблицы product, цена которых меньше цены любого товара, проданного сотрудником с кодом 'GA'.

7. напишите запрос, который вернет все товары из таблицы product, цена которых меньше цены хотя бы одного товара, проданного сотрудником с кодом 'GA'. Убедитесь, что операторы SOME и ANY взаимозаменяемы.

APLANA
QUALITY SERVICES

SQL. Базовый курс

Часть 8. Функции для работы со строками, датами и числами

Функции для работы с числами

ROUND - округляет числа с любой заданной точностью.
ROUND(входное_значение, число_знаков_после_десятичной_точки)

```
SELECT product_name, ROUND(product_price, 0)
FROM product;

SELECT ROUND(1234.5678, 3)        --MSSQL
SELECT ROUND(1234.5678, 3) FROM DUAL;      --Oracle
```

Функция ROUND	Возвращаемое значение
ROUND(1234.5678,4)	1234.5678
ROUND(1234.5678, 3)	1234.568
ROUND(1234.5678, 2)	1234.57
ROUND(1234.5678,1)	1234.6
ROUND(1234.5678,0)	1235
ROUND(1234.5678, -1)	1230
ROUND(1234.5678,-2)	1200
ROUND(1234.5678,-3)	1000

Функции для работы с числами

TRUNC **-** усекает число, понижая его точность.

Функция TRUNC	Возвращаемое значение
TRUNC(1234.5678,4)	1234.5678
TRUNC(1234.5678,3)	1234.567
TRUNC(1234.5678, 2)	1234.56
TRUNC(1234.5678,1)	1234.5
TRUNC(1234.5678,0)	1234
TRUNC(1234.5678,-1)	1230
TRUNC(1234.5678, -2)	1200
TRUNC(1234.5678, -3)	1000

Вспомогательные таблицы

Вспомогательные (dummy) таблицы
Для выполнения функций, без привязки к конкретным таблицам в ряде СУБД необходимо указывать служебную таблицу, поскольку SQL подразумевает конструкцию select … from.

- Oracle – DUAL
- DB2 – SYSDUMMY1
- SYBASE – DUMMY
- MySQL – DUAL
- MSSQL – отсутствует. MSSQL распознает служебные запросы без необходимости указывать dummy-таблицу.

Функции для работы с датами

GETDATE – возвращает текущую дату.

select getdate();

DATEADD – Возвращает дату, полученную как сумму исходной даты date и интервала, добавленного к заданному компоненту datepart даты date.

ADD_MONTHS(величина, количество, начальная дата)
SELECT DATEADD(month, 1, GETDATE());
SELECT DATEADD(year, -2, GETDATE());

Функции для работы с датами

EOMONTH – возвращает последний день любого месяца, указанного в переданной ей дате (MSSQL 2012+).

EOMONTH(дата)

```
SELECT EOMONTH(GETDATE());
SELECT EOMONTH('2015-03-15');
SELECT first_name, last_name, hiredate, EOMONTH(hiredate)+1
FROM person;
```

Функции для работы с датами

DATEDIFF – возвращает количество единиц, разделяющих две даты.

DATEDIFF(величина, начальная дата, конечная дата)

```
SELECT DATEDIFF(millisecond, GETDATE(), SYSDATETIME());
SELECT DATEDIFF(MONTH, '17-AUG-2012', GETDATE());
```

Функции для работы с текстом

UPPER – ставит все символы строки в верхний регистр.

LOWER - ставит все символы строки в нижний регистр.

INITCAP *(oracle)* – изменяет регистр строки на смешанный (первая буква каждого слова будет в верхнем регистре, остальное слово – в нижнем).

```
SELECT UPPER(product_name) FROM product;
SELECT LOWER(product_name) FROM product;
SELECT INITCAP('this TEXT hAd UNpredictABLE caSE') FROM DUAL;
```

Функции для работы с текстом

LEN – определяет длину строки.

```
SELECT product_name, LEN(product_name) LENGTH
FROM product
WHERE LEN(product_name) > 15;
```

Функции для работы с текстом

SUBSTRING – обрезает значение в параметре.

SUBSTRING(исходный_текст, позиция начального символа, количество символов)
SUBSTRING(строка 1, a, [,b])

Возвращает часть «Строка 1», начинающуюся с символа с номером **a**, и имеющую длину **b** символов. Если **a = 0**, это равносильно тому, что **a = 1** (начало строки) если **b** положительно возвращаются символы слева направо. Если **b** отрицательно то, начиная с конца строки и считаются справа налево! Если **b** отсутствует, то по умолчанию возвращаются все символы, до конца строки

Функции для работы с текстом

old_item

item_id	item_desc
MSC-101	Bottle, Small
MSC-102	Bottle, Large
SPB-101	Box, Small
SPB-102	Box, Large

location	item_number	item_desc
MSC	101	Bottle, Small
MSC	102	Bottle, Large
SPB	101	Box, Small
SPB	102	Box, Large

```
SELECT SUBSTRING(item_id, 1, 3) LOCATION,
SUBSTRING(item_id, 5, 3) ITEM_NUMBER
FROM old_item;
```

Функции для работы с текстом

CHARINDEX- находит позицию символа (или символов), разделяющего элементы строк.

CHARINDEX(строка 1, строка 2, [,a])

Возвращает местоположение "строка 1", в "строка 2". "строка 2" просматривается слева, начиная с позиции **a**. Если **a** отрицательно, то "строка 2", просматривается справа. Значением по умолчанию для **a** является 1, что дает в результате позицию, первого вхождения, "строка 1", в "строка 2". Если при заданной **a**, "строка 1" не найдена, возвращается 0

Функции для работы с текстом

CHARINDEX(искомый_символ, текст _для_поиска, позиция _начального_символа)

```
SELECT item_desc, CHARINDEX(',', item_desc, 1 )
FROM old_item;
```

item_desc	column2
Bottle, Small	7
Bottle, Large	7
Box, Small	4
Box, Large	4

Вложение функций

```
SELECT item_desc, SUBSTRING(item_desc, 1, CHARINDEX(',', item_desc, 1))
CATEGORY
FROM old_item;
```

item_desc	CATEGORY
Bottle, Small	Bottle,
Bottle, Large	Bottle,
Box, Small	Box,
Box, Large	Box,

Вложение функций

- SELECT item_desc,
- SUBSTRING(item_desc, 1, CHARINDEX(',', item_desc, 1)-1) CATEGORY,
- SUBSTRING(item_desc, CHARINDEX(',', item_desc, 1)+2, 99) ITEM_SIZE
- FROM old_item;

item_desc	CATEGORY	ITEM_SIZE
Bottle, Small	Bottle	Small
Bottle, Large	Bottle	Large
Box, Small	Box	Small
Box, Large	Box	Large

Практическое задание № 6

1. Используя функции для работы с датами и числами, посчитайте, сколько вам полных лет.
2. Выведите строку 'я ЗнаЮ тЕкСтовыЕ фУнкциИ' в верхнем и нижнем регистре.
3. Узнайте длину этой строки.
4. Работая со столбцом purchase.product_name, выведите:
   - первые три символа
   - все оставшиеся символы, начиная с четвёртого
   - полную строку

Полезные ресурсы

- http://sqlfiddle.com/ - инструмент, эмулирующий пустую БД: позволяет выполнять значительную часть DML, DDR и DR запросов. Поддерживает 5 основных диалектов
- http://www.sql-tutorial.ru/ - интерактивный учебник по SQL на русском
- http://www.sql-ex.ru/ - интерактивный портал для решения задач на SQL
- https://dev.mysql.com/downloads/mysql/ - бесплатный SQL сервер под различные ОС
- https://www.mysql.com/products/workbench/ - бесплатный инструмент для работы с MySql сервером

Спасибо за внимание!

Ваши вопросы?

Компания «Аплана»
Сергей Воробьёв
Ведущий инженер-тестировщик
+7-917-556-13-49
www.aplana.ru

APLANA
QUALITY SERVICES