Управление данными

Лекция 6. *Язык SQL*

Рассматриваемые темы

- Язык SQL. Его назначение.
- Подмножества языка DDL и DML.
  - Операторы DDL: Create, Alter, Drop.
  - Операторы DML: Insert, Update, Delete, Select.
  - Управление правами пользователя Grant, Revoke.
- Реализация специальной логики приложений – триггеры и хранимые процедуры.
- Расширения ANSI SQL.

Язык SQL

- Язык структурированных запросов (Structured queries language);
- Текстовый язык, семантически приближенный к английскому языку;
- Каждый запрос – отдельная команда для СУБД (оператор) с фиксированным синтаксисом.
- Имеет подмножества операторов DDL (определение данных) и DML (манипулирование данными)

DDL

- Операторы создания, изменения и удаления баз данных и объектов схемы данных
- Создание:

```
CREATE <OBJECT> <NAME> [параметры]
```

- Типы объектов:
  - **DATABASE** – база данных;
  - **SCHEMA** – схема данных;
  - **TABLE** – таблица (отношение);
  - **CONSTRAINT** – ограничение;
  - **ATTRIBUTE** – атрибут;
  - **VIEW** – представление;
  - **INDEX** – индекс;
  - **SEQUENCE** – последовательность;
  - **STORED PROCEDURE** – хранимая процедура;
  - **TRIGGER** – триггер;
  - **USER** – пользователь БД.

DDL. Создание таблиц

CREATE TABLE <NAME> (<attributes>
[,<constraints>])

<attribute>: name <datatype> [<constraint>]

<constraint>: CONSTRAINT [name] <c_type>
<parameters>

DDL. Типы данных атрибутов

- Числовые:
  - Счетчик – counter, serial, auto_increment
  - Целое – integer (+ unsigned)
  - Длинное целое – long (+ unsigned)
  - С плавающей запятой – float, double
  - Логический – bit, boolean, smallint
- Строковые
  - Один символ – char
  - Строка n символов – char[n], varchar[n]
- Дата, время – date, time, datetime, timestamp
- Бинарные данные – (long-)(var-)binary

DDL. Ограничения

- Default <val> – принимать значение по умолчанию;
- Not Null – запрет на отсутствие значений
- Unique – запрет повторов
- Primary key – первичный ключ (not null + unique)
- Foreign key references <table> (<PK attribute>) <mode> – внешний ключ (ссылка)
- Check <condition> – требование соблюдать условие

DDL. Пример создания таблицы

```
CREATE TABLE "Поезд"(
    "НомерПоезда" integer                    NOT NULL,
    "Название"    character varying(150)             ,
    "Класс"       integer                    NOT NULL,
    "Режим"       integer                    NOT NULL,

    CONSTRAINT train_pk        PRIMARY KEY ("НомерПоезда"),
    CONSTRAINT train_class_fk  FOREIGN KEY ("Класс")
                               REFERENCES "КлассПоезда" ("КодКласса")
                               ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT train_regim_fk  FOREIGN KEY ("Режим")
                               REFERENCES "РежимКурсирования" ("КодРежима")
                               ON UPDATE CASCADE ON DELETE CASCADE
);
```

(пример для СУБД PostgreSQL)

DDL. Изменение объекта

ALTER <object> <name> [действия по изменению]

Alter table (add column, alter column, drop column) – изменение таблицы

Alter view – изменение представления

Alter database – изменение базы данных

Alter procedure – изменение процедуры

```
ALTER TABLE "Станции" ADD COLUMN "НаселенныйПункт" CHAR(50)

ALTER TABLE "Станции" ADD CONSTRAINT "U2" UNIQUE "НаселенныйПункт"
```

DDL. Удаление объекта

DROP <object> <name>

```
ALTER TABLE "Станции" DROP COLUMN "НаселенныйПункт"

DROP TABLE "Станции"
```

DDL. Порядок создания и удаления объектов схемы:

1. Создается пользователь;
2. Создается база данных;
3. Создается схема;
4. Создаются последовательности;
5. Создаются таблицы (сначала родительские, потом дочерние);
6. Создаются индексы, триггеры и процедуры;
7. Создаются представления.

Удаление – в обратном порядке.

DML

- Операторы манипулирования данными:
  - Извлечение данных – SELECT;
  - Вставка новых данных – INSERT;
  - Изменение данных – UPDATE;
  - Удаление данных – DELETE;
- Объект работы – отношение (таблица) или соединение отношений
- Единица манипулирования – запись

DML. Оператор SELECT

- Оператор предназначен для извлечения из отношения или соединения отношений набора записей, отвечающих заданным условиям.
- Результат работы – новое отношение.
- Реализует все операции реляционной алгебры (объединение, пересечение, проекция, соединение, выборка)

DML. Оператор SELECT

Формат:

SELECT <список атрибутов>
FROM <соединяемые отношения>
[WHERE <условия выборки>]
[ORDER BY <критерии сортировки>]
[GROUP BY <критерии группировки>]
[HAVING <условия отбора групп>]

DML. Оператор SELECT

<список атрибутов>:

Реализует проекцию РА.

- Указываются имена тех атрибутов, извлекаемых из соединенных отношений, которые войдут в результат.
- Для одноименных атрибутов указывается отношение, из которого он извлекается (**table.attribute**)
- Порядок вхождения определяется порядком перечисления.
- Если нужно включить все атрибуты, указывается *.
- Для переименования атрибута в результирующем отношении применяется ключевое слово AS (**attribute as new_name**);
- Вместо атрибута в результат может вставляться результат, возвращаемый функцией (**sin(angle) as “sine of angle”**);
- Для запрета повторений в результате используется директива DISTINCT

DML. Оператор SELECT

<соединяемые отношения>:

Реализует соединение РА.

- Указываются имена соединяемых отношений, декартово произведение которых формирует результат.
- Для удобства записи каждому отношению может быть присвоен псевдоним (**name synonim1, synonim2**)
- Перечисление отношений:
  - Простое перечисление: **FROM table1, table2, …**

    Полное декартово произведение;
  - Соединение по условию: **FROM table1 [INNER | LEFT | RIGHT | FULL] JOIN table2 on <conditon>**

    Декартово произведение, содержащее только строки, отвечающие условию

Примеры соединения:

table1

num	name
1	a
2	b
3	c

table2

num	value
1	xxx
3	yyy
5	zzz

1) SELECT * FROM table1, table2

result

table1.num	name	table2.num	value
1	a	1	xxx
1	a	3	yyy
1	a	5	zzz
2	b	1	xxx
2	b	3	yyy
2	b	5	zzz
3	c	1	xxx
3	c	3	yyy
3	c	5	zzz

2) SELECT * FROM table1 a INNER JOIN table2 b ON a.num=b.num

result

a.num	name	b.num	value
1	a	1	xxx
3	c	3	yyy

Примеры соединения:

table1

num	name
1	a
2	b
3	c

table2

num	value
1	xxx
3	yyy
5	zzz

3) SELECT * FROM table1 a **XXX** JOIN table2 b ON a.num=b.num

result

a.num	name	b.num	value
1	a	1	xxx
2	b		
3	c	3	yyy

XXX = LEFT

result

a.num	name	b.num	value
1	a	1	xxx
3	c	3	yyy
		5	zzz

XXX = RIGHT

result

a.num	name	b.num	value
1	a	1	xxx
2	b		
3	c	3	yyy
		5	zzz

XXX = FULL

DML. Оператор SELECT

<условия выборки>:

Реализует выборку РА.

- Указывается одно логическое выражение, которому должны удовлетворять все записи соединенного отношения.
- Записи, не удовлетворяющие условию, отбрасываются.
- Допускаются логические связки AND, OR, NOT.
- Основные условия:
  - Для чисел и дат:

    <, >, >=, <= , BETWEEN min AND max;
  - Для строк:

    like ‘pattern’ – сравнение с образцом (_, %);
  - Для всех типов:

    =, <> IS NULL, IS NOT NULL - для всех типов (нельзя = NULL!);

    IN (set or subquery), EXISTS (subquery)

Примеры условий

- SELECT Фамилия, Курс FROM СТУДЕНТ WHERE Специальность=’ Математика’ AND Курс=5
- SELECT Фамилия FROM СТУДЕНТ WHERE Специальность IN (’ Математика’, ’Экономика’)
- SELECT Фамилия, Специальность FROM СТУДЕНТ WHERE НомерСтудента BETWEEN 200 AND 300
- SELECT Фамилия, Специальность FROM СТУДЕНТ WHERE НомерСтудента >= 200 AND НомерСтудента <= 300
- SELECT Фамилия AS ‘ФИО’ FROM СТУДЕНТ WHERE Фамилия LIKE ‘Р%’
- SELECT Фамилия FROM СТУДЕНТ WHERE Курс IS NULL

Примеры условий на соединение

- SELECT Фамилия FROM СТУДЕНТ WHERE НомерСтудента IN (SELECT НомерСтудента FROM ЗАПИСЬ WHERE Предмет = ‘А’)
- SELECT Фамилия FROM СТУДЕНТ WHERE СТУДЕНТ.НомерСтудента IN (SELECT ЗАПИСЬ.НомерСтудента FROM ЗАПИСЬ WHERE ЗАПИСЬ.Предмет IN (SELECT ЗАНЯТИЯ. Предмет FROM ЗАНЯТИЯ WHERE ЗАНЯТИЯ.ДеньНедели = 2))
- SELECT СТУДЕНТ.НомерСтудента, СТУДЕНТ.Фамилия, ЗАПИСЬ. Предмет FROM СТУДЕНТ, ЗАПИСЬ WHERE СТУДЕНТ.НомерСтудента = ЗАПИСЬ.НомерСтудента
- SELECT НомерСтудента, Предмет, ДеньНедели FROM СТУДЕНТ, ЗАПИСЬ, ЗАНЯТИЯ WHERE СТУДЕНТ.НомерСтудента = ЗАПИСЬ. НомерСтудента AND ЗАПИСЬ.Предмет = ЗАНЯТИЯ. Предмет AND СТУДЕНТ.Фамилия = ‘Сидоров’

DML. Оператор SELECT

<критерии сортировки>:

- Указываются имена атрибутов, по значениям которых требуется упорядочить записи в результате.
- При указании более одного атрибута – лексикографическая сортировка.
- Для каждого атрибута может быть задано свое направление сортировки:
  - ASC – по возрастанию
  - DESC – по убыванию
- Вместо имени атрибута может быть указан его порядковый номер в результате
- Если направление сортировки не указано, сортировка производится по возрастанию

Примеры сортировки

- SELECT Фамилия, Специальность, Курс FROM СТУДЕНТ WHERE Специальность=’Экономика’ ORDER BY Фамилия
- SELECT Фамилия, Специальность, Курс FROM СТУДЕНТ WHERE Курс IN (1, 2, 4) ORDER BY Фамилия ASC, 3 DESC

DML. Оператор SELECT
Агрегирующие функции

- При необходимости запрос может вернуть не сами записи, а их агрегированные величины:
  - **COUNT** – количество значений поля
  - **SUM** – сумма значений поля
  - **MIN** – минимальное значение поля
  - **MAX** – максимальное значение поля
  - **AVG** – среднее (арифметическое) значение поля
  - **STDDEV** – стандартное отклонение поля
- В этом случае всегда возвращается ОДНА запись.
- НЕЛЬЗЯ в один запрос вставлять поле и агрегированную величину:

```
SELECT COUNT([DISTINCT] ФАМИЛИЯ), SUM(СТИПЕНДИЯ) FROM СТУДЕНТЫ

SELECT ФАМИЛИЯ, COUNT(ИМЯ) FROM СТУДЕНТЫ
```

DML. Оператор SELECT

<критерии группировки>:

- Указываются имена атрибутов, одинаковые значения которых образуют одинаковую групповую запись в результате.
- При использовании группировки для каждой группы можно вычислить агрегированное значение (SUM, COUNT etc.).
- Группировать можно только по тем атрибутам, которые указаны после SELECT, а не по любым атрибутам соединяемых отношений.

DML. Оператор SELECT

<условия отбора групп>:

- Указываются требования, которым должны удовлетворять сформированные группы, чтобы быть отобранными в результат.
- Если группа не удовлетворяет условию, она вся отбрасывается.
- При использовании в одном запросе секций WHERE и HAVING сначала выполняется WHERE (отбор записей), потом GROUP BY (группировка), а потом – HAVING (отбраковка групп).

Примеры группировки

- SELECT Специальность, COUNT(*) FROM СТУДЕНТ
  GROUP BY Специальность
- SELECT Специальность, COUNT(*) FROM СТУДЕНТ
  GROUP BY Специальность HAVING COUNT(*) > 2
- SELECT Специальность, MAX(НомерСтудента) FROM СТУДЕНТ
  WHERE Курс = 4
  GROUP BY Специальность HAVING COUNT(*) > 1

Реализация теоретико-множественных операций

- Объединение R1 ∪ R2:

  (SELECT * FROM R1) **UNION** (SELECT * FROM R2)

- Пересечение R1 ∩ R2:

  SELECT * FROM R1 WHERE IN (SELECT * FROM R2)

- Разность R1\R2:

  SELECT * FROM R1 WHERE NOT IN (SELECT * FROM R2)

- Симметрическая разность R1 Δ R2:

  (SELECT * FROM R1 WHERE NOT IN (SELECT * FROM R2))
  UNION (SELECT * FROM R2 WHERE NOT IN (SELECT * FROM R1))

- Декартово произведение R1 × R2:

  SELECT * FROM R1, R2

DML. Оператор UPDATE

- Оператор предназначен для изменения в отношении (или соединении отношений, если это допускает БД) набора записей, отвечающих заданным условиям.
- Изменяются указанные в запросе поля записей.
- Результат работы – отношение с измененными записями. Сам оператор никакого результата не возвращает (нельзя использовать в SELECT).
- При обновлении могут срабатывать триггеры, а так же выполняться обновления значений внешних ключей в ссылающихся таблицах.
- Запрос может быть не исполнен, если новое значение будет нарушать ограничения.

DML. Оператор UPDATE

```
UPDATE <Name>
SET <field=val [, field=val, …]>
[WHERE <Condition>]
```

Name – имя обновляемой таблицы, или их соединение (join)*

Field – имя обновляемого поля

Val – присваиваемое полю значение (может быть выражение, в том числе, использующее СТАРОЕ значение поля)

Condition – условие, которому должна удовлетворять обновляемая запись

Примеры обновления

- UPDATE Факультеты f INNER JOIN Кафедры k ON f.ID = k.FacID SET f.Бюджет = 0 WHERE k.Выпускающая=true

- UPDATE persons SET street = 'Nissestien 67', city = 'Sandnes' WHERE lastname = 'Tjessem' AND firstname = 'Jakob‘

- UPDATE emp a
  SET deptno = (SELECT deptno FROM dept WHERE loc = ‘BOSTON’),
  (sal, comm) = (SELECT 1.1*AVG(sal), 1.5*AVG(comm) FROM emp b WHERE a.deptno = b.deptno)
  WHERE deptno IN (SELECT deptno FROM dept WHERE loc = ‘DALLAS’ OR loc = ‘DETROIT’)

- UPDATE sales SET SaleDate=Null, Count=0

DML. Оператор DELETE

- Оператор предназначен для удаления в отношении (или соединении отношений, если это допускает БД) набора записей, отвечающих заданным условиям.
- Записи удаляются целиком (нельзя удалить часть записи).
- Результат работы – отношение с удаленными записями. Сам оператор никакого результата не возвращает (нельзя использовать в SELECT)
- При удалении могут срабатывать триггеры, а так же выполняться
  - «об-null-ивание» значений внешних ключей в ссылающихся таблицах;
  - Удаление ссылающихся записей в ссылающихся таблицах

DML. Оператор DELETE

```
DELETE FROM <Name>
[WHERE <Condition>]
```

Name – имя таблицы или соединение (join)*

Condition – условие, которому должны удовлетворять удаляемые записи

Примеры удаления

- **DELETE FROM products WHERE price = 10;**
- **DELETE FROM products;**
- **DELETE FROM Authors a JOIN Articles b ON a.ID=b.Author WHERE AuthorLastName='Henry';**

DML. Оператор INSERT

- Оператор предназначен для вставки в отношение одной или более записей.
- Записи вставляются целиком (нельзя вставить часть записи).
- Результат работы – отношение с добавленными записями. Сам оператор никакого результата не возвращает (нельзя использовать в SELECT)
- При вставке записей могут срабатывать триггеры
- Запись(и) может быть не вставлена, если нарушается условие на ее значения.
- В качестве источника записей может быть использован оператор SELECT
- Неуказанные в запросе поля принимают значение DEFAULT

DML. Оператор INSERT

```
INSERT INTO <Name> ([<Col>, ... ]) VALUES (<val>,...)

INSERT INTO <Name> VALUES (<val>,...)

INSERT INTO <Name> SELECT <cols> FROM <tables>
    WHERE <Condition>
```

Name – имя таблицы
Col – имя столбца (поля)
Val – значение поля во вставляемой записи
SELECT – запрос, извлекающий набор записей, использующихся для вставки

Примеры вставки

- **INSERT INTO films VALUES ('UA502', 'Bananas', 105, '1971-07-13', 'Comedy', '82 minutes')**
- **INSERT INTO films (code, title, did, date_prod, kind) VALUES ('T_601', 'Yojimbo', 106, '1961-06-16', 'Drama')**
- **INSERT INTO films VALUES ('UA502', 'Bananas', 105, DEFAULT, 'Comedy', '82 minutes')**
- **INSERT INTO films (code, title, did, date_prod, kind) VALUES ('B6717', 'Tampopo', 110, '1985-02-10', 'Comedy'), ('HG120', 'The Dinner Game', 140, DEFAULT, 'Comedy');**
- **INSERT INTO films SELECT * FROM tmp_films WHERE date_prod < '2004-05-07';**

Операторы управления пользователем БД

- CREATE USER <username>
- ALTER DATABASE <name> SET OWNER=<username>
- GRANT <privilege> ON <name> TO <username>
- REVOKE <privilege> ON <name> FROM <username>
- DROP USER <username>

Привилегии пользователя

- SELECT
- INSERT
- UPDATE
- DELETE
- REFERENCES
- TRIGGER
- ALL PRIVILEGES

```
GRANT SELECT ON mytable TO PUBLIC;
GRANT SELECT, UPDATE, INSERT ON mytable TO admin;
GRANT SELECT (col1), UPDATE (col1) ON mytable TO miriam_rw;

REVOKE INSERT ON films FROM PUBLIC;
REVOKE ALL PRIVILEGES ON kinds FROM manuel;
```

Управление транзакциями

- Транзакция – последовательность логически связанных запросов, целенаправленно и логически связанно меняющих состояние БД;
- У транзакции имеется начало, набор точек сохранения (отката) и конец.
- В конце транзакцию можно применить (зафиксировать) или откатить
- В процессе исполнения транзакции, до ее завершения (фиксации или отката) объекты, которыми на манипулирует, могут быть «захвачены»

Операторы управления транзакциями:

- BEGIN – применяется для того, чтобы:
  - Зафиксировать, что транзакция началась
  - Указать (при необходимости), какие объекты захватываются и уровень их блокировки
- SAVEPOINT <NAME>
  - Указывает точку возврата, к которой можно откатиться при частичном откате транзакции
- RELEASE SAVEPOINT <NAME>
  - Удаление успешно пройденной точки возврата

Операторы управления транзакциями:

- COMMIT – применяется для того, чтобы:
  - сделать «постоянными» все изменения, сделанные в текущей транзакции (реально данные могут быть изменены несколько позже)
  - очистить все точки сохранения данной транзакции
  - завершить транзакцию
  - освободить все блокировки данной транзакции

Операторы управления транзакциями:

- ROLLBACK – применяется для того, чтобы:
  - отменить все изменения, внесённые начиная с момента начала транзакции или с какой-то точки сохранения (SAVEPOINT).
  - очистить все точки сохранения данной транзакции
  - завершить транзакцию
  - освободить все блокировки данной транзакции

Примеры:

```
BEGIN;
  INSERT INTO table1 VALUES (1);
  SAVEPOINT my_savepoint;
  INSERT INTO table1 VALUES (2);
  ROLLBACK TO SAVEPOINT my_savepoint;
  INSERT INTO table1 VALUES (3);
COMMIT;
```

```
BEGIN;
  INSERT INTO table1 VALUES (3);
  SAVEPOINT my_savepoint;
  INSERT INTO table1 VALUES (4);
  RELEASE SAVEPOINT my_savepoint;
COMMIT;
```

Хранимые процедуры и триггеры

- Используются для реализации сложной бизнес-логики (положений делового регламента, не описываемых ограничениями);
- Хранятся на сервере СУБД
- Пишутся на расширенном языке SQL, содержащем специальные операторы:
  - Передача управления (CALL, GO TO, RETURN)
  - Проверка условий (IF … ELSE, SWITCH)
  - Организация циклов (FOR, WHILE)

Хранимые процедуры

- Вызываются приложением, как запрос с использованием ключевого слова EXECUTE или CALL;
- Могут иметь аргументы и возвращать результат (в том числе – отношение, как и SELECT);
- Аргументы могут быть входящие и исходящие;
- Могут выполнить любое количество запросов на SQL, в том числе – несколько транзакций;
- Результаты внутренних запросов SELECT обрабатываются в виде курсоров

Курсор

- Курсор – временная структура данных (аналог таблицы), хранящий результаты запроса SELECT построчно
- Предназначен для обработки в процедурах
- Имеет операции
  - OPEN – открыть курсор
  - FETCH – перейти к очередной записи
  - CLOSE – закрыть курсор
- Бывают однонаправленные и реверсивные курсоры

Пример хранимой процедуры

```
CREATE OR REPLACE PROCEDURE cs_create_job(v_job_id IN INTEGER) IS
  a_running_job_count INTEGER;
  PRAGMA AUTONOMOUS_TRANSACTION;

 BEGIN
  LOCK TABLE cs_jobs IN EXCLUSIVE MODE;
  SELECT count(*) INTO a_running_job_count FROM cs_jobs WHERE end_stamp IS NULL;
  IF a_running_job_count > 0 THEN COMMIT;
   raise_application_error(-20000, 'Unable to create a new job: a job is currently running.');
  END IF;
  DELETE FROM cs_active_job;
  INSERT INTO cs_active_job(job_id) VALUES (v_job_id);
  BEGIN INSERT INTO cs_jobs (job_id, start_stamp) VALUES (v_job_id, sysdate);
    EXCEPTION WHEN dup_val_on_index THEN NULL;
  END;
  COMMIT;
END;
```

Триггеры

- Хранимые процедуры, привязываемые к таблицам, и вызываемые при ее изменении:
  - Вставка, удаление и/или изменение записей
- Триггеры бывают:
  - Табличные – вызывается при изменении для всей таблицы 1 раз при изменении;
  - Строчные – вызывается при изменении для каждой записи;
- У одной таблицы может быть несколько триггеров, одна и та же процедура может выполнять роль разных триггеров.
- Триггеры могут быть «до» и «после»-триггеры.
- Триггер имеет доступ как к старым (до изменения) так и новым (после изменения) данным.

Пример триггера

```
CREATE OR REPLACE TRIGGER DistrictUpdatedTrigger
 AFTER UPDATE ON district
 BEGIN
  INSERT INTO info VALUES ('table "district" has changed');
 END;


 /* Триггер на уровне строки */
 CREATE OR REPLACE TRIGGER DistrictUpdatedTrigger
 AFTER UPDATE ON district FOR EACH ROW
 BEGIN
  INSERT INTO info VALUES ('one string in table "district" has
     changed');
 END;
```