**1. Основы реляционной алгебры**

Реляционной базой данных называется совокупность отношений, содержащих всю информацию, которая должна храниться в базе.

Каждая строка в таблице является кортежем в реляционной теории. Множество упорядоченных кортежей называется отношением. Применительно к таблице домены – это столбцы.

Строгое определение отношения: Пусть даны N множеств D1,D2, …. Dn (домены), отношением R над этими множествами называется множество упорядоченных N-кортежей вида <d1,d1,...dn>, где d1 принадлежит D1 и тд. Множества D1,D2,..Dn называются доменами отношения R.

Каждый элемент кортежа представляет собой значение одного из атрибутов соответствующего одному из доменов.

В отношении требованием является то, что все кортежи должны различаться. Для однозначной идентификации кортежа существует первичный ключ. Первичный ключ – это атрибут или набор из минимального числа атрибутов, который однозначно идентифицирует конкретный кортеж и не содержит дополнительных атрибутов.

В реляционной БД таблицы взаимосвязаны и соотносятся друг с другом как главные и подчиненные. Связь главной и подчиненной таблицы осуществляется через первичный ключ главной таблицы и внешний ключ подчиненной таблицы.

Основные восемь операций реляционной алгебры были предложены Э.Коддом: объединение, пересечение, вычитание, декартово произведение, выборка, соединение, деление.

Для начала рассмотрим самую простую операцию — имя отношения. Её результатом будет такое же отношение, то есть выполнив операцию PRODUCTS, мы получим копию отношения PRODUCTS.

Проекция является операцией, при которой из отношения выделяются атрибуты только из указанных доменов, то есть из таблицы выбираются только нужные столбцы, при этом, если получится несколько одинаковых кортежей, то в результирующем отношении остается только по одному экземпляру подобного кортежа.

Выборка — это операция, которая выделяет множество строк в таблице, удовлетворяющих заданным условиям. Условием может быть любое логическое выражение.

Умножение или декартово произведение является операцией, производимой над двумя отношениями, в результате которой мы получаем отношение со всеми доменами из двух начальных отношений. Кортежи в этих доменах будут представлять из себя все возможные сочетания кортежей из начальных отношений.

Операция соединения обратна операции проекции и создает новое отношение из двух уже существующих. Новое отношение получается конкатенацией кортежей первого и второго отношений, при этом конкатенации подвергаются отношения, в которых совпадают значения заданных атрибутов. Естественное (натуральное) соединение отличается лишь тем, что при соединении двух таблиц, например, по ID, в результирующем отношении будет один столбец ID.

Результатом операции пересечения будет отношение, состоящее из кортежей, полностью входящих в состав обоих отношений.

Результатом вычитания будет отношение, состоящее из кортежей, которые являются кортежами первого отношения и не являются кортежами второго отношения.

**2. Архитектура системы баз данных, блоки данных, сегменты, экстенты**

Oracle – система управления базами данных, которая предоставляет открытый, всеобъемлющий, интегрированный подход к управлению информацией. Состоит из экземпляра и базы данных.

Экземпляр – средство доступа к базе данных Oracle. Состоит из набора процессов операционной системы и используемой ими памяти. Открывает только одну базу данных в один момент времени.

Физическая структура БД Oracle определяется файлами операционной системы, которые обеспечивают физическое хранение информации в базе данных. (Control files, Data files, Redo log files). Логическая структура представлена иерархией: tablespaces, segments, extends, blocks.

SYSTEM tablespace. Создается вместе с базой данных. Содержит словарь данных. Содержит сегмент отката SYSTEM. Non-SYSTEM tablespaces. Содержит пользовательские сегменты. Управляются администратором БД.

Типы данных Oracle: пользовательские и встроенные: скалярные (CHAR, VARCHAR2, NUMBER и др.), коллекции (VARRAY, TABLE), отношения (REF).

Таблицы в Oracle: обычные таблицы, секционированные таблицы, организованные по индексу, кластерные.

Redo Log. В журналы повторного выполнения записываются все изменения в базе данных для проведения восстановления в случае сбоя. Redo log объединяются в группы. Необходимо как минимум 2 группы. Данные в файлах внутри группы идентичны. Группы Redo log используются циклично. Когда файлы группы заполнены, LGWR переключается на следующую группу. Это называется переключением логов. В этот момент происходит Checkpoint. Информация записывается в control file.

Undo. Перед тем как сделать изменения в блоке данных Oracle копирует его в Undo. Это необходимо для Rollback и Read Consistency.

Структура памяти Oracle состоит из двух основных областей: System Global Area (SGA): выделяется при старте экземпляра, базовый компонент экземпляра Oracle. Program Global Area (PGA): выделяется при запуске серверного процесса.

SGA состоит из следующих структур: Shared pool, Database buffer cache, Redo log buffer. Опциональные структуры SGA: Large pool, Java pool. PGA – это память, зарезервированная для процессов подключающихся к базе данных.

Взаимодействие между физическими и логическими структурами памяти осуществляется с помощью различных фоновых процессов.

Oracle использует неявную транзакционную модель. Транзакция начинается сразу за окончанием предыдущей. COMMIT или ROLLBACK заканчивают транзакцию.

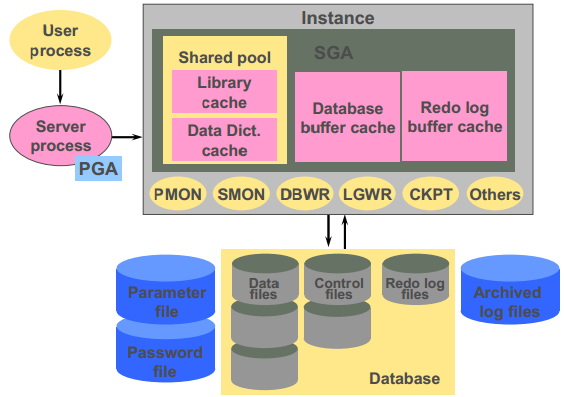
Во время создания базы данных сервер Oracle создает дополнительные объектные структуры в файлах данных: таблицы словаря данных и таблицы dynamic performance.

Словарь данных – это набор read-only таблиц и представлений, которые содержат информацию о базе данных. Описывает базу данных и ее объекты. Словарь данных предоставляет информацию о: логической и физической структуре БД, свойствах и размерах объектов, ограничениях целостности, пользователях, ролях, привилегиях, аудите.

Блоки данных (Data Block) - мельчайший строительный блок базы данных Oracle, состоящий из определенного количества байт на диске. Блок данных Oracle - логический компонент базы данных. Диски на которых располагаются блоки Oracle, сами делятся на блоки данных. Обычно блоки данных диска соответствуют блокам данных Oracle. Размер блока базы данных Oracle устанавливается параметром DB\_BLOCK\_SIZE в файле init.ora. Размер блока следует воспринимать, как минимальную единицу обновления, выбора или вставки данных.

Экстенты (extent) - это два или более последовательных блоков данных Oracle, представляющий собой единицу выделения места на диске. Когда комбинируется несколько непрерывных блоков данных, они называются экстентом. Когда вы создаете объект базы данных вроде таблицы или индекса, вы выделяете им некоторый начальный объем пространства, называемый начальным экстентом, и, кроме того, указываете размер следующего экстента.

Сегменты (segments) - набор экстентов, которые вы выделяете логической структуре, такой как таблица или индекс (или некоторый другой объект). Набор экстентов формирует следующую более крупную единицу хранения, именуемую сегментом. Oracle называет сегментом все пространство, выделенное любому конкретному объекту базы данных.



**8. TCL, DML, DDL, DCL**

DDL

DDL - (Data Defenition Language) предложения для определения структуры базы данных или схемы. Примеры:

- CREATE - создает объекты базы данных (таблицы, представления и т.д.)

- ALTER - Изменяет структуру и объекты базы данных

- DROP - Удаляет объекты базы данных

- TRUNCATE - Удаляет все записи из таблицы

- COMMENT - Добавляет комментарии в словарь данных

- RENAME - Переименовывает объект (alter table <old\_name> rename to <new\_name>)

DML

DML - (Data Manipulation Language) предложения для управления данными. Примеры:

- SELECT - Возвращает данные из базы данных

- INSERT - Вставляет данные в таблицу

- UPDATE - Обновляет существующие данные в таблице

- DELETE - Удаляет все записи в таблице

- MERGE - UPSERT операция (insert или update)

- CALL - вызов подпрограммы PL/SQL или Java

- EXPLAIN PLAN - Предоставляет план запроса

- LOCK TABLE - Управление параллелизмом

DCL

DCL - Data Control Language. Примеры:

- GRANT - Дает пользователю привелегии доступа к базе данных и ее объектам

- REVOKE - Забирает у порльзователя привелегии данные командой GRANT

TCL

TCL - (Transaction Control) предложения используемые для управления изменениями сделанными предложениями DML. Это позволяет объединять предложения DML в логические транзакции.

- COMMIT - Сохраняет изменения

- SAVEPOINT - Определяет точку транзакции до которой потом можно откатиться

- ROLLBACK - Восстанавливает базу данных на момент последней операции COMMIT. Откатывает транзакцию

- SET TRANSACTION - Изменяет опции транзакции, такие как: уровень изоляции и какой сегмент отката использовать

Предложения DML автоматически не сохраняются, т.е. вы можете использовать откат транзакции, но результаты DDL предложений сохраняются автоматически.

**20. Синонимы**

Синоним (Synonym) – это альтернативное имя (псевдоним) для объекта схемы. Если для какого-либо объекта базы данных Oracle существует синоним, то к объекту из SQL запроса можно обращаться либо по его настоящему имени, либо по синониму. Так же они обеспечивают некоторый уровень безопасности, поскольку скрывают имя объекта и его владельца, а также делают прозрачным местоположение удаленных объектов распределенных баз данных.

Синонимы позволяют переименовывать и перемещать базовые объекты. При том переопределяется только синоним, а приложение не требует никаких модификаций.

Различают два типа синонимов:

Частный (PRIVATE)- синонимы содержаться в схеме конкретного пользователя и доступны только самому пользователю, и тем, кому он предоставил соответствующие права доступа.

Общий (PUBLIC)- этими синонимами владеет специальная группа пользователей – PUBLIC, в результате чего эти синонимы доступны всем пользователям базы данных.

Чтобы создать частный синоним необходима привилегия CREATE SYNONYM, для того чтобы иметь права на создание синонима в схеме другого пользователя, необходима привилегия CREATE ANY SYNONYM. Для создания общих синонимов, необходима привилегия CREATE PUBLIC SYNONYM.

Для создания синонимов используется конструкция CREATE SYNONYM или CREATE PUBLIC SYNONYM, в зависимости от типа создаваемого синонима. При создании наличие реального объекта и привилегий доступа к нему не требуется.

CREATE SYNONYM [имя\_синонима] FOR[имя\_объекта]

CREATE PUBLIC SYNONYM [имя\_синонима] FOR[имя\_объекта]

В своей схеме пользователь может удалять любые частные синонимы. Для удаления частных синонимов в схеме другого пользователя, необходима привилегия DROP ANY SYNONYM, для удаления общих – DROP PUBLIC SYNONYM.

Для удаления используется оператор – DROP SYNONYM. Для удаления общего синонима - DROP PUBLIC SYNONYM.

При удалении синонима, его определение удаляется из словаря данных. Все объекты ссылающиеся на удаленный синоним остаются, однако они становятся недействительными и их использование невозможно. Объекты, для которых создавался синоним остаются без изменений и доступны для использования.

Если удален объект, для которого создавался синоним, то синоним остается, но при обращении к нему будет сообщено об ошибке. Если объект, для которого создавался синоним, был пересоздан, то синоним требуется перекомпилировать. Для этого используется предложение:

ALTER SYNONYM [имя\_синонима] COMPILE;

**17. Коллекции и работа с ними**

Коллекцией называется упорядоченная группа элементов одного типа. Язык PL/SQL поддерживает три вида коллекций: вложенные, индексированные таблицы и varray-массивы

Доступ к любому элементу вложенной таблицы или varray-массива осуществляется по его индексу, который указывается в скобках после имени переменной типа коллекции.

Для создания коллекции следует определить тип коллекции – TABLE или VARRAY – и объявить переменную этого типа. Определение типа выполняется в секции объявлений блока PL/SQL, подпрограммы или пакета.

TYPE type\_name IS TABLE OF element\_type [NOT NULL];

Параметр type\_name указывает имя определяемого типа, а element\_type - это любой допустимый тип данных PL/SQL, исключая некоторые типы, в том числе VARRAY, TABLE, BOOLEAN, LONG, REF CURSOR и т.п.

Вложенную таблицу можно рассматривать как одномерный массив, в котором индексами служат значения целочисленного типа в диапазоне от 1 до 2147483647. Вложенная таблица может иметь пустые элементы, которые появляются после их удаления встроенной процедурой DELETE. Вложенная таблица может динамически увеличиваться.

Индексированные таблицы позволяют работать со столбцами как с единой переменной - массивом.Определение индексированной таблицы (index-by tables):

TYPE type\_name IS TABLE OF element\_type [NOT NULL] INDEX BY BINARY\_INTEGER;

Индексированная таблица - это вариант вложенной таблицы, в которой элементы могут иметь произвольные целочисленные значения индексов. Такой тип данных очень удобен, если в качестве индекса использовать значение первичного ключа.

Определение типа Varray-массива может иметь следующее формальное описание:

TYPE type\_name IS {VARRAY | VARYING ARRAY} (size\_limit) OF element\_type [NOT NULL];

Параметр type\_name указывает имя определяемого типа, size\_limit - максимальное количество элементов, а element\_type - это любой допустимый тип данных PL/SQL, исключая некоторые типы, такие как VARRAY, TABLE, BOOLEAN, LONG, REF CURSOR и т.п.

Максимальное количество элементов в Varray-массиве указывается при определении типа и не может изменяться динамически. Доступ к каждому элементу Varray-массива осуществляется по индексу. Varray-массивы можно передавать в качестве параметров. Varray-массивы не могут иметь пустот, так как для них нет операции удаления произвольного элемента массива.

DECLARE

TYPE d1 IS VARRAY(365) OF DATE;

TYPE rec1 IS RECORD (v1 VARCHAR2(10), v2 VARCHAR2(10));

- Массив записей

TYPE arr\_rec IS VARRAY(250) OF rec1;

- Вложенная таблица

TYPE F1T1 IS TABLE OF tbl1.f1%TYPE;

CURSOR c1 IS SELECT \* FROM tbl1;

- Массив записей, основанный на курсоре

TYPE t1 IS VARRAY(50) OF c1%ROWTYPE;

- Индексированная таблица

TYPE t2 IS TABLE OF tbl1%ROWTYPE INDEX BY BINARY\_INTEGER;

- Объявление переменной

rec\_t2 t2;

BEGIN

/\* Использование переменной типа "индексированная таблица" \*/

SELECT \* INTO rec\_t2(120) WHERE f1 = 120;

END;

Для инициализации коллекции используется конструктор - автоматически создаваемая функция, одноименная с типом коллекции. Конструктор без параметров – пустая коллекция.

В PL/SQL реализован ряд встроенных методов для работы с коллекциями. Эти методы вызываются как collection\_name.method\_name[(parameters)]

Методы: EXISTS(n), COUNT, LIMIT (для вложенных таблиц возвращает NULL), DELETE (m,n) (указываем только n – удаляется элемент n; m и n – с m до n; без аргументов – удалить всё), FIRST, LAST (наименьший и наибольший индекс элементов), PRIOR(n) (индекс элкмента, предшествующего n), NEXT(n) (индекс следующего), EXTEND (n,i) (Если параметров не задано, то в коллекцию добавляется один null-элемент, а если указан только параметр n, то добавляются n null-элементов. Если задано оба параметра, то добавляются n элементов, являющихся копиями i-го элемента коллекции), TRIM(n) (Если параметры не указаны, то удаляется один последний элемент, а при задании параметра удаляются n последних элементов коллекции. Если значение параметра превышает реальное количество элементов, возвращаемое функцией COUNT, то инициируется исключение. Если элемент был ранее удален функцией DELETE, то он все равно будет входить в число удаляемых функцией TRIM элементов).

**18. Управление правами пользователей: объектные, системные привилегии, роли**

DDL-операторы GRAND и REVOKE. Эти операторы нельзя использовать непосредственно в PL/SQL. Существуют привилегии двух различных видов: объектные и системные. Объектная привилегия (object privilege) разрешает выполнение определенной операции над конкретным объектом (например, над таблицей). Системная привилегия (system privilege) разрешает выполнение операций над целым классом объектов. Существует множество системных привилегий, соответствующих практически всем возможным операциям DDL. Например, системная привилегия CREATE TABLE, позволяет ее обладателю создавать таблицы. А, вот системная привилегия CREATE ANY TABLE дает возможность создавать таблицы в других схемах.

GRANT system-privilege/role to user/role/public WITH ADMIN OPTION

system\_privilege - предоставляемое системное полномочие.

role - предоставляемая роль.

TO - определяет пользователей или роли, которым предоставляются системные полномочия.

PUBLIC – указывает, что системные полномочия, определяемые администратором, предоставляются всем пользователям.

WITH ADMIN OPTION - позволяет получившему системные полномочия или роль предоставлять их в дальнейшем другими пользователям или ролям. Такое решение в частности включает и возможность изменение или удаления роли.

Основных операций в языке DDL три - это CREATE, ALTER, DROP.

Группа ALTER:

ALTER DATABASE - Позволяет изменять саму БД.

ALTER USER - Позволяет изменять пользователя и его параметры (пароль, профиль, роль)

ALTER PROFILE - Позволяет изменять профили.

ALTER TABLESPACE - Позволяет изменять табличные пространства.

Для любого объекта - ANY:

ALTER ANY PROCEDURE - Разрешает изменение любой хранимой функции процедуры или пакета в любой схеме.

ALTER ANY ROLE - Разрешает изменение любой роли БД.

ALTER ANY SEQUENCE - Разрешает изменение любой последовательности в БД.

ALTER ANY TABLE - Разрешает изменение любой таблицы или вида в схеме БД.

ALTER ANY TRIGGER - Позволяет разрешать, запрещать компилировать любой триггер в любой схеме БД.

ALTER ANY INDEX - Разрешает изменение любого индекса в любой схеме.

Группа CREATE:

Позволяет создавать в [любой] схеме соответствующий объект:

CREATE [ANY] PROCEDURE/ SEQUENCE/ TABLE/ TRIGGER/ VIEW/ INDEX;

CREATE SESSION/ROLE;

Удаление объектов в [любой] схеме, а также очистка таблиц:

DELETE [ANY] TABLE/ PROCEDURE/ SEQUENCE/ TABLE/ TRIGGER/ VIEW/ INDEX;

И еще полезные системные привилегии:

GRANT EXECUTE [ANY] PROCEDURE

GRANT ANY PRIVILEGE/ ROLE; INSERT ANY TABLE; LOCK ANY TABLE;

SELECT ANY TABLE; SELECT ANY SEQUENCE;

Объектные привилегии:

GRANT object\_privilege/all/privilege column ON schema.object to USER/Role/Public WITH ADMIN OPTION

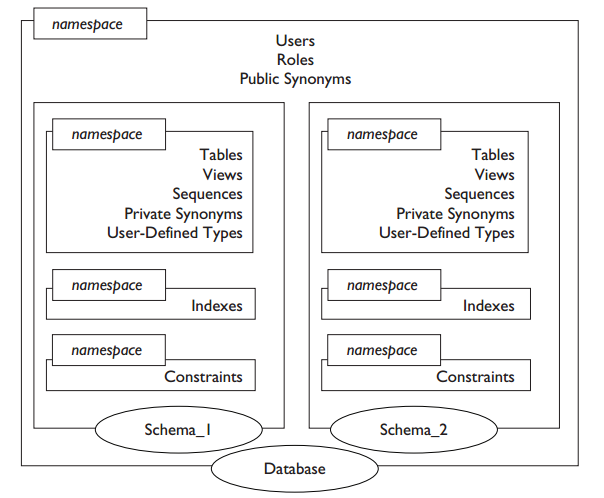
object\_privilege: ALTER,SELECT,UPDATE,DELETE,INSERT,EXECUTE,INDEX,REFERENCES

Роль - это комплексный набор полномочий. Для разграничения доступа роль можно наделить привелегиями и назначить пользователям. CREATE ROLE role1 NOT IDENTIFIED;

**GRANT role1 to usr0;**

**19. Пространства имен**

Пространство имен – некоторое множество, под которым подразумевается модель, абстрактное хранилище или окружение, созданное для логической группировки уникальных идентификаторов (т.е. имен).



**5. Нормальные формы (1-6, Бойса-Кодда)**

Первая нормальная форма (1NF)

* Все строки должны быть различными.
* Все элементы внутри ячеек должны быть атомарными (не списками). Другими словами, элемент является атомарным, если его нельзя разделить на части, которые могут использоваться в таблице независимо друг от друга.

Методы приведения к 1NF:

- Устраните повторяющиеся группы в отдельных таблицах (одинаковые строки)

- Создайте отдельную таблицу для каждого набора связанных данных

Вторая нормальная форма (2NF)

- Таблица должна находиться в первой нормальной форме

- Любое ее поле, не входящее в состав первичного ключа, функционально полно зависит от первичного ключа.

Методы приведения к 2NF:

- Создайте отдельные таблицы для наборов значений, относящихся к нескольким записям

- Свяжите эти таблицы с помощью внешнего ключа

Третья нормальная форма (3NF)

- Таблица находится во второй нормальной форме

- Любой ее не ключевой атрибут функционально зависит от первичного ключа

Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы (справочники).

Нормальная форма Бойса-Кодда

Эта форма почти то же, что и третья. С одним небольшим дополнительным условием.

- Таблица находится в третьей нормальной форме

- В таблице должен быть только один потенциальный первичный ключ

Другими словами, в таблице должен быть только один первичный ключ и не должно быть других потенциальных вариантов (например, наборе не ключевых полей этой таблицы).

Четвертая нормальная форма (4NF)

Ну, тут как и во всех предыдущих формах требования, включают в себя требования всех предыдущих форм + что-то еще. В это форме дополнительное правило должно исключать многозначные зависимости. Другими словами, все строки таблицы должны быть независимыми друг от друга. В том смысле, что наличие какой-то строки X, не должно означать, что строка Y тоже где-то есть в этой таблице.

Пятая нормальная форма (5NF)

Если таблица не может быть разбита на две таблицы без потерь зависимости, то говорят о наличии в ней зависимости по соединению (или зависимость соединения).

Таблица, в которой имеется зависимость соединения, не находится в пятой нормальной форме.

(Важно то, что нельзя разбить на ДВЕ таблицы без потерь зависимости, если можно разбить на большее количество таблиц, то после разбиения получим 5 нормальную форму)

Шестая нормальная форма (6NF)

Переменная отношения находится в шестой нормальной форме тогда и только тогда, когда она удовлетворяет всем нетривиальным зависимостям соединения. Из определения следует, что переменная находится в 6НФ тогда и только тогда, когда она неприводима, то есть не может быть подвергнута дальнейшей декомпозиции без потерь. Каждая переменная отношения, которая находится в 6НФ, также находится и в 5НФ.

**3. Реляционная и объектно-реляционная структура данных**

Реляционная база данных – база данных, основанная на реляционной модели данных. Для работы с реляционными БД применяют реляционные СУБД.

Объектно-реляционная база данных – база данных, поддерживающая некоторые технологии, реализующие объектно-ориентированный подход.

Реляционные базы данных используют набор таблиц, представляющих простые данные. Дополнительная или связанная информация хранится в других таблицах. Часто для хранения одного объекта в реляционной базе данных используется несколько таблиц.

Однако работать приходится именно с объектами. Для решения этой проблемы и созданы реляционные системы управления базами данных. Они должны уметь как обрабатывать данные в объектно-ориентированном виде, так и уметь сохранить эти данные в реляционной форме. ORM (object-relational mapping) – объектно-реляционное отображение, как раз та самая технология программирования.

Почему же тогда не сделать объектно-ориентированную базу данных? Потому что как-то с 80-х годов так и не удалось. Причины:

- **Популярность**. Под реляционные базы создано множество продуктов, которые необходимо поддерживать. В них уже вложено много денег. А с использованием ООБД разработано сравнительно мало серьезных коммерческих продуктов.

- **Язык запросов и его стандартизация.** В 1986 году был принят первый стандарт SQL-86, который определил всю судьбу реляционных БД. Для объектно-ориентированных баз данных пока стандарта языка запросов нет.

- **Математический аппарат.** Для реляционных БД используется математический аппарат реляционной алгебры. Он объясняет, как должны выполняться основные операции над отношениями в базе данных, доказывает их оптимальность. Для ООБД пока нет такого аппарата.

- **Проблема хранения данных и методов.**  В реляционных БД хранятся только голые данные. Что с ними будет делать приложение, зависит уже от приложения. В ООБД же, напротив должны храниться объекты, а объект - это совокупность его свойств и методов. Здесь так же нет единого мнения, как ООБД должна осуществлять хранение объектов и как разработчик должен эти объекты разрабатывать и проектировать. Здесь же возникает и проблема хранения иерархии объектов, хранение абстрактных классов и т.п.

**7. Кластеры (в применении к таблицам в реляционных и объектно-реляционных БД).**

Кластер (англ. cluster) — в СУБД Oracle Database специализированный объект базы данных, используемый для физически совместного хранения одной или нескольких таблиц, которые часто соединяются вместе в SQL-запросах. Кластеры хранят взаимосвязанные строки разных таблиц вместе в одних и тех же блоках данных, что позволяет сократить количество операций дискового ввода-вывода и улучшить время доступа для соединений таблиц, входящих в кластер. После создания кластера в нем можно создавать таблицы. Перед тем как добавлять строки в кластеризованные таблицы необходимо создать индекс кластера.

Кластеры не влияют на проектирование модели данных приложений, их существование прозрачно для пользователей и приложений. Данные, хранящиеся в кластере, обрабатываются с помощью тех же инструкций SQL, что и данные, хранящиеся в некластеризованных таблицах.

Кластеры целесообразно использовать для хранения одной или нескольких таблиц, которые часто используются в запросах и для которых запросы часто выполняют соединение данных из нескольких связанных таблиц, либо извлекают связанные данные из одной таблицы.

Использование кластеров замедляет операции добавления, обновления, удаления строк таблицы по сравнению с хранением таблицы вне кластера со своим собственным индексом. Кроме того кластеры используют дополнительный объём дисковой памяти, поскольку каждая отдельная таблица в кластере занимает больше блоков, чем если бы она хранилась вне кластера. Поэтому перед созданием кластера следует убедиться, что планируемый выигрыш в производительности запросов превысит дополнительную затрату ресурсов на сопровождение кластера.

CREATE CLUSTER employees\_departments\_cluster

(department\_id NUMBER(4))

SIZE 512;

CREATE INDEX idx\_emp\_dept\_cluster ON CLUSTER employees\_departments\_cluster;

CREATE TABLE employees ( ... )

CLUSTER employees\_departments\_cluster (department\_id);

CREATE TABLE departments ( ... )

CLUSTER employees\_departments\_cluster (department\_id);

**ООП**

Объектно-ориентированное программирование особенно подходит для создания повторно используемых компонентов и комплексных приложений. В PL/SQL объектно-ориентированное программирование основано на объектных типах. Объектные типы позволяют моделировать объекты реального мира, разделяя интерфейсы и детали внутренней реализации, позволяют постоянно хранить объектно-ориентированные данные в базе данных.