

Отчет по лабораторной работе № 3
«Создание и модификация компонентов базы данных»

дата	Оценка (max 5)	Бонус за сложность	подпись
------	-------------------	-----------------------	---------

Цели работы:

Изучение основных компонентов и инструментария СУБД Oracle, создание и модификация основных элементов тестовой базы данных

Задачи работы:

- Создание и модификация основных компонентов БД (таблиц, ограничений и т.п.)

Задание повышенной сложности (бонус за сложность – 10 баллов):

Верификация схемы HR

Краткий конспект теоретической части (ответы на контрольные вопросы)

Виды ограничений _____

Создание индексов

Создание индексов

Создание синонимов

Создание представлений	
------------------------	--

CREATE/ALTER TABLE - СОЗДАНИЕ И ИЗМЕНЕНИЯ ТАБЛИЦ

В своей схеме создайте (а потом удалите) тестовую таблицу.

Синтаксическая диаграмма	Тело скрипта
	<pre>CREATE TABLE components (comp_cost INTEGER NULL ,comp_qua INTEGER NULL ,comp_type VARCHAR(20) NULL ,comp_name VARCHAR(20) NOT NULL ,comp_id INTEGER NOT NULL); -- Создание таблицы components</pre>
	<p>Вклеить текст собственного скрипта</p>

** Опция **DEFAULT** - Для столбца может быть задано значение по умолчанию. Если значение поля не указано при создании строки, то значение по умолчанию гарантирует нас от появления в поле пустого значения (или от ошибки, если на это поле задана опция **NOT NULL**). Значением по умолчанию может быть литерал или выражение, не содержащее имен других столбцов. Допускаются системные функции, такие как **SYSDATE** и **USER**.*

Вклеить скрипты создания собственных таблиц в своей схеме для модуля АСУ фирмы (информационная модель разработана в лабораторной работе 7-8 за 7 семестр)

```

CREATE TABLE components (
    comp_cost INTEGER NULL
    ,comp_qua  INTEGER NULL
    ,comp_type VARCHAR(20) NULL
    ,comp_name VARCHAR(20) NOT NULL
    ,comp_id   INTEGER NOT NULL
); -- Создание таблицы components

CREATE TABLE defect (
    dfct_date DATE NULL
    ,dfct_type VARCHAR(20) NULL
    ,dfct_desc VARCHAR(200) NULL
    ,dfct_name VARCHAR(20) NOT NULL
    ,dfct_id   INTEGER NOT NULL
); -- Создание таблицы defect

CREATE TABLE documentation (
    docs_auth VARCHAR(20) NULL
    ,docs_date DATE NULL
    ,docs_type VARCHAR(20) NULL
    ,docs_name VARCHAR(20) NOT NULL
    ,docs_id   INTEGER NOT NULL
); -- Создание таблицы documentation

CREATE TABLE equipment (
    eqpt_check DATE NULL
    ,eqpt_date  DATE NULL
    ,eqpt_type  VARCHAR(20) NULL
    ,eqpt_name  VARCHAR(20) NULL
    ,eqpt_id    INTEGER NOT NULL
); -- Создание таблицы equipment

CREATE TABLE employees (
    empl_num  INTEGER NULL
    ,empl_job  VARCHAR(20) NULL
    ,empl_secn VARCHAR(20) NOT NULL
    ,empl_surn VARCHAR(20) NOT NULL
    ,empl_name VARCHAR(20) NOT NULL
    ,empl_id   INTEGER NOT NULL
); -- Создание таблицы employees

CREATE TABLE failure (
    fail_dfct_id INTEGER NULL
    ,fail_res    VARCHAR(20) NOT NULL
    ,fail_date   DATE NULL
    ,fail_type   VARCHAR(20) NULL
    ,fail_prds_id INTEGER NOT NULL
    ,fail_id     INTEGER NOT NULL
); -- Создание таблицы failure

CREATE TABLE products (
    prds_stat  VARCHAR(20) NOT NULL
    ,prds_id    INTEGER NOT NULL
    ,prds_date  DATE NOT NULL
    ,prds_dfct_id INTEGER NOT NULL
    ,prds_eqpt_id INTEGER NOT NULL
    ,prds_docs_id INTEGER NOT NULL
    ,prds_empl_id INTEGER NOT NULL
); -- Создание таблицы products
    
```

СОЗДАНИЕ ОГРАНИЧЕНИЙ

Oracle позволяет накладывать ограничения на таблицы и столбцы для обеспечения соблюдения некоторых правил как внутри таблицы, так и во взаимосвязи с другими таблицами БД. Допускаются ограничения на двух уровнях: Ограничения на таблицы могут быть определены на один или более столбцов таблицы и у задаются отдельно от описания столбцов таблицы. Ограничения на таблицу могут быть добавлены и после ее создания, а также временно отключены (см. команду ALTER TABLE в следующем разделе),

Все ограничения фиксируются в словаре данных. Каждому ограничению присваивается имя; удобнее давать ограничениям свои имена при их создании, это облегчит ссылку на ограничения в последующем. Если имя не задано, то система сама дает ограничению имя, генерируемое в форме: SYS_Cn, где n -уникальный номер. Ключевое слово **CONSTRAINT** позволяет вам давать ограничениям свои имена.

Типы ограничений

Вы можете задавать следующие типы ограничений:

- NULL/NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY (внешняя ссылка)
- CHECK

Ограничение NOT NULL

Этот ограничитель запрещает указанным полям иметь пустые значения. Столбцы без ограничителя NOT NULL могут иметь пустые значения. NOT NULL - это одно из ограничений целостности, которое может быть задано в описании таблицы.

Привести пример задания данного типа ограничения в своей схеме

```
CREATE TABLE components (  
    comp_cost INTEGER NULL  
    ,comp_qua  INTEGER NULL  
    ,comp_type VARCHAR(20) NULL  
    ,comp_name VARCHAR(20) NOT NULL  
    ,comp_id   INTEGER NOT NULL  
); -- Создание таблицы components
```

Ограничитель UNIQUE

Этот квалификатор объявляет столбец или комбинацию столбцов уникальным ключом. Две разных строки таблицы не могут иметь одинаковое значение ключа. При этом допускаются пустые значения ключевого поля, если ключ задан на один столбец.

Привести пример задания данного типа ограничения в своей схеме

```
CREATE UNIQUE INDEX unq_components_id ON  
    components (  
        comp_id  
    );
```

Ограничитель PRIMARY_KEY

Как и уникальные ключи, первичный ключ (PRIMARY KEY) обеспечивает уникальность значений столбца или комбинации полей столбцов, на которые он задан. Также создается уникальный индекс для поддержания ограничения. Отличие от квалификатора **UNIQUE** заключается в том, что, во-первых, первичный ключ может быть в таблице только один; во-вторых, в нем запрещены пустые значения (NULL). Первичный ключ, таким образом, может служить для однозначной идентификации строк таблицы.

Синтаксис задания на уровне таблицы:

Привести пример задания данного типа ограничения в своей схеме

```
ALTER TABLE components ADD (  
    CONSTRAINT i_components_pk PRIMARY KEY ( comp_id )  
); -- Создание первичного ключа для таблицы components
```

<p>[CONSTRAINT имя ограничения] PRIMARY KEY (столбец, столбец) Синтаксис задания на уровне отдельного столбца:</p> <p>[CONSTRAINT имя ограничения] PRIMARY KEY Заметим, что одна и та же комбинация столбцов не может быть задана одновременно в первичном и уникальном ключах.</p>	
---	--

<p>Ограничитель FOREIGN KEY Внешние ключи (FOREIGN KEY) обеспечивают поддержание целостности данных как внутри одной таблицы, так и на уровне реляционных взаимосвязей различных таблиц. Внешний ключ используется в совокупности с уникальным или первичным ключом (в том смысле, что столбцы, на которые делаются ссылки, должны иметь ограничение UNIQUE или PRIMARY KEY).</p> <p>Синтаксис задания: На уровне таблицы: [CONSTRAINT имя ограничения] FOREIGN KEY (столбец, столбец) REFERENCES таблица (столбец, столбец,...)</p> <p>На уровне отдельного столбца: [CONSTRAINT имя ограничения] REFERENCES таблица (столбец) Заметим, что слова "FOREIGN KEY" не указываются в ограничениях на уровне столбцов.</p>	<p>Привести пример задания данного типа ограничения в своей схеме</p> <pre>ALTER TABLE defect ADD CONSTRAINT c_comp_fk FOREIGN KEY (comp_id) REFERENCES components ; -- Добавление внешнего ключа в таблицу defect</pre>
--	--

* В результате ограничения строка не может быть удалена, их основной таблицы до тех пор пока из дочерней таблице не будут удалены связанные записи. Ограничение можно также задать таким образом, что при удалении информации из основной таблицы, будут удалены и соответствующие записи из дочерней таблицы. Это достигается заданием опции **ON DELETE CASCADE** в спецификации ограничения:

<p>Ограничитель CHECK Ограничение CHECK задает условие, которому должно удовлетворять значение столбца в каждой строке таблицы. Условия, которые могут при этом задаваться, аналогичны заданию критерия поиска в предложении WHERE команды SELECT, но со следующими ограничениями:</p> <ul style="list-style-type: none"> подзапросы не допускаются; ссылки на псевдостолбцы, такие SYSDATE, также запрещены. <p>Синтаксис задания: [CONSTRAINT имя ограничения] CHECK (условие)</p>	<p>Привести пример задания данного типа ограничения в своей схеме</p> <pre>ALTER TABLE products ADD CONSTRAINT p_products_ch CHECK (prds_date >= TO_DATE('01-JAN-2000', 'DD-MON-YYYY'));</pre>
--	---

<p>Создание таблицы на базе запроса к другим таблицам</p> <p>Существует другая форма команды CREATE TABLE, в которой таблица создается на базе команды SELECT - запроса к существующим в БД таблицам:</p> <pre>CREATE TABLE DEPT [(имя столбца, ...)] AS SELECT текст-команды SELECT;</pre>	<p>Привести пример задания данного типа ограничения в своей схеме</p> <pre>CREATE TABLE standarts AS (SELECT * from documentation WHERE docs_type='standart');</pre>
--	--

УПРАВЛЕНИЕ ТАБЛИЦАМИ

<p>Изменение таблицы (ALTER TABLE)</p> <p>Для внесения изменений в описание таблицы используйте команду ALTER TABLE.</p> <p>ADD</p> <p>Для добавления к таблице нового столбца или навешивания на таблицу нового ограничения, задайте ключевое слово ADD:</p> <p>ПРИМЕР: ALTER TABLE EMP ADD (FIRST_NAME CHAR(32));</p> <p>Предложение MODIFY</p> <p>В случае когда необходимо модифицировать описание столбца таблицы применяют ключевое слово MODIFY.</p> <p>ПРИМЕР: ALTER TABLE имя таблицы MODIFY (столбец тип [NULL]);</p>	<p>Привести пример задания данного типа ограничения в своей схеме</p> <pre>ALTER TABLE products ADD CONSTRAINT products_ch CHECK (prds_date >= TO_DATE('01-JAN-2000', 'DD-MON-YYYY'));</pre>
--	---

Существует ряд правил, которые необходимо соблюдать при добавлении и модификации описания столбцов:

- Вы не можете добавлять к столбцу опцию NOT NULL, если в нем есть пустые значения
- Вы не можете добавить новый столбец с опцией NOT NULL. Сначала создайте его без этой опции, заполните его значения во всех строках и затем добавьте опцию NOT NULL.
- Вы не можете уменьшить размер столбца или изменить его тип, если столбец содержит какие-то данные.
- Вы не можете исправлять с помощью опции MODIFY ограничения на таблицы, за исключением опции NULL/NOT NULL. Чтобы исправить другие ограничения, вы должны сначала удалить их, затем создать снова.

Предложение DROP

Задавайте предложение DROP для удаления ограничения из описания таблицы.

Синтаксис:

ALTER TABLE имя таблицы		
DROP	CONSTRAINT имя ограничения	[CASCADE]
	PRIMARY KEY	
	UNIQUE (столбец, столбец,...)	

Опция CASCADE ставится в предложении DROP для удаления ограничений, связанных с тем, которое удаляется.

Предложение ENABLE/DISABLE

Это предложение команды ALTER TABLE позволяет временно включать/выключать действие заданных ограничений, не удаляя их из описания таблицы. Синтаксис:

DISABLE	UNIQUE (столбец, столбец, ...)	[CASCADE]
ENABLE	PRIMARY KEY	
	CONSTRAINT имя ограничения	

Как и в предложении DROP, опция CASCADE означает, что все связанные ограничения также на время отключаются.

ПРИМЕР: ALTER TABLE DEPT DISABLE CONSTRAINT DEPT PRIM CASCADE;

DROP TABLE - УДАЛЕНИЕ ТАБЛИЦ

<p>Синтаксис: DROP TABLE имя_таблицы [CASCADE CONSTRAINTS]</p> <p>Удаление таблицы уничтожает все данные в ней и используемые ею индексы. Опция CASCADE CONSTRAINTS удаляет все внешние ссылки на столбцы таблицы. Без этой опции, при наличии внешних ссылок, таблица не будет удалена.</p> <p>Замечания:</p> <ul style="list-style-type: none">• Все данные в таблице удаляются вместе с ней.• Все синонимы (SYNONYM) и представления (VIEW) на таблицу остаются в словаре данных, но обращаться к ним при этом нельзя.• Все связанные с таблицей незавершенные транзакции закрываются. Если в таблице есть заблокированные строки, то удаления таблицы не происходит.• Только создатель таблицы и системный администратор могут удалить ее.	<p>Привести пример удаления таблицы в своей схеме</p> <pre>DROP TABLE defect CASCADE CONSTRAINTS;</pre>
---	---

COMMENT – ЗАДАНИЕ КОММЕНТАРИЕВ

<p>Синтаксис: Выполняйте команду COMMENT для занесения в словарь данных комментариев о таблицах и их столбцах. Комментарий - строка, содержащая, до 255 символов.</p> <p>COMMENT ON TABLE EMP IS 'Employee Information';</p> <p>Чтобы завести комментарий на столбец таблицы COMMENT ON COLUMN EMP.EMPNO IS 'Unique Employee Number';</p> <p>Чтобы уничтожить комментарий на столбец таблицы, необходимо выполнить команду COMMENT ON COLUMN EMP.EMPNO IS</p> <p>Чтобы просмотреть комментарии на таблицы и их столбцы, следует задать запрос к соответствующим представлениям словаря данных: USER_TAB_COMMENTS или ALL_TAB_COMMENTS - комментарии на таблицы; ALL_COL_COMMENTS или USER_COL_COMMENTS - на столбцы таблиц</p>	<p>Привести пример задания комментариев в своей схеме</p> <pre>COMMENT ON TABLE defect IS "Это таблица дефектов"</pre>
---	--

RENAME ИЗМЕНЕНИЯ ИМЕН ОБЪЕКТОВ БД

RENAME старое_имя TO новое_имя; ВНИМАНИЕ: все предложения/программы/отчеты, ссылающиеся на переименовываемую таблицу по имени, должны при этом быть откорректированы.	Привести пример в своей схеме <code>RENAME documentation to docs;</code>
---	---

TRUNCATE TABLE - УДАЛИТЬ ВСЕ СТРОКИ ИЗ ТАБЛИЦЫ

То же самое может быть сделано командой DELETE, но очистка всей таблицы командой TRUNCATE более эффективна, поскольку производится не через механизм транзакций, а непосредственно, без возможности впоследствии восстановить данные путем отката транзакций. Синтаксис: TRUNCATE TABLE имя_таблицы [REUSE STORAGE] Опция REUSE STORAGE освобождает также память, зарезервированную за таблицей. По умолчанию очищенная область памяти остается за таблицей.	Привести пример в своей схеме <code>TRUNCATE TABLE docs;</code>
---	--

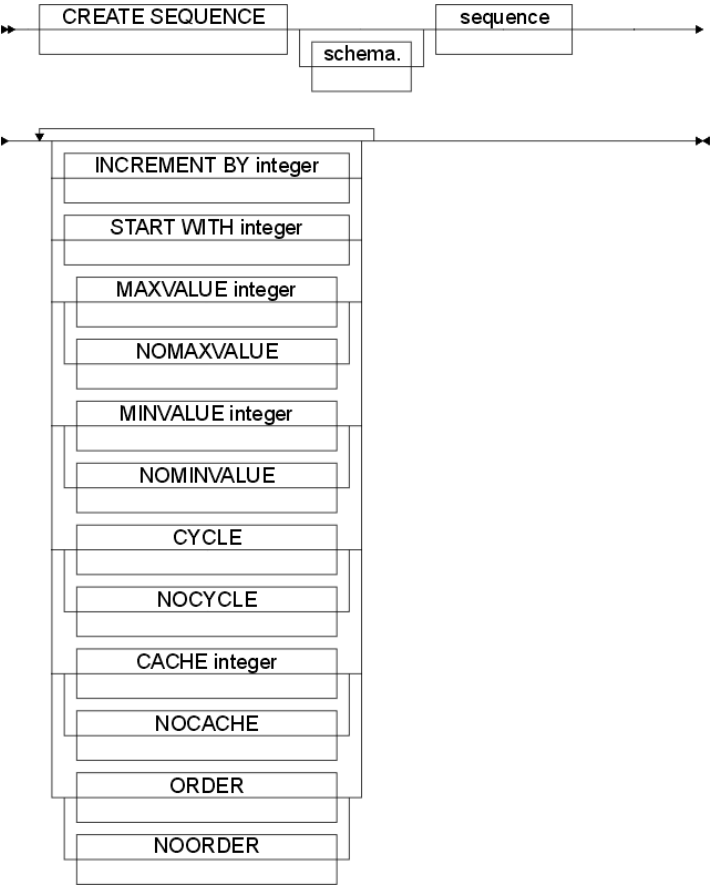
CREATE SYNONYM - СОЗДАЕТ СИНОНИМ

<pre>CREATE PUBLIC SYNONYM schema. synonym; FOR schema. object @dblink;</pre>
Пример: Сначала удаляем публичный синоним для таблицы DOCTORS, а потом его заново создаем. PROMPT Создает синоним для таблицы, представления – вклеить свой PROMPT последовательности, хранимой процедуры или PROMPT функ., пакетной процедуры или др. синонима. DROP PUBLIC SYNONYM doctors; CREATE PUBLIC SYNONYM doctors FOR doctors;
CREATE PUBLIC SYNONYM имя FOR [хозяин.] имя_объекта; Синоним может быть удален командой DROP: DROP [PUBLIC] SYNONYM имя;

Для обращения к таблице другого пользователя вам необходимо к имени таблицы добавить слева в качестве префикса имя ее хозяина, отделенное точкой. Как альтернатива, вы можете создать синоним (второе имя) на таблицу или представление и обращаться к ней по этому имени. Только администратор БД (точнее, пользователь, обладающий привилегией **CREATE PUBLIC SYNONYM**) может создать синоним, доступный всем пользователям.

CREATE SEQUENCE - СОЗДАНИЕ ГЕНЕРАТОРА ПОСЛЕДОВАТЕЛЬНОСТИ

Создает новую последовательность для генерации первичных ключей.



Привести пример в своей схеме

```
CREATE SEQUENCE s_dept
INCREMENT BY 10 START WITH
10 MAXVALUE 10000;
```



```
CREATE SEQUENCE s_documentation
START WITH 100
INCREMENT BY 2
```

```
PROMPT создаем сиквенс и устанавливаем его начальное значение в единицу

CREATE SEQUENCE s_prj_nnn
START WITH 1;
*****

PROMPT получаем номер текущей задачи

SELECT s_prj_nnn.NEXTVAL FROM dual;
```

Сервер возвращает номер текущей задачи «N», поле чего создаем каталог V000N

`CREATE SEQUENCE [пользователь.]имя последовательности [INCRENENT BY n] [START WITH n] [MAXVALUE n I NOMAXVALUE] [MINVALUE n | NOMINVALUE];`

Все стоящие в прямоугольных скобках опции не обязательны для задания и разъясняются ниже.

пользователь	Хозяин последовательности. По умолчанию ставится имя пользователя, выполняющего команду CREATE SEQUENCE.
имя последовательности	Должно удовлетворять стандартным требованиям на имя объекта в языке SQL.
INCREMENT BY	Определяет интервал между соседними элементами последовательности. Если значение шага положительное, то последовательность возрастающая; если отрицательное - то последовательность убывающая. Можно задавать любое не равное нулю целое число. По умолчанию ставится значение 1.
START WITH	Задаёт первое значение последовательности при ее создании. По умолчанию задается 1 для возрастающих последовательностей и MAXVALUE для убывающих.

MINVALUE NOMINVALUE	Минимальное значение последовательности, которое может быть сгенерировано (нижняя граница). По умолчанию задается 1 для возрастающих последовательностей и 10e27-1 для убывающих.
MAXVALUE NOMAXVALUE	Максимальное значение, которое может быть сгенерирована. Задает верхнюю границу последовательности. По умолчанию определяется 1 для убывающих последовательностей и 10e27-1 для возрастающих. Любая попытка сгенерировать число, превышающее верхнюю границу последовательности, вызовет ошибку. (Если вы не используете последовательность для кодирования столбца с уникальными значениями, то при задании опции CYCLE генерация элементов последовательности может быть продолжена со стартового элемента).

После того как последовательность создана, она может использоваться для генерации уникальных числовых кодов.

Генерация последовательных чисел с помощью псевдостолбца NEXTVAL

Псевдостолбец **NEXTVAL** используется для генерации и выбора элементов заданной числовой последовательности. Когда задан псевдостолбец **NEXTVAL**, формируется очередное значение последовательности. Когда генерируется очередной элемент последовательности, ее значение возрастает на шаг независимо от того, произойдет ли закрытие или откат транзакции. Если два пользователя одновременно обращаются к одной последовательности, то один из них получит номер больший, другой - меньший, поскольку каждое обращение генерирует новый номер. Два пользователя никогда не сгенерируют одинаковых номеров, обращаясь к одной последовательности. Некоторые номера последовательности могут оказаться пропущенными, если пользователь не закрывает транзакцию или транзакция прерывается ненормально.

Доступ к номерам последовательности с помощью псевдостолбца CURRVAL

Для выбора номера последовательности, который уже был сгенерирован (текущий номер последовательности), используют псевдостолбец **CURRVAL**. Он возвращает последнее сгенерированное пользователем значение.

Ограничения на псевдостолбцы NEXTVAL и CURRVAL

Псевдостолбцы **NEXTVAL** и **CURRVAL** не могут быть заданы:

- в предложении **SELECT** на представления
- с ключевым словом **DISTINCT**
- одновременно с предложениями **ORDER BY**, **GROUP BY**, **CONNECT BY** или **HAVING** команды **SELECT**
- с операторами **UNION**, **INTERSECT**, **MINUS**
- в подзапросах.

Обращение к последовательностям аналогично обращению к таблицам, их параметры могут корректироваться командой **ALTER**; они могут быть удалены командой **DROP**.

Владелец последовательности может давать привилегии на доступ к ней другим пользователям.

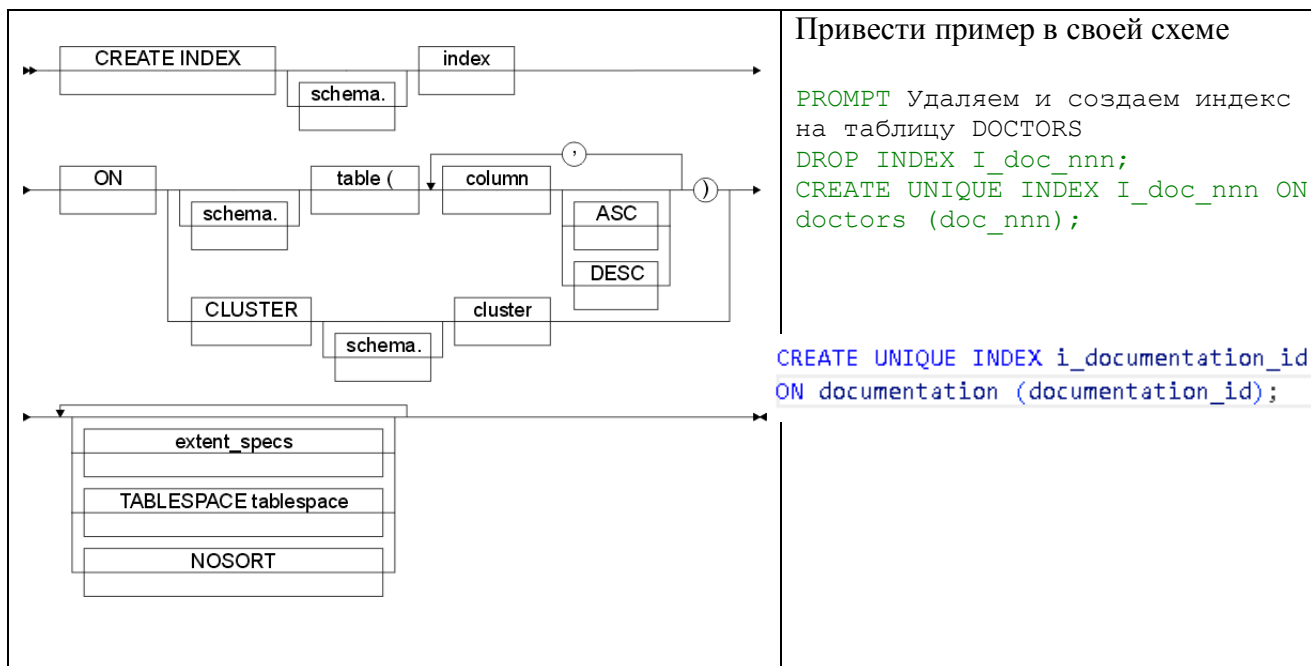
Удаление последовательности

Для удаления описания последовательности из словаря данных используйте команду **DROP SEQUENCE**. Синтаксис команды:

DROP SEQUENCE [пользователь.] имя_последовательности;

Чтобы удалить последовательность, вы должны быть ее владельцем или обладать привилегией администратора.

CREATE INDEX – СОЗДАНИЕ ИНДЕКСА



Индексы Oracle имеют два основных назначения:

1. Ускорить выбор данных по запросам к определенным столбцам.
2. Обеспечить уникальность значений столбца или группы столбцов, используемых обычно в качестве первичного ключа таблицы.

Для повышения производительности системы задание индексов настоятельно рекомендуется, и обычно одним из первых индексов создается индекс на первичный ключ.

Oracle8 автоматически создает индексы на столбцы, которые объявлены как `PRIMARY KEY` или `UNIQUE`.

Владелец таблицы может создавать на нее индексы. Любой пользователь Oracle, имеющий на таблицу привилегию `INDEX`, может также индексировать ее столбцы.

После того как индекс создан, Oracle использует его везде, где можно, для ускорения доступа к данным. Использование индекса производится системой автоматически и не требует от пользователя дополнительных действий; он даже не обязан знать, что индекс существует.

Структура индекса Oracle

Oracle использует сбалансированные двоичные деревья для построения индексов. Это эффективный метод обеспечения примерно одинакового времени доступа к любой строке таблицы независимо от того, находится ли она в начале, середине или конце таблицы. Время доступа также практически не зависит от объема индексируемых данных.

Каждый построенный в БД Oracle индекс состоит из набора страниц памяти, организованных в виде бинарного дерева; каждая страничка содержит набор ключевых значений и указателей на нижестоящие в дереве страницы, и так до тех пор, пока ключевое значение не указывает на само значение поля. Oracle поддерживает эту структуру непосредственно в момент занесения и удаления строк. Пустые значения не хранятся в индексе и не занимают памяти.

Типы индекса

Тип	Термин	Пояснение
Уникальный	UNIQUE	Обеспечивает уникальность значений заданного столбца или комбинации столбцов.
Не уникальный	ON UNIQUE	Обеспечивает самый быстрый доступ к данным при выполнении запроса (по умолчанию)
Простой	INGLE COLUMN	Индекс только на один столбец таблицы
Конкатенированный	ONCATENATED	Индекс на комбинацию столбцов (до 16) для обеспечения уникальности комбинации значений ил повышения эффективности доступа по сцепленному ключу.

Создание индекса Индексы могут быть созданы с помощью команды CREATE INDEX. Синтаксис: CREATE [UNIQUE] INDEX имя индекса ON таблица (столбец1,столбец2,...)	Удаление индекса Для удаления индекса выполните команду DROP INDEX. Синтаксис: DROP INDEX имя-индекса;
Привести пример в своей схеме <pre>CREATE UNIQUE INDEX i_documentation_id ON documentation (documentation_id);</pre>	Привести пример в своей схеме <pre>DROP INDEX i_documentation_id</pre>

Когда используется индекс?

Использование системой индексов частично зависит от того, какой в данный момент применяется оптимизатор. Oracle7 допускает два вида оптимизации выполнения SQL-запроса: на основе правил (**rule-based**) и на основе стоимостной оценки (**cost-based**).

Использование индексов на основе правил

Oracle определяет, когда следует использовать индекс. Oracle имеет информацию о том, какие столбцы проиндексированы и какие типы индексов заданы. На основе этой информации система принимает решение согласно определенным правилам:

- Индексированный столбец должен быть указан в предложении WHERE.
- Индекс не будет использован, если ссылка на столбец в предложении WHERE является частью функции или выражения. В следующем примере индекс не будет использован, так как ссылка на столбец задана в функции: `select * from EMP where UPPER(ENAME)='JONES'`
- индекс на столбец не должна стоять в выражении.

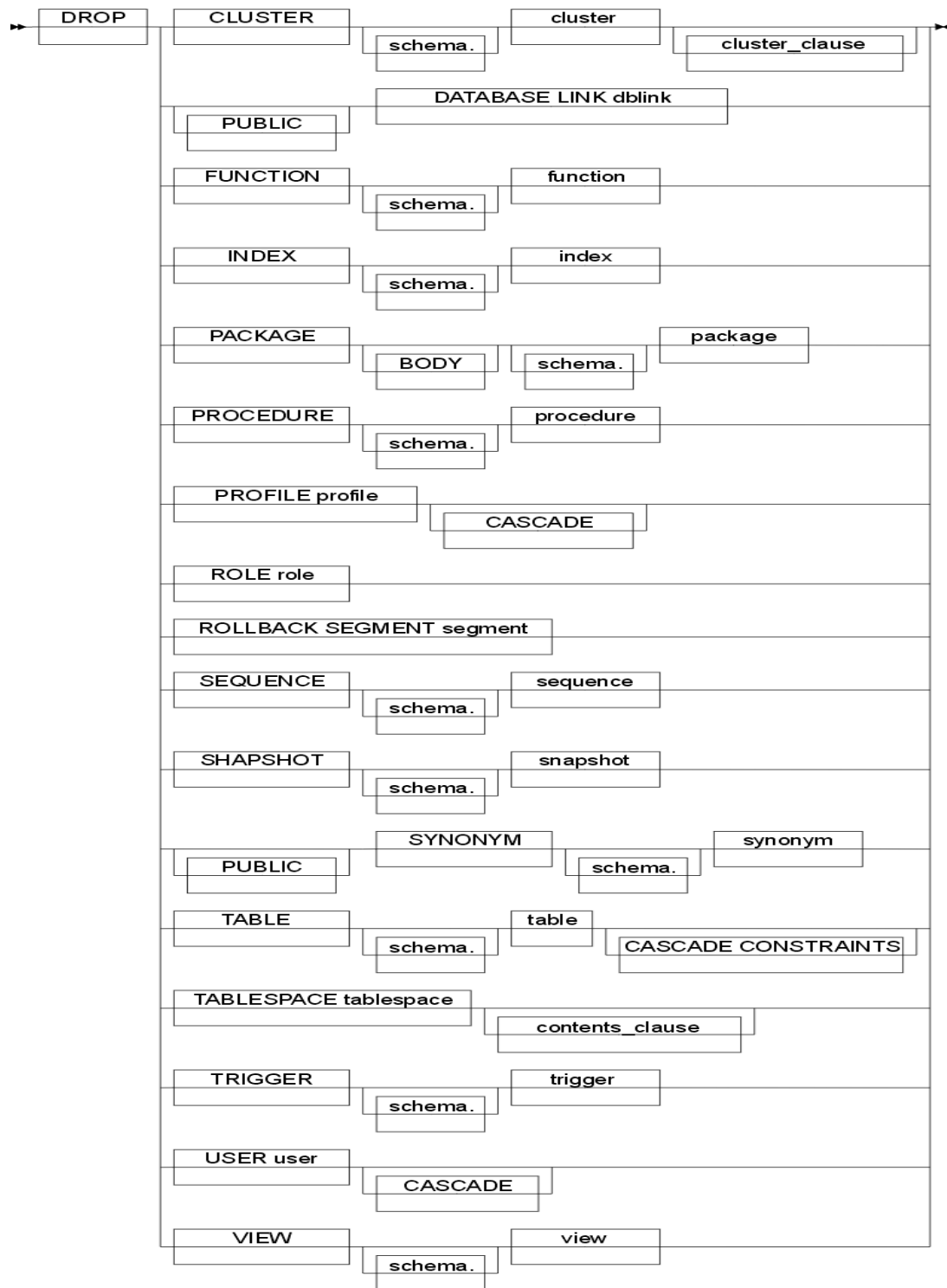
Использование индексов на основе стоимостной оценки

Стоимостной оптимизатор строит план выполнения SQL-запроса путем вычисления стоимости альтернативных путей, используя хранящуюся в базе данных статистику, где это возможно. Обычно стоимостной оптимизатор дает наилучшие результаты оценки порядка задействования индексов и предпочтителен для применения в новых приложениях Oracle7. Тем не менее, этот оптимизатор может использовать индексы только при условии соблюдения приведенных выше правил.

Оптимизатор допускает включение в SQL-команды подсказок пользователя о том, следует ли задействовать определенный индекс.

DROP - УДАЛЕНИЕ ОБЪЕКТОВ И ОГРАНИЧЕНИЙ ИЗ БАЗЫ ДАННЫХ

<p>Эта команда. Для этого действия требуются соответствующие привилегии. Например, для удаления общего канала связи базы данных требуется привилегия</p> <p>Чтобы удалить таблицу из БД, выполните команду DROP TABLE.</p> <p>Синтаксис: DROP TABLE имя_таблицы [CASCADE CONSTRAINTS]</p> <p>Удаление таблицы уничтожает все данные в ней и используемые ею индексы. Опция CASCADE CONSTRAINTS удаляет все внешние ссылки на столбцы таблицы. Без этой опции, при наличии внешних ссылок, таблица не будет удалена.</p> <p>Замечания:</p> <ul style="list-style-type: none"> • Все данные в таблице удаляются вместе с ней. • Все синонимы (SYNONYM) и представления (VIEW) на таблицу остаются в словаре данных, но обращаться к ним при этом нельзя. • Все связанные с таблицей незавершенные транзакции закрываются. Если в таблице есть заблокированные строки, то удаления таблицы не происходит. • Только создатель таблицы и системный администратор могут удалить ее. 	Привести пример в своей схеме <pre>DROP TABLE defect CASCADE CONSTRAINTS;</pre>
--	--



Приведите примеры удаления разных объектов вашей схеме:

```
DROP TABLE defect CASCADE CONSTRAINTS;
```

```
DROP INDEX i_documentation_id
```

EXPLAIN PLAN - Описывает каждый шаг плана выполнения оператора SQL и помещает (если задано) это описание в указанную таблицу

	<p>Привести пример в своей схеме</p> <pre>EXPLAIN PLAN FOR SELECT * FROM documentation; SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);</pre>
--	--

ROLLBACK (управление транзакцией)

Отменяет все изменения, сделанные до контрольной точки. Отменяет все изменения, произведенные в текущей транзакции, если контрольная точка не задана.

	<p>Привести пример в своей схеме</p> <pre>-- Создаем точку сохранения SAVEPOINT after_insert; -- ROLLBACK TO SAVEPOINT after_insert;</pre>
--	---

Контрольные вопросы

1. Виды ограничений?
2. Создание индексов?
3. Создание синонимов?
4. Удаление объектов БД?
5. Формирование плана выполнения SQL оператора?
6. Управление транзакциями?

СПИСОК ЛИТЕРАТУРЫ

1. Власов А.И., Лыткин С.Л., Яковлев В.Л. Краткое практическое руководство по языку PL/SQL - М.: Машиностроение. 2000. 64 с.
2. Сервер Oracle. Справочное руководство по языку SQL / Под ред. А.В.Емельяненко, Н.В. Емельяненко - Протвино, АО РДТех, 1994.
3. Сервер ORACLE. Основные концепции/ Под ред. А.В.Емельяненко, Н.В. Емельяненко - Протвино, АО РДТех, 1996.
4. Проектирование и эксплуатация конструкторско-технологических баз данных на основе СУБД Oracle/ Конспект лекций - М.: Москва, МГТУ им. Н.Э. Баумана, 2001, 120 с.