Oracle Core
Тема 2
Основные концепции

Бобович Павел
Senior developer группы разработки
Oracle

Содержание

- Схема
- Таблица
- Типы таблиц
  - Временные таблицы
- Индекс
  - Типы индексов
- Констрейнт
- Представление
- Последовательность
- Триггер
- Пакет/процедура/функция
- Db Link

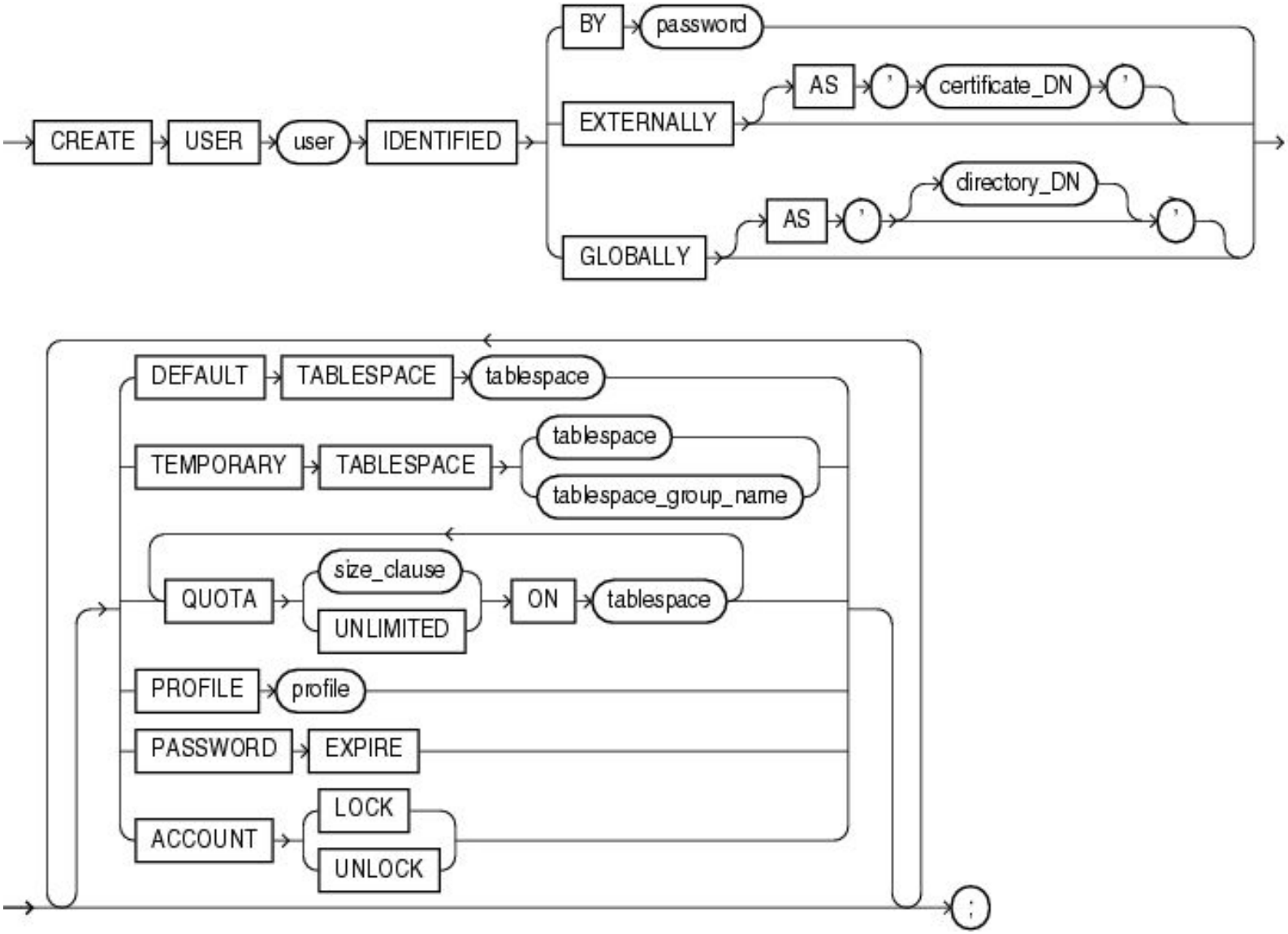
История (основные вехи)

1979 год – Oracle v2. Не поддерживала транзакции, но реализовывала основную функциональность SQL. Первая коммерческая система управления реляционными БД на основе языка запросов SQL
1983 год – v3. поддержка Commit и Rollback для реализации транзакций
1984 год – v4. поддержка средства управления параллельным выполнением операций, такие как
1985 год – v5. Работа в клиент-серверных средах, поддержка распределенных запросов, кластерных таблиц.
1988 год – v6. Добавлена поддержка блокировок на уровне строк и средств горячего резервирования. Появляется поддержка встроенного языка PL/SQL в среде разработки приложений Oracle Forms v3
1992 год – v7. Поддержка ссылочной целостности, хранимых процедур, триггеров
1997 год – v8. Более высокая надежность, поддержка большего количества пользователей и больших объемов данных. Появляется поддержка объектно-ориентированной разработки, секционирование. Oracle становится .
1998 год – 8i “I” – internet, символизируя поддержку Интернета. Начиная с 8.1.5 появляется встроенная в СУБД JVM. На JAVA написаны некоторые клиентские утилиты.
2001 год – v9i. Продвинутая работа с XML, хранящихся в БД через XML DB. Oracle RAC, Oracle Streams – механизма создания репликаций, OLAP и Data Mining, переименование столбцов и constraint-ов.
2004 год – v10g, “g” – grid (сеть), символизирующая поддержку
2007 год – v11g. Появляется возможность создания в БД резидентного пула соединений (DRCP), позволяющего поддерживать пул из постоянных соединений с БД (для веб-серверов Apache, IIS, и т.п.)
2009 год – v11.2 Возможность «горячего», без остановки сервера, внесения изменения в метаданные и бизнес-логику на PL/SQL, сделано с помощью механизма одновременной поддержки нескольких версий схемы и логики, именуемых *editions*
2013 год – v12c. “c” – естественно, означает cloud. Поддержка подключаемых БД, обеспечивающая

Схема

- Схема (schema) – это набор объектов логической структуры базы данных (таблицы, последовательности, представления, снимки, индексы, процедуры и функции, пакеты, синонимы, связи базы данных, триггеры, кластеры и др.).
- Схема ассоциируется с именем пользователя-владельца ее объектов и имеет такое же имя.
- Пользователь имеет доступ ко всем объектам в своей схеме.
- Доступ пользователя к объектам "чужой" схемы возможен при наличии соответствующих привилегий.
- Схема — пространство имен

Схема


CREATE
USER
user
IDENTIFIED
BY
password
EXTERNALLY
AS
'
certificate_DN
'
GLOBALLY
AS
'
directory_DN
'
DEFAULT
TABLESPACE
tablespace
TEMPORARY
TABLESPACE
tablespace
tablespace_group_name
QUOTA
size_clause
UNLIMITED
ON
tablespace
PROFILE
profile
PASSWORD
EXPIRE
ACCOUNT
LOCK
UNLOCK
;

Схема

```
create user TEST_USER
  identified by NOTEST_USER
  default tablespace SOME_TS
  temporary tablespace TEMP
  quota unlimited on large_data
  quota unlimited on large_idx
-- Grant object privileges
   grant select on TEST.TEST to  TEST_USER;
-- Grant role privileges
   grant test_role to TEST_USER;
   grant connect to TEST_USER;
```

Таблица

Реляционные базы данных хранят все данные в таблицах.

Таблица это структура, состоящая из множества неупорядоченных(как правило) горизонтальных строк (rows), каждая из которых содержит одинаковое количество вертикальных столбцов (colums).

Пересечение отдельной строки и столбца называется полем (field), которое содержит специфическую информацию.

Многие принципы работы реляционной базы данных взяты из определений отношений (relations) между таблицами.

Таблица

Всего в Oracle имеется девять основных типов таблиц:

- Традиционные (heap organized table)
- Индекс-таблицы(index organized table - IOT)
- Кластеризованные индекс-таблицы (index clustered table)
- Кластеризованные хеш-таблицы(hash clustered table)
- Отсортированные кластеризованные хеш-таблицы (sorted hash clustered table)
- Вложенные таблицы (nested table)
- Временные таблицы(temporary table)
- Объектные таблицы (object table)
- Внешние таблицы (external table)

Таблица

Традиционные таблицы(heap organized table)

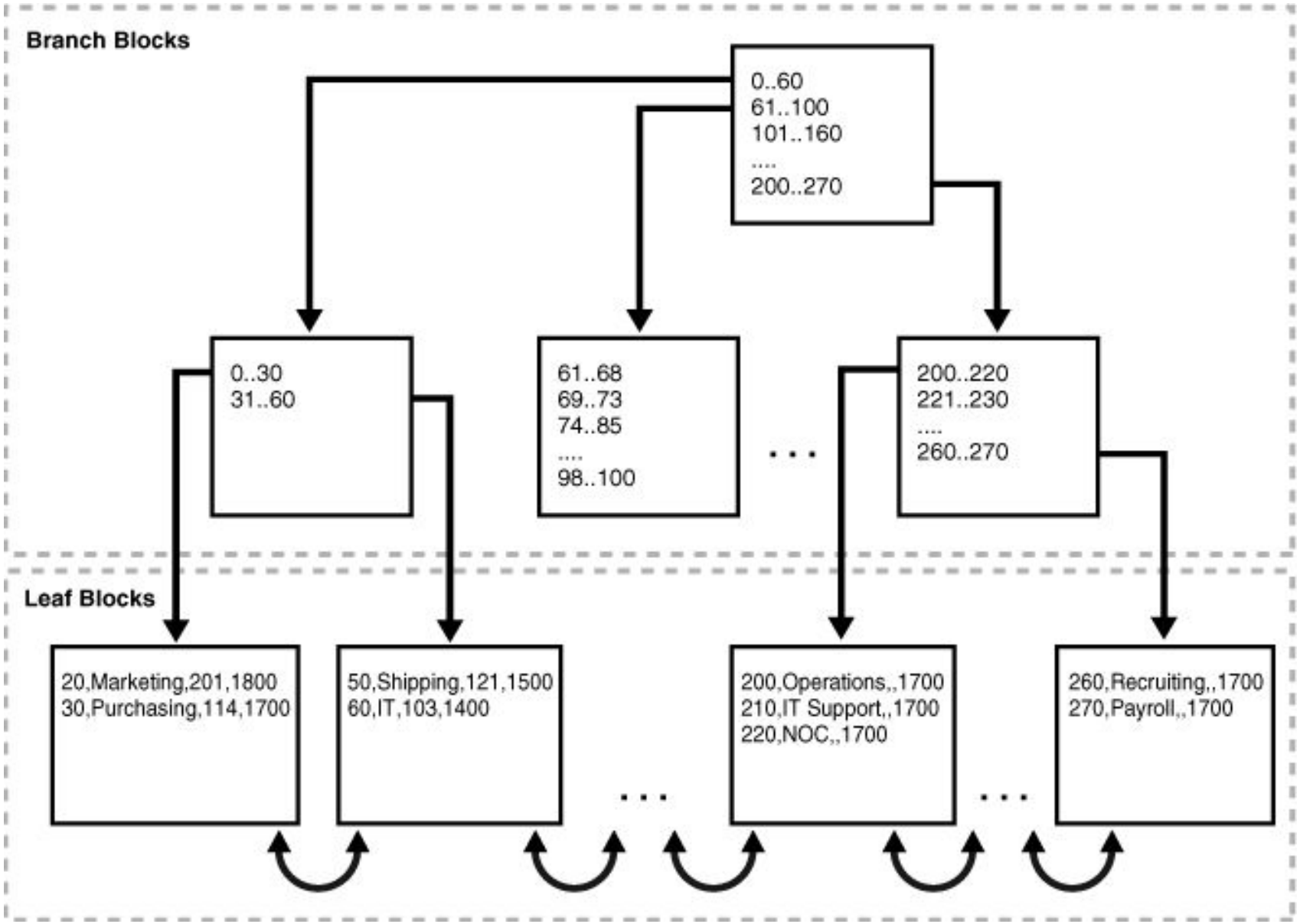
Стандартные таблицы базы данных. Данные распределены подобно куче. При добавлении данных для них используется первое обнаруженное в сегменте и подходящее по размеру пространство. При удалении данных из таблицы пространство, которое они занимали становится доступным для повторного использования последующими операторами INSERT и UPDATE. *Куча* – область пространства, которая используется до определенной степени произвольно

Таблица

Индекс-таблицы(index organized table - IOT)

Эти таблицы хранятся в индексной структуре, что накладывает определенный физический порядок на сами строки. Если в традиционных таблицах данные размещаются там, где они могут поместиться (в первом попавшемся месте), то в индекс-таблицах данные сохраняются в сортированном порядке, в соответствии с первичным ключом.

Таблица


Branch Blocks
0..60
61..100
101..160
....
200..270
0..30
31..60
61..68
69..73
74..85
....
98..100
200..220
221..230
....
260..270
Leaf Blocks
20,Marketing,201,1800
30,Purchasing,114,1700
50,Shipping,121,1500
60,IT,103,1400
200,Operations,,1700
210,IT Support,,1700
220,NOC,,1700
260,Recruiting,,1700
270,Payroll,,1700

Таблица

Кластеризованные индекс-таблицы(index clustered table)

Кластеры – это группы, состоящие из одной или более таблиц, которые физически хранятся в одних и тех же блоках базы данных, и все строки в которых разделяют общее значение ключа кластера и физически находятся близко друг к другу.

Во-первых, множество таблиц могут храниться вместе с физической точки зрения, что удешевляет стоимость запросов, которые получают, в подавляющем большинстве случае, данные из объединения этих таблиц.

Во-вторых, она позволяет хранить рядом все данные, содержащие одинаковое значение ключа кластера. Данные кластеризуются вокруг этого значения ключа кластера, ключ создается с помощью индекса B-tree

Таблица

Кластеризованные хеш-таблицы(hash clustered table)

Эти таблицы похожи на кластеризованные индекс-таблицы, но вместо того, чтобы использовать индекс B-Tree для определения местонахождения данных по ключу кластера, хеш-кластер хеширует ключ кластера, чтобы найти блок базы данных, в котором должны располагаться данные. В хеш-кластере данные – это (образно говоря) и есть индекс. Такие таблицы подходят для хранения данных, которые читаются часто с помощью применяемой к ключу операции сравнения типа «равно».

Таблица

Отсортированные кластеризованные хеш-таблицы (sorted hash clustered table)

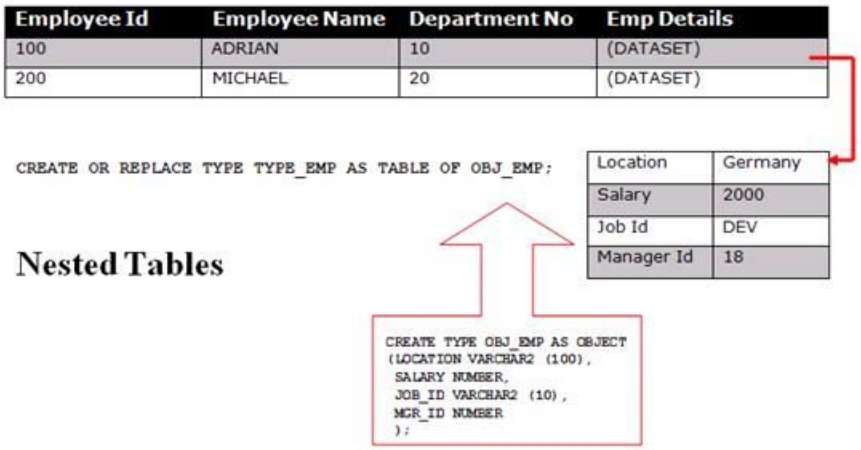
Эти таблицы появились с 10-й версии и сочетают качества кластеризованных хеш таблиц и индекс-таблиц. Концепция: имеется некоторое ключевое значение, по которому должны хешироваться строки, и ряд связанных с этим ключом записей, которые должны поступать в отсортированном порядке и обрабатываться в таком же отсортированном порядке.

В такой системе отсортированный хеш-кластер может оказаться наиболее подходящей структурой.

Таблица

Вложенные таблицы(nested table)

Эти таблицы – часть объектно-реляционных расширений Oracle. Они представляют собой просто генерируемые и обслуживаемые системой дочерние таблицы, состоящие в отношениях «родитель-дочерний». Вложенная таблицы в этом случае не автономная таблица



Таблица

Временные таблицы(temporary table)

Эти таблицы хранят временные данные на протяжении транзакции или сессии (on commit delete rows/on commit preserve rows). При необходимости они выделяют временные сегменты из временного табличного пространства текущего пользователя. Каждый сеанс видит только свои экстенты, он никогда не будет видеть данные созданные любым другим сеансом.

Таблица

Внешние таблицы(external table)

Данные этих таблиц не хранятся в самой базе данных, они располагаются за пределами БД в обычных файлах операционной системы. Внешние таблицы позволяют запрашивать файл, хранящийся вне БД так, будто бы он является обычной таблицей внутри БД. Они наиболее полезны в качестве средства помещения информации в БД (являются очень мощным инструментом загрузки данных). Более того, с версии 10g, в которой появилась возможность выгрузки данных во внешние таблицы, они позволяют легко переносить данные из одной БД Oracle в другую, не используя связи БД. В вот не фига, это бага в доке. В новой доке уже все исправили. Загружать данные можно, выгружать нет!

Особенности таблиц

•До 1000 столбцов (увы, надо быть скромнее) даже более 254 не стоит использовать (это вам не oracle siebel), иначе храниться будут строки в виде отдельных фрагментов

•Таблица может иметь практически неограниченное количество строк. На самом деле ограничения есть, вот TS может содержать 1022 файла (без учета TS BIGFILE). И если вы используете файлы по 32 Гб, и строки от 80 до 100 байт, то не заморачивая всех тут вычислениями получаем всего лишь 342 миллиарда строк ☹. А если секционировать на 1024 секции, то 342 триллиона строк. Но напрягаться такому скромному количеству строк не стоит, в жизни вы быстрее из-за других ограничений поймете что что-то пошло ни так.

•Таблица может иметь столько индексов, сколько в ней существует перестановок столбцов, и перестановок функций на столбцах. Однако, ограничения другого плана (вставка) заставят вас отказаться от такого кол-ва индексов раньше, чем вы переберете все возможные комбинации.

•Нет ограничений и на количество таблиц, в одной БД (см. систему SAP, Парус)

rowid

Rowid – псевдостолбец, возвращающий адрес строки, содержит информацию, по которой можно найти строчку:

-object_id (dba_objects.object_id)

-Блок данных в дата файле в котором валяется строка

-Позиция строка в блоке данных (первая строка – это 0)

-Дата-файл, в котором живет строка (а тут файлы нумеруются с 1, сюрприз, сюрприз). Номер файла по отношению к TS.

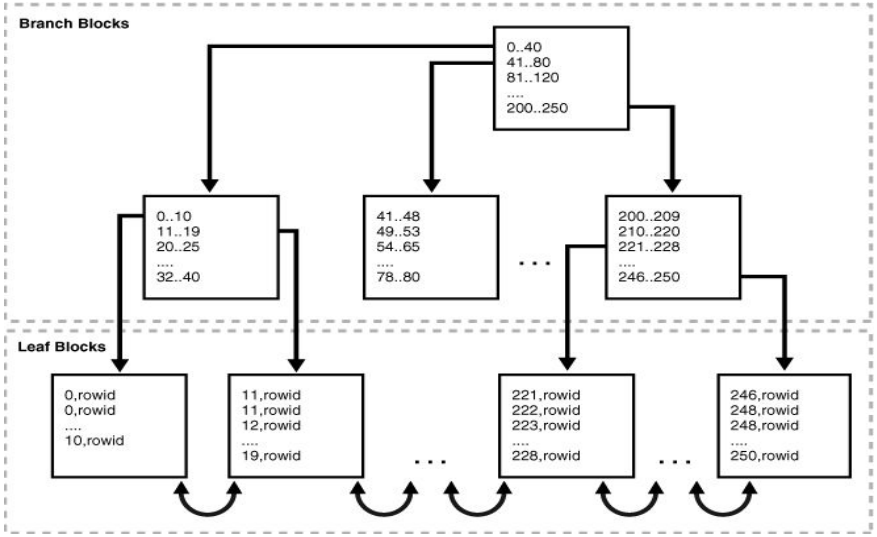
Обычно rowid уникально идентифицирует строку в БД. Однако, строки в разных таблицах хранящиеся вместе в одном кластере могут иметь одинаковый rowid

rowid

```
SQL> SELECT dbms_rowid.rowid_object(ROWID) o_id,
  2         dbms_rowid.rowid_row_number(ROWID) r_n,
  3         dbms_rowid.rowid_block_number(ROWID) b_n,
  4         dbms_rowid.rowid_relative_fno(ROWID) f_n
  5  FROM   sh.tbl cr
  6  WHERE  rownum < 2;
     O_ID        R_N        B_N        F_N
--------- ---------- ---------- ----------
   317822          8     257059        404
```

Индекс

Индексы - это механизм для улучшения быстродействия поиска данных. Индекс определяет столбцы которые могут быть использованы для эффективного поиска и сортировки в таблице.
Однако, стоит понимать, что если индексов слишком много, то пострадает производительность модификации данных (insert/update/delete/merge).
Когда индексов слишком мало – страдает производительность операций DML(insert/update/delete).
Задача – найти оптимальную пропорцию для производительности.



Индексы

- Индексы со структурой B-дерева
  - Индекс-таблицы
  - Кластерные индексы со структурой B-дерева
  - Индексы упорядоченные по убыванию
  - Индексы по реверсивным ключам
- Битовый индексы
- Битовые индексы соединений
- Функциональные индексы
- Индексы предметной области

Индексы

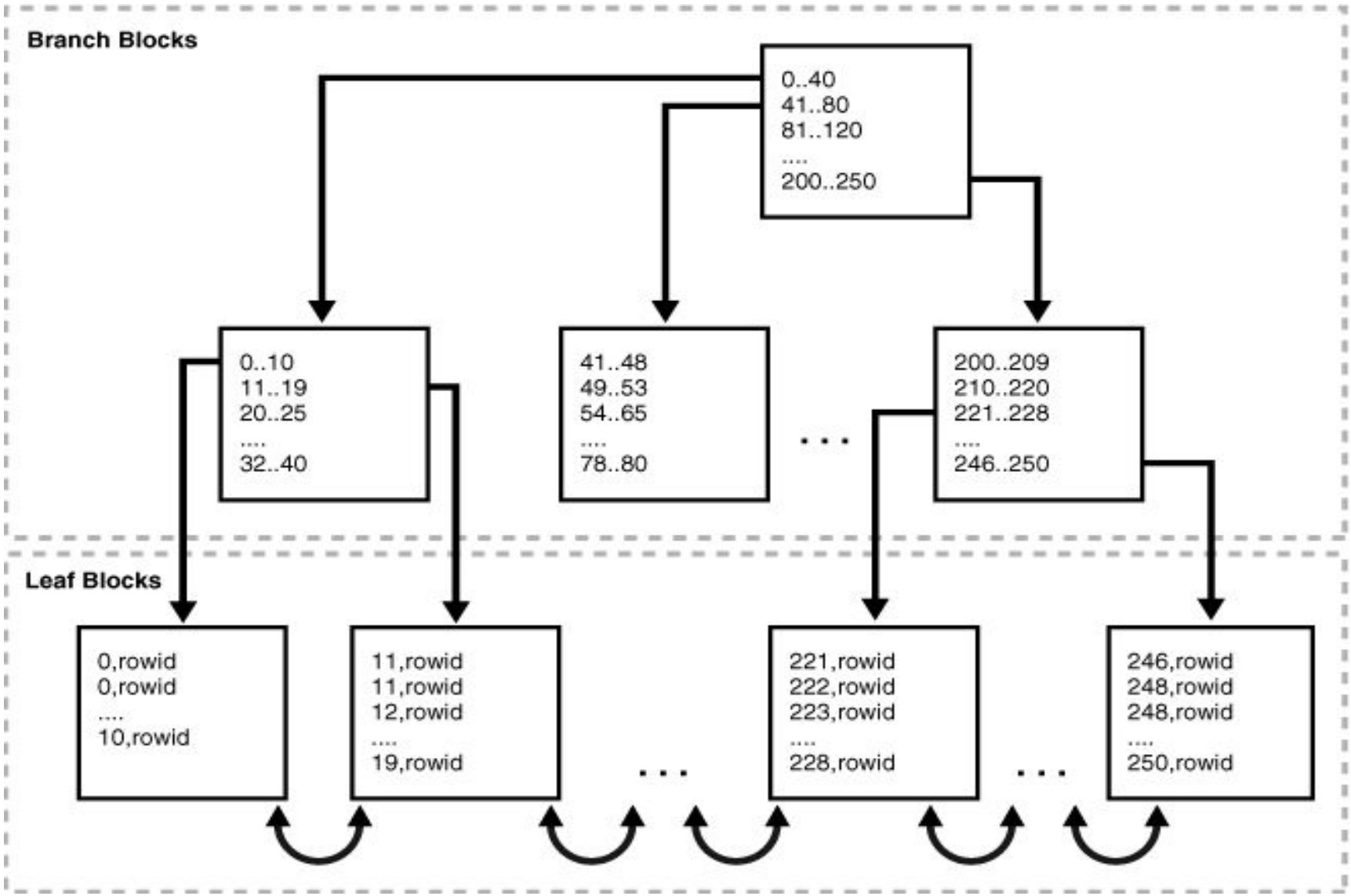
Индексы со структурой B-дерева, там называемые обычные индексы.

Самые распространенные в OLTP-системах.

Подобен двоичному дереву, B-Tree индекс предоставлять быстрый доступ по ключу к индивидуальной строке или диапазону строк, обычно требуя нескольких операций чтения для нахождения нужной строки.

“B” - это НЕ binary(двоичный), a balanced (сбалансированный)

Индексы


Branch Blocks
0..40
41..80
81..120
....
200..250
0..10
11..19
20..25
....
32..40
41..48
49..53
54..65
....
78..80
200..209
210..220
221..228
....
246..250
Leaf Blocks
0,rowid
0,rowid
....
10,rowid
11,rowid
11,rowid
12,rowid
....
19,rowid
221,rowid
222,rowid
223,rowid
....
228,rowid
246,rowid
248,rowid
248,rowid
....
250,rowid

Индексы

Индекс-таблицы (IOT) Это таблицы, хранящиеся в структуре B-Tree. В то время как строки данных в традиционной таблице сохраняются как в куче, данные в индекс-таблице сохраняются и сортируются по первичному ключу. С точки зрения приложения индекс-таблицы ведут себя как «обычные» таблицы, для доступа используется обычный SQL.

Кластерные индексы со структурой B-дерева (B-Tree cluster index)- это небольшая вариация обычных B-Tree индексов. Используются для индексации кластерных ключей. Вместо ключа, указывающего на строку, эти индексы имеют кластерные ключи, указывающие на блок, который содержит строки, объединенные по кластерному ключу.

Индексы, упорядоченные по убыванию(desc index) – это индексы, которые позволяют данным быть отсортированным по убыванию в отличии от порядка по возрастанию, принятого в индексной структуре.

Индексы

Индексы по реверсивным ключам – Это B-tree индексы, в которых байты ключа обращены. Применяются для достижения более равномерного распределения элементов в индексах, которые наполняются в порядке возрастания.
Пример:
Если мы для PK используем последовательность и получаем значения 987500,987501,987502… то они монотонна, и при применении обычного индекса B*Tree они имеют тенденцию поступать в один и тот же правосторонний блок, увеличивая его содержимое. При использовании реверсивного индекса мы получаем 20578,105789,005789, в итоге значения , которые до обращения следовали бы в индекса друг за другом, оказываются разбросанными. Обращение байтов индекса рассеивает вставляемые ключи по множеству блоков.

Битовые Индексы (DWH)

Битовые индексы (bitmap index). B-tree – это отношение «один к одному» между элементом индекса и строкой. Тут как раз наоборот, используется битовая карта для указания на множество строк одновременно. Идеально для данных с высокой повторяемостью (DWH), которые в основном доступны только для чтения. Не используются в OLTP.

Битовые индексы соединений (bitmap join index). Средство денормализации данных в индексной структуре вместо таблицы. Это битмап индекс для соединения двух или более таблиц. Хранит результат джойна, т.о. сам джойн БД уже делать не надо. Так же не используется в OLTP

Индексы

Функциональные индексы (function-based index). Это индексы B-tree (или битовые) которые хранят вычесленный результат функции по значению столбца(ов). Это как индексу по виртуальному столбцу, который не хранится в БД. Используем в select-ах так как написано в индексе upper(name)=:p_name.

Индексы предметной области (application domain index). Эти индексы вы строите и сохраняете самостоятельно, либо в СУБД, либо за ее пределами. ВЫ сообщаете оптимизатору, насколько индекс селективен, насколько дорого его использование, а оптимизатор на основе этой информации решает использовать его или нет

Индексо-мания

Connected as …..

```
SQL> create table dbadmin.del_me10 (id number);
Table created
SQL> create index ni_1234_4321 on DEL_ME10 (id);
Index created
SQL> alter index ni_1234_4321 MONITORING usage;
Index altered
SQL> insert into dbadmin.del_me10 values(1);
1 row inserted
SQL> commit;
Commit complete
SQL> SELECT table_name, index_name, monitoring, used FROM   v$object_usage t WHERE  t.index_name
= 'NI_1234_4321';
TABLE_NAME                     INDEX_NAME                     MONITORING USED
------------------------------ ------------------------------ ---------- ----
DEL_ME10                       NI_1234_4321                   YES        NO
SQL> SELECT * FROM   dbadmin.del_me10 WHERE  id = 2;
        ID
----------
SQL> SELECT table_name, index_name, monitoring, used FROM   v$object_usage t WHERE  t.index_name
= 'NI_1234_4321';
TABLE_NAME                     INDEX_NAME                     MONITORING USED
------------------------------ ------------------------------ ---------- ----
DEL_ME10                       NI_1234_4321                   YES        YES
SQL> ALTER INDEX ni_1234_4321 MONITORING USAGE;
Index altered
```

Констрейнт

Констрейнт – правило которое ограничивает значения в БД.
Oracle позволяет вам создать 6 типов контсрейнтов и позволяет вам описать их двумя способами.

Великолепная шестерка:

- NOT NULL **constraint**
- **Unique constraint**
- **Primary key constraint** = NOT NULL + unique
- **Foreign key constraint** – требует, чтобы значения в одной таблице соответствовали значениям в другой таблице
- **Check constraint** – требует, чтобы значение в БД соответствовало указанному условию
- **REF constraint** позволяет вам в будущем описать связь между REF column и объектом на который он ссылается

Синтаксически два пути:

- Как часть описания колонки или атрибута, это называется **inline** specification
- Как часть описания таблы – **out-of-line** specification

Констрейнт

```
create table sh.ACCOUNT_MOVE
(
  id          NUMBER not null,
  id_account  NUMBER not null,
……….
alter table sh.ACCOUNT_MOVE
  add constraint PK_ACCOUNT_MOVE primary key (ID)
……….
alter table sh.ACCOUNT_MOVE
  add constraint FK_ACCNT_MOVE_ID_ACCNT foreign key
(ID_ACCOUNT)
  references sh.ACCOUNT (ID);
……….
alter table sh.ACCOUNT_MOVE
  add constraint CK_ACCNT_MOVE_TYPE
  check (move_type IN ('+', '-'));
```

Представление

Представление (view) – виртуальная таблица, представляющая собой поименованный запрос(синоним к запросу), который будет представлен как подзапрос при использовании представления.

Таблицы на которых основана VIEW называются базовыми таблицами.

Допускают DML, но имеется масса ограничений, поэтому стараются использовать view как проекцию таблицы БД, стараясь не добавлять туда никакую БЛ.

Пример:

```
CREATE OR REPLACE VIEW sh.vi_some_view AS
SELECT id,
       name,
       ….
FROM   sh.some_table;
```

Последовательность

Сиквенс(sequence) – объект БД, используя который пользователи могут генерировать уникальные числа. Как правило используется для генерации PK.

Когда сиквенс сгенерировал число, значение увеличивается в не зависимости от того был commit или rollback.

Если два пользователя конкурентно увеличивают один и тот же сиквенс, то значения сиквенса, которые получат два пользователя могут иметь дыры.

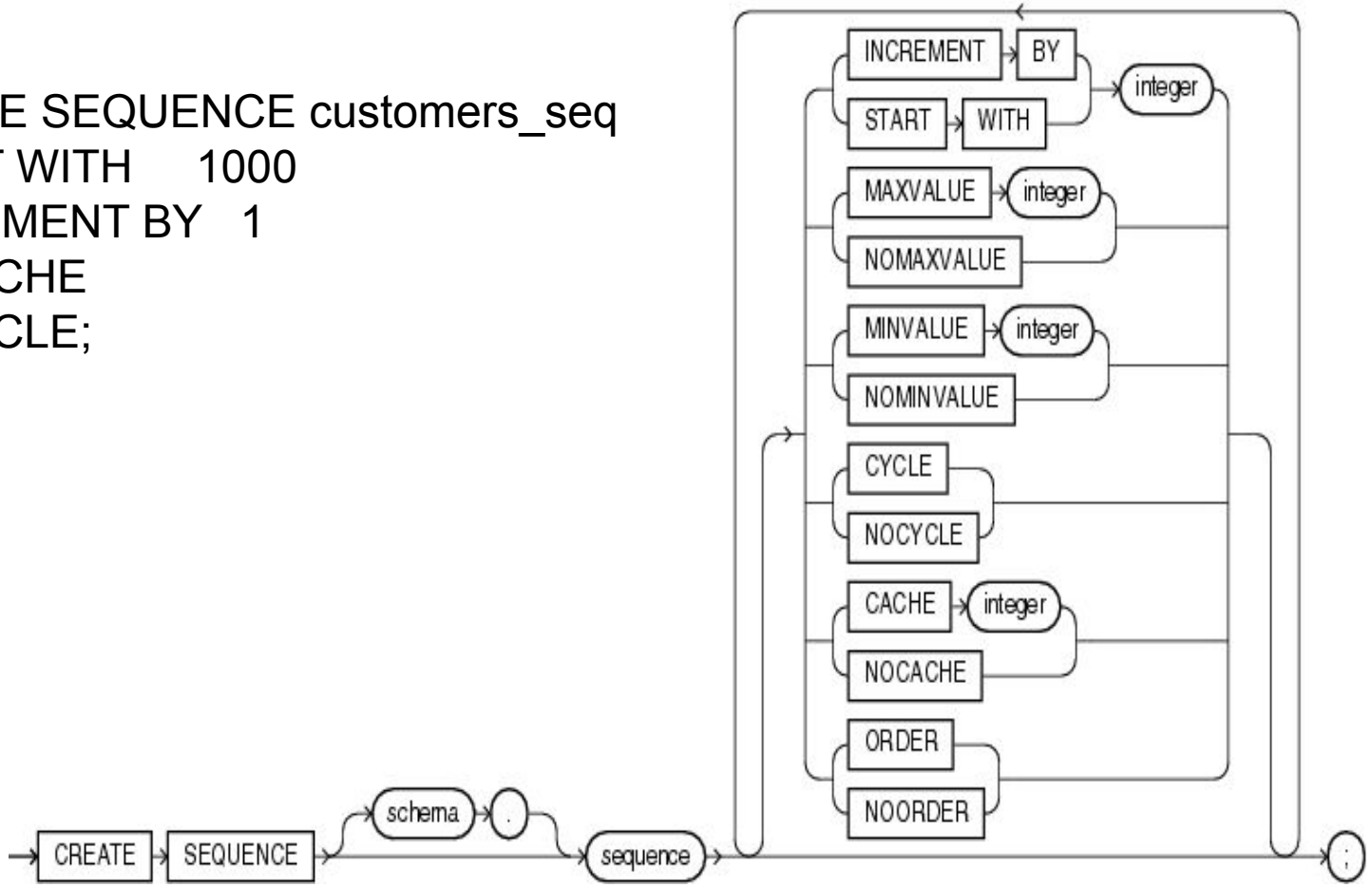
Один пользователь никогда не получит число сгенерированное для другого пользователя.

Сиквенсы не привязаны к таблице, вы можете использовать один сиквенс для нескольких таблиц.

После создания вы получаете значение используя CURRVAL, NEXTVAL.

Последовательность

```
CREATE SEQUENCE customers_seq
 START WITH     1000
 INCREMENT BY   1
 NOCACHE
 NOCYCLE;
```



Триггер

Триггер – фрагмент PL/SQL , который будет выполнен БД Oracle когда указанное условие будет исполнено.

Как только вы создали триггер он включается автоматически, при этом вы можете его перевести в DISABLE или ENABLE.

Вы можете места внедрения триггеров:

- Before – перед событием
- After – после события
- Instead of (только для VIEW) – вместо события. Да да, ВМЕСТО выполнения вашего update-а (к примеру) будет выполнен код триггера.

И следующие события:

- DML
- Ddl_event
- Database_event

Процедура/функция

Процедура/функция – набор PL/SQL операторов, которые вы можете вызывать по имени.

Хранимые процедуры и функции имеют преимущества в разработке, целостности, безопасности, производительности и распределения памяти.

Отличие функции от процедуры только в наличии возвращаемого значении (аналоги int/void JAVA в описании метода).

Функции могут использоваться как часть SQL-выражения (здесь подразумевается, что вы не используете часть параметров, как выходные).

Deterministic – указывается в названии, чтобы показать, что функция возвращает одно и то же значение для одних и тех же значений входных аргументов.

Для использования в функциональных индексах указание этого слова обязательно.

Процедура/функция

```
SQL>
SQL> CREATE OR REPLACE FUNCTION mult(n NUMBER) RETURN NUMBER IS
  2  BEGIN
  3      RETURN power(n, 2);
  4  END;
  5  /
Function created
SQL> select mult(5) from dual;
   MULT(5)
----------
        25
```

Пакет

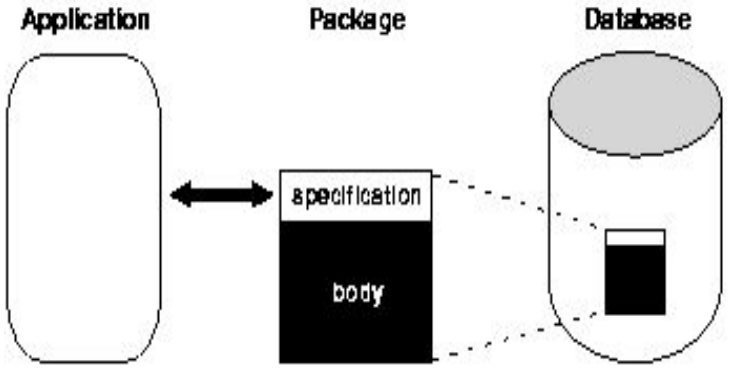
Пакет – объект схемы, который группирует логически связанные PL/SQL типы, константы, подпрограммы (функции/процедуры). *Обычно*, пакет имеет две части, спецификацию и тело, хотя тельце иногда и не обязательно.

Спецификация (spec)– это интерфейс для вашего приложения. Описывает типы, переменные, константы, исключения, курсоры, и подпрограммы доступные для использования вовне.

Тело(body) – реализация спецификации.

Очень образно в понятиях джавы. Spec- интерфейс, body – реализация

Описанное в body и не описанное в Spec равносильно модификатору private и не видно снаружи пакета



Преимущества пакетов

Модульность – инкапсуляция логически связанных типов, элементов, подпрограмм в именованном PL/SQL модуле. Каждый пакет легок в понимании и интерфейс между пакетами также прост.

Легкость дизайна приложения – во время дизайна все, что вам надо сделать в начале – только интерфейс в спецификации пакета. Кодируете и компилируется spec без body. Потом вы можете ссылаться на этот пакет и все компилируется.

Дополнительная функциональность – публичные переменные пакета и курсоры доступны в течении сессии. Т.о. они могут быть разделяемыми между подпрограммами. Вы можете передавать данные между транзакциями БЕЗ сохранения данных в БД.

Преимущества пакетов

Лучшая производительность – Когда вы вызываете подпрограмму пакета первый раз, весь пакет загружается в память. Позже, все вызовы подпрограмм пакета не потребуют операций ввода-вывод с диска.

Сокрытие информации - С пакетами, вы можете указать, какие типы, элементы и подпрограммы являются public или private. Например, если пакет содержит 4 подпрограммы, 3 могут быть public и 1 private. Пакет скрывает реализацию части подпрограмм, так что только пакет (не приложение) влияет на изменение реализации. Это упрощает сопровождение и развитие. Кроме того, скрывая детали реализации от пользователей, вы обеспечиваете целостность пакета.

DBlink

DBLink – объект схемы в одной БД который позволяет получить доступ к объектам в другой БД. Другая БД не обязательно должна быть БД Oracle. Правда для не Oracle БД нужен Oracle Гетерогенный сервис.

После создания линка мы можем использовать ссылки на таблицы, вьюхи, PL/SQL объекта в другой БД добавляя @dblink_name к таблицам, вьюхам, pl/sql объектам.

Select/insert/update/delete – все это доступно с удаленными объектами.

Линки можно использовать в select , комбинирую таблицы из одной БД и из другой.

Пример:

```
CREATE OR REPLACE VIEW sh.my_view AS
SELECT id,
      data1,
      data2,
      …..
FROM   remote_table@my_db_link;
```

Schema Object Names and Qualifiers

Объекты схемы разделяющие один namespace:

- Tables
- Views
- Sequences
- [illegible]
- [illegible]
- [illegible]
- [illegible]

Объект[illegible]

- [illegible]
- [illegible]
- Clusters
- Database triggers
- Private database links
- Dimensions

Следующие вне схемные объекты также имеют свои namespace-ы:

- User roles
- Public synonyms
- Public database links
- Tablespaces
- Profiles
- Parameter files (PFILEs) and server parameter files (SPFILEs)

Summarazing

- Схема
- Таблица
- Rowid
- Индекс
- Констрейнт
- Представление
- Последовательность
- Процедура и функция
- Пакет и тело пакета
- DBLINK
- Schema Object Names and Qualifiers