Базовые принципы реляционной модели баз данных. Правила Кодда

1. Каждая таблица представляет отдельный тип данных: реляционная модель предполагает, что информация хранится в виде набора таблиц, каждая из которых содержит данные по определенному типу объектов или явлений.

2. Каждая таблица имеет уникальное имя: каждая таблица в базе данных должна иметь уникальное имя, позволяющее обращаться к ней из других таблиц.

3. В таблице хранятся записи: каждая запись в таблице представляет собой конкретный объект или явление, для которого записываются соответствующие атрибуты.

4. Каждый атрибут имеет уникальное имя: атрибуты являются полями, которые содержат данные об объектах в таблице. Каждый атрибут должен иметь уникальное имя, чтобы можно было обращаться к нему из других таблиц.

5. Каждый атрибут должен иметь определенный тип данных: реляционная модель баз данных предполагает, что каждый атрибут имеет определенный тип данных, например, числовой, дата-время и т.д.

6. Каждая запись является уникальной: записи в таблице должны содержать уникальные значения, чтобы можно было однозначно идентифицировать каждый объект или явление.

7. Связи между таблицами осуществляются через ключи: связи между таблицами в реляционной модели баз данных осуществляются через ключи, которые связывают записи в разных таблицах на основе общего атрибута или совокупности атрибутов.

8. Нормализация данных: процесс, в ходе которого данные в таблице разбиваются на более мелкие части, чтобы обеспечить их логическую структуру и уменьшить дублирование информации.

**Правила Э.Ф.Кодда**

1. Данные хранятся в столбцах и строках таблиц

2. Доступ к данным можно получить, указав имя таблицы, имя столбца и ключ

3. СУБД должна обрабатывать пропущенные значения (пустые данные). Для обозначения пустых данных используется ключевое слово NULL

4. СУБД должна включать оперативный каталог, содержащий сведения о самой базе данных

5. Для определения данных, их обработки и других операций определяется специальный подъязык. В настоящее время таким языком является SQL

6. Должны поддерживаться представления таблиц, или виртуальные таблицы, которые строятся динамически по запросам

7. Должна быть включена поддержка транзакций. Транзакция обеспечивает коллективное изменение или отмену всех связанных изменений данных. Транзакции имеют четыре свойства: атомарность, согласованность, изолированность и продолжительность (ACID).

Атомарность – транзакция либо выполняется полностью, либо не выполняет ничего.

Согласованность – с каждой транзакцией БД переходит из одного согласованного состояния в другое согласованное состояние данных.

Изолированность – изменения, происходящие в процессе транзакции, не видны пользователям до завершения транзакции.

Продолжительность – сделанные в процессе транзакции изменения должны сохраниться в БД

8. Физическое хранение данных отделено от пользователя. Пользователь имеет дело только с логической структурой БД

9. Логическую структуру данных можно изменять с минимальным воздействием на пользователей и программы

10.Правила целостности данных хранятся в каталоге БД. Любые их изменения не должны влиять на прикладные программы

11.Приложения должны работать в распределенной среде (когда данные хранятся в различных местах)

12. СУБД должна обеспечивать безопасность и целостность базы данных

# Хранимые процедуры PL/SQL, средства их создания, изменения и удаления

Хранимая процедура PL/SQL - это блок кода, который сохраняется в базе данных Oracle и может быть вызван из другой процедуры, триггера или приложения. Они используются для управления и обработки данных в базе данных и могут выполнять различные задачи, например, создание, изменение или удаление записей в таблицах базы данных.

PL/SQL-процедуры создаются с помощью команды CREATE PROCEDURE. Код процедуры размещается между ключевыми словами BEGIN и END. Параметры процедуры могут быть переданы через список параметров, который указывается после имени процедуры.

Пример создания процедуры:

CREATE PROCEDURE update\_salary (emp\_id IN NUMBER, salary IN NUMBER)

IS

BEGIN

UPDATE employees SET salary = salary WHERE employee\_id = emp\_id;

COMMIT;

END;

Для изменения хранимой процедуры необходимо использовать команду ALTER PROCEDURE. При этом можно изменять параметры процедуры, ее тело и другие свойства.

Пример изменения процедуры:

ALTER PROCEDURE update\_salary (emp\_id IN NUMBER, salary IN NUMBER)

IS

BEGIN

UPDATE employees SET salary = salary + 500 WHERE employee\_id = emp\_id;

COMMIT;

END;

Для удаления процедуры необходимо использовать команду DROP PROCEDURE, за которой следует имя удаляемой процедуры.

Пример удаления процедуры:

DROP PROCEDURE update\_salary;

# Архитектура системы Oracle. База данных, экземпля

## Логическая структура базы данных

* **одно или несколько табличных пространств**
* **пользовательские объекты базы данных (например, таблицы, представления, индексы, кластеры, последовательности и хранимые процедуры)**

## Табличные пространства

База данных

**Табличное пространство**

**Табличное пространство**

**DATAFILE1**

**DATAFILE2 DATAFILE3**

**Диск 2**

Диск 1

## Схемы и объекты схем

***Схема* – это набор объектов.**

Объекты схемы – это логические структуры, которые непосредственно связаны с данными БД.

* **Таблица – это основная единица хранения данных в базе данных Oracle**
* **Представления – это взгляд пользователя на данные, принадлежащие одной или нескольким таблицам**
* **Последовательность строит последовательный список уникальных чисел для числовых столбцов таблиц базы данных**
* **Программные единицы – процедуры, функции и пакеты**
* **Синоним – это альтернативное имя таблицы, представления, последовательности или программной единицы**
* **Индексы, кластеры и хеш-кластеры – это необязательные структуры, связанные с таблицами. Ускоряют обработку данных**

Блоки данных, экстенты и сегменты – логические структуры для выделения дискового пространства под объекты БД.

## Физические структуры

* **Файлы данных. Каждая база данных Oracle имеет несколько (от одного и более) физических файлов данных. Файлы данных базы данных содержат все данные БД**
* **Журнальные файлы. Каждая база данных Oracle имеет набор из двух и более журнальных файлов. Набор журнальных файлов БД называется журналом базы данных. Главным назначением журнала является сбор и хранение всех изменений, сделанных в данных**
* **Управляющие файлы. Каждая БД Oracle имеет *управляющий файл*. Управляющий файл хранит информацию о физической структуре БД**

## Экземпляр базы данных Oracle

Экземпляр – это структуры Oracle в физической памяти и набор процессов, которые выполняют специфические операции, связанные с работой БД.

# Пакеты PL/SQL, средства их создания, изменения и удаления. Структура пакета

Пакеты PL/SQL представляют собой набор объектов PL/SQL, который обычно содержит определения типов данных, константы, процедуры и функции. Они используются для структурирования приложения, повторного использования кода и ускорения выполнения задач.

Создание пакета происходит с помощью команды CREATE PACKAGE. Описание объектов пакета размещается между ключевыми словами PACKAGE и END PACKAGE. Каждый объект, такой как тип данных, константа, процедура или функция, определяется в отдельном блоке в рамках пакета.

Пример создания пакета:

CREATE PACKAGE payroll\_pkg IS

TYPE emp\_salary IS RECORD (emp\_id NUMBER, salary NUMBER);

PROCEDURE get\_salary (emp\_id IN NUMBER, salary OUT NUMBER);

END payroll\_pkg;

Для изменения пакета необходимо использовать команду ALTER PACKAGE. При этом можно изменять определения объектов в рамках пакета.

Пример изменения пакета:

ALTER PACKAGE payroll\_pkg ADD PROCEDURE increase\_salary (emp\_id IN NUMBER, amount IN NUMBER);

Для удаления пакета необходимо использовать команду DROP PACKAGE, за которой следует имя удаляемого пакета.

Пример удаления пакета:

DROP PACKAGE payroll\_pkg;

Структура пакета обычно состоит из двух разделов: спецификации и тела. Спецификация содержит объявления объектов, которые доступны вне пакета, в то время как тело содержит реализацию процедур и функций.

Пример структуры пакета:

CREATE PACKAGE payroll\_pkg IS

PROCEDURE calculate\_payroll (start\_date IN DATE, end\_date IN DATE);

END payroll\_pkg;

CREATE PACKAGE BODY payroll\_pkg IS

PROCEDURE calculate\_payroll (start\_date IN DATE, end\_date IN DATE)

IS

-- реализация расчета заработной платы

END calculate\_payroll;

END payroll\_pkg;

# Правила Кодда. Физическая структура базы данных Oracle

Правила Кодда (Codd's Rules) - это теоретические принципы, определяющие требования к объектно-реляционным системам управления базами данных (ORDBMS). Они были разработаны Эдгаром Коддом в 1985 году и включают 12 правил, которые должны быть выполнены в любой объектно-реляционной системе.

Некоторые из правил Кодда включают следующее:

- Система должна уметь хранить произвольные объекты, включая текст, графику и звук.

- Все данные должны быть представлены в виде таблиц.

- Все данные должны быть логически завершены и однозначно определены.

- Поддерживаемые языки должны быть декларативными, а не императивными.

Физическая структура базы данных Oracle включает в себя несколько компонентов, которые взаимодействуют друг с другом для обеспечения хранения, доступа и управления данными. Эти компоненты включают в себя следующее:

- Таблицы: это основной объект хранения данных в базе данных. Каждая таблица состоит из столбцов и строк, которые содержат фактические данные.

- Индексы: это объекты, которые используются для ускорения поиска и выборки данных из таблиц. Они хранят данные в отсортированном порядке и обеспечивают быстрый доступ к данным.

- Представления: это объекты, которые представляют данные из одной или нескольких таблиц в виде виртуальной таблицы. Они упрощают доступ к данным и уменьшают количество кода, необходимого для работы с данными.

- Хранимые процедуры: это блоки кода PL/SQL, которые хранятся в базе данных и могут быть вызваны из других процедур, триггеров или приложений. Они используются для управления и обработки данных в базе данных и могут выполнять различные задачи.

- Триггеры: это блоки кода PL/SQL, которые вызываются автоматически при определенных событиях в базе данных, таких как вставка, обновление или удаление записей в таблицах. Они используются для выполнения автоматических операций на основе изменения данных в базе данных.

Физическая структура базы данных Oracle обеспечивает надежное хранение данных и упрощает доступ к данным, используя различные компоненты. Кроме того, она обладает рядом мощных функций, таких как транзакционность, репликацию и безопасность, что делает ее очень удобной для использования в крупных проектах.

# Триггеры PL/SQL, средства их создания, изменения и удаления

Триггеры PL/SQL - это блоки кода, которые выполняются автоматически при определенных событиях, таких как вставка, обновление или удаление в таблице базы данных. Они используются для выполнения автоматических операций на основе изменения данных в базе данных.

Создание триггера происходит с помощью команды CREATE TRIGGER. Код триггера размещается между ключевыми словами BEGIN и END. Триггер может быть включен перед или после события в таблице базы данных.

Пример создания триггера:

CREATE OR REPLACE TRIGGER update\_employee\_salary

BEFORE INSERT OR UPDATE OF salary, commission\_pct ON employees

FOR EACH ROW

BEGIN

IF :NEW.salary > 5000 THEN

:NEW.salary := 5000;

END IF;

END;

Для изменения триггера необходимо использовать команду ALTER TRIGGER. При этом можно изменять условия для выполнения триггера и тело триггера.

Пример изменения триггера:

ALTER TRIGGER update\_employee\_salary

ENABLE

FOR EACH ROW

BEGIN

IF :NEW.salary > 10000 THEN

:NEW.salary := 10000;

END IF;

END;

Для удаления триггера необходимо использовать команду DROP TRIGGER, за которой следует имя удаляемого триггера.

Пример удаления триггера:

DROP TRIGGER update\_employee\_salary;

Триггеры могут выполнять различные операции, такие как обновление данных, изменение значений поля, добавление данных в другую таблицу и т.д. Они очень полезны, когда необходимо выполнить автоматическую операцию на основе изменения данных в базе данных.

# Правила Кодда. Логическая структура базы данных Oracle

Правила Кодда (Codd's Rules) - это теоретические принципы, определяющие требования к объектно-реляционным системам управления базами данных (ORDBMS). Они были разработаны Эдгаром Коддом в 1985 году и включают 12 правил, которые должны быть выполнены в любой объектно-реляционной системе.

Некоторые из правил Кодда включают следующее:

- Система должна поддерживать язык высокого уровня, который позволяет пользователю описывать данные в терминах конкретных объектов и отношений между ними.

- Должна быть возможность использовать язык общего назначения для определения требований пользователя к данным.

- Система должна обеспечивать контроль целостности данных, включая сохранение уникальности данных, проверку ссылок и т.д.

- Система должна быть способна поддерживать все виды запросов на выборку и обновление данных, использующие различные типы связей между объектами.

Логическая структура базы данных Oracle - это способ организации данных в базе данных, который определяет взаимосвязи между таблицами и объектами в базе данных. Каждая таблица в базе данных представляет собой набор строк и столбцов, которые содержат фактические данные.

База данных Oracle поддерживает логическую структуру данных, основанную на реляционной модели, которая опирается на теорию множеств и логический алгебру. Каждая таблица в базе данных представляется в виде отношения, которое состоит из набора столбцов и строк. Столбцы представляют собой атрибуты таблицы, в то время как строки представляют фактические данные.

Логическая структура базы данных Oracle также включает в себя связи между таблицами. Связи между таблицами представляют собой отношения между атрибутами таблиц, которые связаны между собой в базе данных. Связи могут быть односторонними или двусторонними, и могут быть выражены в виде внешних ключей и других типов ограничений.

Логическая структура базы данных Oracle обеспечивает эффективное хранение и управление данными в базе данных, позволяя пользователям быстро и легко получать доступ к нужным данным. Она является ключевым элементом объектно-реляционных систем управления базами данных, обеспечивая высокую производительность и надежность.