

# Содержание

<b>I</b>	<b>Реляционная модель данных</b>	<b>2</b>
1	Основные функции СУБД	2
2	Модель данных сущность—связь: сущности, атрибуты и множества сущностей	2
3	Модель данных сущность—связь: связи. Использование сущностей и связей при проектировании БД	3
4	Реляционная модель данных: отношения, таблицы, домены, атрибуты. Описание таблиц в языке SQL	4
5	Ограничения целостности: уникальность атрибута, нулевые значения, значения по-умолчанию	4
6	Ограничения целостности: первичный ключ	5
7	Ограничения целостности: внешний ключ	5
8	Ограничения целостности: ограничение на значения атрибута	5
9	Ограничения целостности: вычисляемые атрибуты	5
10	Реляционная модель данных: алгебраические операции	5
11	Реляционная модель данных: реляционные операции	6
12	Отображение модели сущность—связь в реляционную. Представление объектов	8
13	Отображение модели сущность—связь в реляционную. Представление связей	8
14	Нормализация: декомпозиция отношений. Нормальные формы	8
15	Нормализация: многозначные зависимости	9
<b>II</b>	<b>Язык SQL</b>	<b>9</b>
16	Язык запросов SQL: операции реляционной алгебры	9
17	Язык запросов SQL: вложенные подзапросы	10
18	Язык запросов SQL: агрегирование и упорядочение	10
19	Типы данных SQL	11
20	Временные таблицы и табличные переменные	11
21	Представления (обычные, с фиксацией схемы, индексируемые)	11
22	Обобщённые табличные выражения	12
23	Хранимые процедуры в базе данных	12
24	Курсоры в базах данных	12
25	Скалярные функции, определяемые пользователем	13
26	Табличные функции, определяемые пользователем	14
27	Активные базы данных: триггеры	14

<b>III Индексы</b>	<b>14</b>
28 Кластеризованные таблицы и индексы	14
29 Первичные и вторичные индексы. Плотные и неплотные индексы	15
30 Индексы: B-дерево	15
31 Индексы: отфильтрованные, покрывающие, составной ключ	15
32 Индексы: полнотекстовые	15
33 Индексы: хеширование	15
34 Индексы: битовые шкалы	15
35 Индексы: R-дерево	15

## Часть I

# Реляционная модель данных

## 1. Основные функции СУБД

**Определение 1.** *База данных* — это упорядоченный набор структурированной информации или данных которые обычно хранятся в электронном виде в компьютерной системе.

База данных обычно управляется *системой управления базами данных (СУБД)*.

Функции СУБД:

- средства постоянного хранения данных;
- поддержка безопасности данных и защита от несанкционированного доступа;
- обеспечение согласованности данных;
- поддержка высокоуровневых эффективных языков запросов/

## 2. Модель данных сущность—связь: сущности, атрибуты и множества сущностей

Основные элементы диаграммы “сущность—связь”:

- сущность (прямоугольник);
- атрибут (овал);
- связь (ромб).

**Определение 2.** *Сущность* — это абстрактный объект определённого вида. Любой предмет или понятие, информацию о котором мы будем хранить/

У каждой должно быть уникальное имя. К одному и тому же имени всегда должна применяться одна и та же интерпретация.

**Определение 3.** Набор экземпляров сущностей образует *множество*.

Отдельные характеристики объекта называются *атрибутами*.

Требования к атрибутам:

- каждый атрибут имеет *имя* и *тип данных*;
- сущность может обладать любым количеством атрибутов;

- значение атрибута атомарно;
- сущность и её атрибуты на диаграмме соединяются ненаправленными дугами;
- значения атрибутов выбираются из соответствующего множества значений.

**Определение 4.** *Атрибут* — именованная характеристика сущности. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей. Атрибуты используются определения того, какая информация должна быть собрана о сущности.

**Определение 5.** *Ключ* — это один или несколько атрибутов объекта, по которому объект можно однозначно идентифицировать.

### 3. Модель данных сущность—связь: связи. Использование сущностей и связей при проектировании БД

**Определение 6.** *Связь* — это отношение между объектами.

В любой связи выделяются два конца, на каждом из которых указывается:

- имя конца связи;
- степень конца связи (сколько экземпляров данной сущности связывается).

#### Свойства.

1. Связи могут иметь собственные атрибуты.
2. Подобные связи объединяются в множества.
3. Связи не могут существовать без связываемых сущностей.
4. Ключ связи включает ключи связываемых сущностей и, возможно, выделенные атрибуты связи.

Характеристики связи:

- Размерность:
  - бинарные;
  - тернарные;
  - $n$ -арные;
  - рекурсивные.
- Мощность:
  - 1:1;
  - 1: $N$ ;
  - $M$ : $N$ .
- Модальность:
  - обязательные (“должен”) — экземпляр одной сущности **обязан** быть связан не менее чем с одним экземпляром другой сущности;
  - необязательные (условные, “может”) — экземпляр одной сущности **может быть** связан с одним или несколькими экземплярами другой сущности, а может быть и не связан ни с одним экземпляром.

*Есть только такое:*

Шаги при создании ERD:

1. определить сущности;
2. определить атрибуты сущностей;

3. определить первичные ключи;
4. определить отношения между сущностями;
5. определить мощность связей;
6. нарисовать ERD;
7. проверить ERD.

## 4. Реляционная модель данных: отношения, таблицы, домены, атрибуты. Описание таблиц в языке SQL

**Определение 7.** *Реляционная модель данных* — логическая модель данных, описывающая:

- структуры данных в виде наборов отношений;
- теоретико-множественные операции над данными: объединение, пересечение, разность и декартово произведение;
- специальные реляционные операции: селекция, проекция, соединение и деление;
- специальные правила, обеспечивающие целостность данных.

Структуры данных:

- *домены* — множества, элементы которых рассматриваются как скалярные значения;
- *отношения* — предикаты, заданные на прямом произведении доменов;
- *атрибуты* — заголовок отношения; количество атрибутов — *размерность* отношения.

Атрибуты:  $A_1, A_2, \dots, A_n$ .

Домены:  $D_1, D_2, \dots, D_n$ .

Кортежи:  $t = \langle a_1, a_2, \dots, a_n \rangle$ ,  $a_i \in D_i$ ; Отношения:  $R \subset D_1 \times D_2 \times \dots \times D_n$ .

**Замечание.** Кортежи отличаются друг от друга только значением своих атрибутов. В реляционной БД не может быть двух одинаковых кортежей в одной таблице.

**Определение 8.** *Таблицы* — это единицы хранения данных в базе. Таблицы хранят все данные, к которым может обращаться пользователь. Реляционную БД можно рассматривать как коллекцию простых таблиц, связанных между собой.

Таблицы в БД создаются с помощью оператора CREATE:

```
CREATE TABLE Table_Name(
    {имя столбца} {тип данных} [значение по-умолчанию] [правила целостности]}+)
```

## 5. Ограничения целостности: уникальность атрибута, нулевые значения, значения по-умолчанию

**Определение 9.** *Правила целостности* — это правила (ограничения), которым должны соответствовать данные. Их используют для того, чтобы обеспечить:

- связи между сущностями (*ссылочная целостность*);
- ограничения по значениям атрибутов в сущностях (*сущностная целостность*).

Уникальность значения:

```
CREATE TABLE people (id INT UNIQUE NULL)
```

Несколько полей (или их комбинаций) могут быть уникальны. Уникальное значение может быть NULL.

## 6. Ограничения целостности: первичный ключ

```
CREATE TABLE people (id INT PRIMARY KEY)
```

PRIMARY KEY всегда NOT NULL. PRIMARY KEY только один в таблице.

Первичный ключ может состоять из нескольких атрибутов:

```
CREATE TABLE people (name CHAR(20), address VARCHAR(35), PRIMARY KEY(name, address))
```

## 7. Ограничения целостности: внешний ключ

```
CREATE TABLE phone (id_p INT REFERENCES people (id))
```

или

```
CREATE TABLE phone (id_p INT, FOREIGN KEY (id_p) REFERENCES people (id))
```

Требования к внешнему ключу:

- поле, на которое ссылается внешний ключ, должно быть PRIMARY KEY или UNIQUE;
- оба поля должны быть строго одного типа.

## 8. Ограничения целостности: ограничение на значения атрибута

```
CREATE TABLE people (id INT PRIMARY KEY, gender CHAR CHECK (gender IN ('F', 'M')))
```

```
CREATE TABLE people (id INT PRIMARY KEY, birthday DATE, beg_date DATE,  
CHECK (birthday < beg_date))
```

```
CREATE TABLE people (id INT PRIMARY KEY, gender CHAR,  
CONSTRAINT chk_Person CHECK (gender IN ('F', 'M')))
```

## 9. Ограничения целостности: вычисляемые атрибуты

```
CREATE TABLE people (id INT PRIMARY KEY, salary INT, tax AS salary * 0.13)
```

Расширение PostgreSQL:

```
CREATE TABLE people (id INT PRIMARY KEY, salary INT,  
tax INT GENERATED ALWAYS AS (salary * 0.13) STORED)
```

## 10. Реляционная модель данных: алгебраические операции

**Определение 10.** *Схема отношения* — конечное множество упорядоченных пар вида  $\langle A, T \rangle$ , где  $A$  — имя атрибута, а  $T$  — имя некоторого базового типа или ранее определённого домена.

**Определение 11.** Количество атрибутов называется *арностью* (размерностью) отношения.

Все операции *реляционной алгебры* производятся над отношением, и результатом операции является отношение.

Реляционная алгебра является *замкнутой*: в качестве аргументов в реляционные операторы можно подставлять другие реляционные операторы, подходящие по типу. В реляционных выражениях можно использовать вложенные выражения сколь угодно вложенной структуры.

Каждое отношение обязано иметь уникальное имя в пределах БД. Если отношения подставляются в качестве аргументов в другие реляционные выражения, то они могут быть неименованными.

Операции реляционной алгебры делятся на два типа:

- теоретико-множественные операции;
- специальные реляционные операции.

Теоретико-множественные операции:

- объединение отношений;

- пересечение отношений;
- разность отношений;
- декартово произведение отношений.

Два отношения *совместимы по взятию декартова произведения* в том и только в том случае, если пересечение имён атрибутов, взятых из схем отношений, пусто. Любые два отношения могут стать совместимыми по взятию декартова произведения, если применить к одному из них операцию переименования.

Свойства операций:

- ассоциативность (кроме разности);
- коммутативность (кроме разности).

## 11. Реляционная модель данных: реляционные операции

Специальные реляционные операции:

- *ограничение отношения (селекция)* — горизонтальная вырезка;
- *проекция отношения* — вертикальная вырезка;
- *соединение отношений* (по условию, эквисоединение, естественное соединение);
- *деление отношений*.

### Селекция

R **WHERE** f

$$\sigma_f(R)$$

Условие ограничения имеет вид:

- (*a операция\_сравнения b*), где *a* и *b* — имена атрибутов ограничиваемого отношения; атрибуты *a* и *b* определены на одном домене, для значений которого поддерживается операция сравнения;
- (*a операция\_сравнения const*), где *a* — имя атрибута ограничиваемого отношения; атрибут *a* должен быть определён на домене или базовом типе, для значений которого поддерживается операция сравнения.

Условие может состоять из нескольких простых логических выражений, связанных булевскими операторами AND, NOT, OR.

Результатом селекции является отношение, заголовок которого совпадает с заголовком отношения-операнда, а в тело входят те кортежи отношения-операнда, для которых условие ограничения выполнено.

### Проекция

**Определение 12.** *Проекцией* отношения *R* по атрибутам  $X, Y, \dots, Z$  где каждый из атрибутов принадлежит отношению, называется отношение с заголовком  $(X, Y, \dots, Z)$  и телом, содержащим множество кортежей вида  $(x, y, \dots, z)$  таких, для которых найдутся кортежи со значением атрибута *X* равным *x*, ..., значением атрибута *Z* равным *z*.

$$\pi_{(X,Y,\dots,Z)}(R) = \{ x, y, \dots, z : \exists a_1, a_2, \dots, a_n \in R \ \& \ x = a_{i1}, y = a_{i2}, \dots, z = a_{im} \}$$

### Соединение по условию ( $\theta$ -соединение)

Три операнда: соединяемые отношения и условие. Операнды должны быть совместимы по взятию декартова произведения.

A **JOIN** B **WHERE** f = (A \$ \times \$ B) **WHERE** f

$$R \bowtie_f S = \sigma_f(R \times S)$$

## Эквисоединение

**Определение 13.** Операция соединения называется операцией *эквисоединения* (EQUI JOIN), если условие соединения имеет вид  $(a = b)$ .

Если соединение происходит по атрибутам с одинаковыми именами, то в результирующем отношении появляется два атрибута с одинаковыми значениями. В таком случае нужно брать проекцию по всем атрибутам, кроме одного из дублирующих.

## Естественное соединение

Операция *естественного соединения* применяется к паре отношений  $R(A, X)$  и  $S(X, B)$ , обладающих общим атрибутом  $X$ .

$$T(A, X, B), \quad A \triangleright \triangleleft B$$

В синтаксисе естественного соединения не указывается, по каким атрибутам производится соединение. Естественное соединение производится **по всем** одинаковым атрибутам.

$$R \triangleright \triangleleft S = \pi_{\text{атрибуты } R, S \setminus S.A} \sigma_{R.A=S.A}(R \times S)$$

## Деление

**Определение 14.** Результатом *деления*  $A$  на  $B$  является “унарное” отношение  $C(a)$ , тело которого состоит из кортежей  $v$  таких, что в теле отношения  $A$  содержатся кортежи  $\langle v, w \rangle$  для любого  $w$  из  $B$ .

$$(A \text{ DIVIDE BY } B) = C : \quad C \times B \subset A$$

$$R \text{ DIVIDE BY } S = \pi_{1,2,\dots,r-s}(R) - \pi_{1,2,\dots,r-s}(\pi_{1,2,\dots,r-s}(R \times S) \setminus R)$$

### Свойства.

#### 1. Коммутативность

- для декартова произведения:

$$R_1 \times R_2 = R_2 \times R_1$$

- для соединений:

$$R_1 \triangleright \triangleleft_f R_2 = R_2 \triangleright \triangleleft_f R_1$$

#### 2. Ассоциативность:

- для декартова произведения:

$$(R_1 \times R_2) \times R_3 = R_1 \times (R_2 \times R_3)$$

- для соединений:

$$(R_1 \triangleright \triangleleft_{f_1} R_2) \triangleright \triangleleft_{f_2} R_3 = R_1 \triangleright \triangleleft_{f_1} (R_2 \triangleright \triangleleft_{f_2} R_3)$$

#### 3. Комбинация (каскад):

- для селекций:

$$\sigma_{f_1}(\sigma_{f_2}(R)) = \sigma_{f_1 \vee f_2}(R)$$

- для проекций:

$$\pi_{A_1, A_2, \dots, A_m}(\pi_{B_1, B_2, \dots, B_n}(R)) = \pi_{A_1, A_2, \dots, A_m}(R), \quad \text{где } \{A_m\} \subset \{B_n\}$$

#### 4. Перестановка

- селекции и проекции:

$$\sigma_f \pi_{A_1, A_2, \dots, A_m}(R) = \pi_{A_1, A_2, \dots, A_m} \sigma_f(R)$$

- селекции и объединения:

$$\sigma_f(R_1 \cup R_2) = \sigma_f(R_1) \cup \sigma_f(R_2)$$

- селекции и декартова произведения:

$$\sigma_f(R_1 \times R_2) = (\sigma_{f_1}(R_1)) \times (\sigma_{f_2}(R_2))$$

- селекции и разности:

$$\sigma_f(R_1 \setminus R_2) = \sigma_f(R_1) \setminus \sigma_f(R_2)$$

- селекции и пересечения:

$$\sigma_f(R_1 \cap R_2) = \sigma_f(R_1) \cap \sigma_f(R_2)$$

- проекции и декартова произведения:

$$\pi_{A_1, A_2, \dots, A_m}(R_1 \times R_2) = (\pi_{B_1, B_2, \dots, B_n}(R_1)) \times (\pi_{C_1, C_2, \dots, C_r}(R_2))$$

## 12. Отображение модели сущность—связь в реляционную. Представление объектов

Каждая сущность превращается в таблицу. Имя сущности становится именем таблицы.

Каждый атрибут становится столбцом. Столбцы для необязательных атрибутов могут содержать неопределённые значения, столбцы для обязательных — не могут.

Компоненты уникального идентификатора сущности превращаются в ключ таблицы.

## 13. Отображение модели сущность—связь в реляционную. Представление связей

Связи также хранятся в отношении. Схема данного отношения составляется из ключевых атрибутов, участвующих в связи.

## 14. Нормализация: декомпозиция отношений. Нормальные формы

**Определение 15 (функциональная зависимость).**  $R\{A_1, A_2, \dots, A_n\}$ ,  $X, Y \subset \{A_1, A_2, \dots, A_n\}$   
 $X \rightarrow Y$ , если любому значению  $X$  соответствует ровно одно значение  $Y$ .  
 $X$  называется *детерминантом*,  $Y$  — *зависимой частью*.

$$X \rightarrow Y \iff |\pi_Y(\sigma_{X=x}(R))| \leq 1$$

**Определение 16.** *Ключ* — минимальный набор атрибутов, который функционально определяет все остальные.

**Определение 17.**  $Y$  *полностью функционально зависит* от  $X$ , если  $Y$  функционально зависит от всех атрибутов, входящих в состав  $X$ , а не от какой-то его части.

**Определение 18.** Функциональная зависимость  $A \rightarrow C$  называется *транзитивной*, если существует такой атрибут  $B$ , что  $A \rightarrow B$  и  $B \rightarrow C$  и отсутствует функциональная зависимость  $C \rightarrow A$ .

**Определение 19.** *Декомпозиция* — это разбиение на множества, может быть пересекающиеся, такие, что их объединение — это исходное отношение.

Восстановить исходное отношение можно только естественным соединением.

Нормальные формы:

- Первая:

Значение каждого атрибута в таблице должно быть атомарно.



- Вторая:  
Таблица находится в 2НФ, если она находится в 1НФ, и каждый атрибут полностью зависит от любого его ключа, но не от подмножества ключа.
- Третья:  
Отношение находится в 2НФ, и любой атрибут, не являющийся первичным, нетранзитивно зависит от любого возможного ключа.
- Бойса—Кодда:  
Если  $X \rightarrow A$ ,  $A \notin X$ , то  $X \supseteq$  ключ  $R$ .

## 15. Нормализация: многозначные зависимости

**Определение 20.** Пусть  $A$  и  $B$  — два атрибута отношения  $R$ .

Между этими атрибутами существует *многозначная* зависимость, если значению  $a$  атрибута  $A$  соответствует множество значений  $\{b_1, b_2, \dots, b_k\}$  атрибута  $B$ .

**Обозначение.**  $A \twoheadrightarrow B$

**Лемма 1 (Фейджина).** В отношении  $R\{A, B, C\}$  выполняется MVD  $A \twoheadrightarrow B$  в том и только в том случае, когда выполняется MVD  $A \twoheadrightarrow C$ .

**Теорема 1 (Фейджина).**  $R\{A, B, C\}$

Переменная-отношение  $R$  будет равна соединению её проекций  $[A, B]$  и  $[A, C]$  тогда и только тогда, когда для переменной-отношения  $R$  выполняется многозначная зависимость  $A \twoheadrightarrow B|C$ .

Четвёртая нормальная форма:

В структуре MVD, определённой на множестве атрибутов  $A_R$ , имеются только тривиальные MVD и/или такие нетривиальные MVD, что левая часть любой из них является ключом  $R$ .

## Часть II Язык SQL

**Примечание.** В этой части предполагается, что читатель в состоянии самостоятельно сходу написать несколько десятков запросов на каждую конструкцию. В презентации было слайдов 150 только на примеры.

## 16. Язык запросов SQL: операции реляционной алгебры

Проекция:

```
SELECT name FROM person
```

Селекция:

```
SELECT name, price FROM items WHERE price >= 100 AND price <= 150
```

Декартово произведение:

```
SELECT * FROM person, dept
```

Соединение:

- Внутреннее:

```
SELECT ... FROM t_1, t_2 WHERE t_1.f_1 = t_2.f_2
SELECT ... FROM t_1 [INNER] JOIN t_2 ON t_1.f_1 = t_2.f_2
```

- Внешнее:

- JOIN: строки, в которых есть хотя бы одно совпадение в обеих таблицах;
- LEFT JOIN: строки из левой таблицы, даже если нет совпадения в правой;
- RIGHT JOIN: строки из правой таблицы, даже если нет соединения в левой;
- FULL JOIN: строки из обеих таблиц.

- Естественное:

```
SELECT group, st FROM S JOIN R ON S.lang = R.lang
SELECT * FROM S JOIN R USING (lang)
SELECT * FROM S NATURAL JOIN
```

Теоретико-множественные операции:

- UNION;
- INTERSECT;
- EXCEPT.

Отношения должны быть совместимы, т. е. иметь одинаковое количество полей с совместимыми типами данных. По-умолчанию все операции DISTINCT.

## 17. Язык запросов SQL: вложенные подзапросы

**Определение 21.** Вложенный SQL-запрос — это отдельный запрос, который используется внутри SQL инструкции.

Инструкцию, в которой используется вложенный запрос, называют *внешним SQL-запросом*.

Вложенные запросы:

- SELECT:

```
SELECT *, (SELECT name FROM car_owner WHERE inn = owner) FROM car
```

- FROM:

```
SELECT region FROM (SELECT * FROM car WHERE ...)
```

- WHERE:

```
SELECT mark FROM T WHERE mark = (SELECT MAX(mark) FROM T)
```

## 18. Язык запросов SQL: агрегирование и упорядочение

Агрегатные функции:

- COUNT — количество записей в выходном наборе;
- MIN, MAX — наименьшее/наибольшее из множества значений;
- AVG — среднее значение;
- SUM — сумма множества значений.

Упорядочение:

```
SELECT name FROM person ORDER BY name [DESC | ASC] [NULLS {FIRST | LAST}]
```

## 19. Типы данных SQL

Типы данных могут различаться в зависимости от реализации SQL. Простейшими типами данных в PostgreSQL можно считать:

- целочисленные значения;
- вещественные значения;
- строки фиксированной длины;
- строки переменной длины;
- дата и время.

Числовые типы:

Имя	Размер, байт	Описание
SMALLINT	2	целое в небольшом диапазоне
INTEGER	4	целое
BIGINT	8	целое в большом диапазоне
DECIMAL, NUMERIC	переменный	вещественное число с указанной точностью
REAL	4	вещественное число с переменной точностью
DOUBLE PRECISION	8	вещественное число с переменной точностью
SMALLSERIAL	2	небольшое целое с автоувеличением
SERIAL	4	целое с автоувеличением
BIGSERIAL	8	большое целое с автоувеличением

Строковые типы:

Имя	Описание
CHARACTER VARYING(n)	строка ограниченной переменной длины
CHARACTER(n)	строка фиксированной длины, дополненная пробелами
TEXT	строка неограниченной переменной длины

Типы даты и времени:

Имя	Размер, байт	Описание
TIMESTAMP [(p)] [WITHOUT TIME ZONE]	8	дата и время (без часового пояса)
TIMESTAMP [(p)] WITH TIME ZONE	8	дата и время (с часовым поясом)
DATE	4	дата
TIME [(p)] [WITHOUT TIME ZONE]	8	время суток
TIME [(p)] WITH TIME ZONE	12	время суток (с часовым поясом)
INTERVAL [поля] [(p)]	16	временной интервал

p — количество знаков после запятой в поле секунд (от 0 до 6).

Задание пользовательских типов:

```
CREATE TYPE mytype AS VARCHAR(11) NOT NULL
CREATE TYPE mood AS ENUM ('sad', 'ok', 'happy')
```

Явное преобразование типов:

```
SELECT CAST (10.6 AS INT)
SELECT CONVERT (INT, 10.6)
```

## 20. Временные таблицы и табличные переменные

**TODO:** Временные таблицы и табличные переменные.

## 21. Представления (обычные, с фиксацией схемы, индексируемые)

Представления используются:

- для упрощения и настройки восприятия информации в БД каждым пользователем;

- в качестве механизма безопасности;
- для предоставления интерфейса обратной совместимости.

Создание представлений:

```
CREATE [OR REPLACE] VIEW citizens AS SELECT client_id, last_name, city FROM client WHERE client = 'Moscow'
```

Изменение данных в представлении возможно, если при его создании:

- не используется ключевое слово DISTINCT;
- данные извлекаются только из одной таблицы;
- в списке полей нет арифметических выражений;
- не используются подзапросы;
- не используется группирование;
- не используется UNION;

Создание материализованных представлений:

```
CREATE MATERIALIZED VIEW view_name AS select_stmt [WITH [NO] DATA]
```

WITH NO DATA — представление создаётся, но не заполняется данными.

Обновление данных:

```
REFRESH MATERIALIZED VIEW view_name
REFRESH MATERIALIZED VIEW CONCURRENTLY view_name
```

Во втором случае необходим уникальный индекс, построенный до обновления.

Создание временных представлений (создаются на действующую сессию):

```
CREATE TEMP VIEW view_name AS select_stmt
```

## 22. Обобщённые табличные выражения

**Определение 22.** *Обобщённое табличное представление* представляет собой временно именованный результирующий набор (CTE), за которым следуют одиночные инструкции SELECT, INSERT, UPDATE или DELETE.

Структура CTE:

```
WITH expr_name (col_name[, ...]) AS (CTE_query_definition)
```

Рекурсивные обобщённые табличные выражения:

```
WITH RECURSIVE t(n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM t WHERE n < 15) SELECT n FROM t
WITH RECURSIVE t(n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM t) SELECT n FROM t LIMIT 100
```

## 23. Хранимые процедуры в базе данных

**Определение 23.** *Хранимая процедура* — это набор операторов процедурного языка и языка SQL, который компилируется системой в единый “план исполнения”.

Хранимые процедуры хранятся как объекты в базе. Они создаются и удаляются операторами CREATE и DROP.

```
CREATE [OR REPLACE] PROCEDURE my_sql_proc (a INT) LANGUAGE SQL AS
    $$ INSERT INTO tbl (field_1) VALUES (a) $$
CALL my_sql_proc (100)
```

## 24. Курсоры в базах данных

**Определение 24.** *Курсор* — это область в памяти базы данных, которая предназначена для хранения запроса SQL.

В памяти сохраняется и строка данных запроса, называемая *текущим значением*, или *текущей строкой* курсора. Указанная область в памяти поименована и доступна для прикладных программ.

Действия с курсорами:

- DECLARE — создание курсора;
- OPEN — открытие курсора, т. е. наполнение его данными;
- FETCH — выборка из курсора и изменение строк данных с его помощью;
- CLOSE — закрытие курсора;
- DEALLOCATE — освобождение курсора.

```
DECLARE c_name [[NO] SCROLL] CURSOR [(param)] FOR SELECT ...  
DECLARE c_name REFCURSOR
```

В схеме со *статическим курсором* информация читается из БД один раз и хранится в виде моментального снимка. На время открытия курсора сервер устанавливает блокировку на все строки, включённые его полный результирующий набор. Статический курсор не изменяется после создания и всегда отображает тот набор данных, который существовал на момент его открытия.

```
DECLARE c_name INSENSITIVE [SCROLL] CURSOR FOR select_stmt
```

*Динамические курсоры* отражают все изменения строк в результирующем наборе при прокрутке курсора. Значения типа данных, порядок и членство строк в результирующем наборе могут меняться для каждой выборки. Обновления видны сразу, если они сделаны посредством курсора.

```
DECLARE c_name [SCROLL] CURSOR FOR select_stmt  
[FOR {READ_ONLY | UPDATE [OF col[, ...]]}]
```

Считать текущую строку в переменные:

```
DECLARE id INT, name VARCHAR;  
FETCH FROM c_1 INTO id, name;
```

Прокручиваемый курсор позволяет использовать больше возможностей FETCH:

```
FETCH {NEXT | PRIOR | FIRST | LAST | ABSOLUTE n | RELATIVE n} FROM ...
```

**Замечание.** Курсор почти всегда использует дополнительные ресурсы сервера, что ведёт к резкому падению производительности.

## 25. Скалярные функции, определяемые пользователем

```
CREATE [OR REPLACE] FUNCTION name (params) RETURNS res_type {IMMUTABLE | STABLE | VOLATILE} AS  
$$ ... $$ LANGUAGE plpgsql
```

Функции:

- VOLATILE — может изменять данные в базе; может возвращать различные результаты с одинаковыми аргументами;
- STABLE — не может изменять данные в базе; возвращает одинаковый результат для всех вызовов с одинаковыми аргументами в одном операторе;
- IMMUTABLE — не может изменять данные в базе; всегда возвращает одинаковые результаты для одних и тех же аргументов.

```
CREATE FUNCTION fun_1 (a INT) RETURNS INT AS  
$$ BEGIN  
    RETURN a + 1;  
END; $$ LANGUAGE plpgsql;
```

## 26. Табличные функции, определяемые пользователем

```
CREATE FUNCTION get_table (a INTEGER)
  RETURNS TABLE (
    new_id INT,
    new_name TEXT
  )
  AS $$ BEGIN
    RETURN QUERY SELECT id, name FROM table_1 WHERE id > a;
  END; $$ LANGUAGE plpgsql;
```

## 27. Активные базы данных: триггеры

**Определение 25.** *Триггер* — это откомпилированная SQL-процедура. Исполнение обусловлено наступлением определённых событий внутри базы.

Назначение триггеров:

- проверка корректности введённых данных и выполнение сложных ограничений целостности;
- накопление аудиторской информации;
- автоматическое оповещение других модулей об изменениях информации;
- реализация бизнес-правил;
- организация каскадных воздействий на таблицы БД;
- поддержка репликации.

```
CREATE TRIGGER tr BEFORE UPDATE ON table_1 FOR EACH ROW EXECUTE PROCEDURE pt()
```

При запуске триггера есть доступ к двум записям NEW и OLD.

Пример триггерной функции:

```
CREATE FUNCTION fun() RETURNS TRIGGER AS
  $$ BEGIN
    IF NEW.id IS NULL THEN
      RAISE EXCEPTION 'id cannot be null';
    END IF;
    RETURN NEW;
  END; $$ LANGUAGE plpgsql;
```

## Часть III

# Индексы

## 28. Кластеризованные таблицы и индексы

Файлы конфигурации и файлы данных, используемые кластером базы данных, традиционно хранятся вместе в каталоге данных кластера, который обычно называют PGDATA.

Для каждой БД в кластере существует подкаталог внутри PGDATA/base, названный по OID базы в pg\_database. Этот подкаталог по-умолчанию является местом хранения файлов базы данных.

Каждая таблица и индекс хранятся в отдельном файле. Для каждой таблицы и индекса есть карта свободного пространства. Когда объём таблицы или индекса превышает заданный размер, она делится на сегменты (ноды).

Основной единицей хранения данных является страница. Место на диске для размещения файла данных в БД логически разделяется на страницы с непрерывной нумерацией. Дисковые операции ввода-вывода осуществляются на уровне страницы. СУБД считывает или записывает целые страницы данных.

- 29. Первичные и вторичные индексы. Плотные и неплотные индексы
- 30. Индексы: В-дерево
- 31. Индексы: отфильтрованные, покрывающие, составной ключ
- 32. Индексы: полнотекстовые
- 33. Индексы: хеширование
- 34. Индексы: битовые шкалы
- 35. Индексы: R-дерево